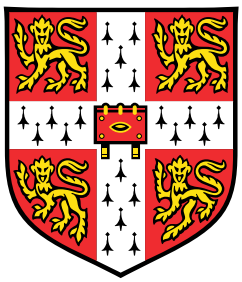
Global Inducing Point Posterior Approximations for Federated Bayesian Neural Networks



Maximiliaan Olivier Jean Bronckers

Supervisors: Prof. Richard Turner
Matthew Ashman

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Lucy Cavendish College August 2022

This thesis—a body of original research—is to me the culmination of my formal education. I, therefore, would like to dedicate this work to my friends and family, close-by or far away, who have supported me in my intellectual pursuits over the past years.

Declaration

I, Maximiliaan Olivier Jean Bronckers of Lucy Cavendish College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this thesis and the work described in it are my own work, unaided except as may be specified below, and that the thesis does not contain material that has already been used to any substantial extent for a comparable purpose. This thesis contains fewer than 15,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

All software used for the purpose of this thesis was written in Python and its standard libraries from scratch by myself, except where indicated otherwise.

Word count: 14,070.

Maximiliaan Olivier Jean Bronckers
August 2022

Acknowledgements

I would like to thank my supervisor Professor Richard Turner and co-supervisor Matthew Ashman, for the guidance I received under their supervision. The ability to talk through problems and brainstorm in the frequent meetings after working on something for a long time was an incredibly helpful to becoming unstuck. Thank you for being so generous with your time, especially outside working hours.

And Matt, thank you for your help in the beginning of the project. Our initial pair-programming sessions were one of the greater learning experiences of the past year.

I would also like to thank Vincent Fortuin for his involvement and help through comments and discussions over the past few months.

Finally, thanks to my friends in Cambridge that shaped my past year. To the Blues (CUHC), thank you for the many great memories both on and off the pitch—you were a large part of my Cambridge experience. To the Stymies (CUGC), what a special week we had at S&A—memories of a lifetime. I still do not know how we all managed to squeeze in as much golf as we did.

Abstract

In federated learning, we seek to train a predictive model when the training data is sharded across (possibly many) different clients or users' devices. One attractive model type is a Bayesian Neural Network (BNN) as it combines the predictive capacity of a neural network with natural regularization and uncertainty estimates by having distributions over the model parameters. Since the posterior distribution over the BNN's weights is intractable, we specify an approximating variational posterior distribution that is tractable and optimize this approximate posterior with respect to the training data using Variational Inference (VI). The structure of this approximating posterior consequently affects the performance of our model. A standard—but simplistic—approximating posterior is the mean-field variational approximation (MFVI). Ober and Aitchison [2021] recently proposed a posterior approximation method based on global inducing points (GI) that outperformed the MFVI method in standard Bayesian learning settings.

In this work, we extend the GI variational approximation for BNNs to the federated learning setting and evaluate its performance. In the first part of this thesis, we provide a more in-depth background for the problem set-up described above. We then develop the theory necessary to extend the GI method to federated BNNs and provide a theoretical analysis of the communication and computational complexity of a federated BNN with GI as approximate posterior. In our evaluations, we compare it to using a standard MFVI approximation. In the second part of this thesis, we complete an empirical evaluation of GI and MFVI in BNNs under various federated learning settings. Due to the computational complexity of Bayesian federated learning, we limit our experiments to shallow BNNs with a small number of clients and medium-scale datasets. We evaluate said methods based on their predictive performance, communication and time efficiency, and convergence dynamics across both homogeneous and inhomogeneous client data distributions. We demonstrate that, across every setting, GI outperforms MFVI in all aspects except for training time efficiency, due to its computational complexity.

Table of contents

List of figures

List of tables

Nomenclature

Other Symbols

$\mathbb{R}$ Real numbers

$\mathcal{N}$ Gaussian distribution

Acronyms / Abbreviations

BNN Bayesian Neural Network

e.g. Exempli gratia ("For the sake of an example")

ELBO Evidence lower bound

ELL Expected log-likelihood

GI Global Inducing Point posterior approximation

i.e. Id est ("it is")

IID Identically and independently distributed

KL Kullback-Leibhler

MC Monte Carlo

MCMC Monte Carlo Markov Chain

MFVI Mean-Field Variational Inference

PVI Partitioned Variational Inference

VI Variational Inference

w.r.t. with respect to

Notation

We use the following notation throughout this thesis. Bold lower case letters ($\mathbf{x}$) denote vectors, bold upper case letters ($\mathbf{X}$) denote matrices, and standard weight letters (x) denote scalar quantities. We use subscripts to denote either entire rows / columns (with bold letters, $\mathbf{x}_i$), or specific elements (x_{ij}). We use subscripts to denote variables as well (e.g. $\boldsymbol{W}_1 : M \times D$), with corresponding lower case indices to refer to specific rows / columns. We refer to the element index of a specific variable via a second subscript: for example, $w_{1,md}$ denotes the element at row m column d of the variable $\mathbf{W}_1$. Lastly, we use $\boldsymbol{\omega}$ to denote a set of variables (e.g. the set of all BNN layers' weights by $\boldsymbol{\omega} = \{\mathbf{W}_1, ..., \mathbf{W}_L\}$) and subscript with functions, e.g. $f_{\boldsymbol{\omega}}$, to denote a function parameterized by the variables $\boldsymbol{\omega}$.

Chapter 1

Introduction

The training of a predictive model based on user data is generally difficult in many real-world settings because the training dataset is distributed amongst the users' devices, which are typically unable or unwilling to upload their data to a server due to privacy reasons and network issues. For example, consider a software company looking to improve its recommendation system for its mobile device users. It is likely that users are unable to train a good recommendation system locally with just their own data, but simultaneously are also unwilling to have their data be stored on a central server. Still, the company would like to train a single recommendation model on all clients' data while satisfying the privacy concerns of its users. *Federated learning* offers a solution to this, enabling clients to jointly train a shared model without data sharing.

In real-world applications, we often also care about uncertainty when using a predictive model. In tasks such as medical diagnosis and policymaking, it is paramount to understand how uncertain a model is about its prediction. In order to rely on its predictions, the model should also be calibrated in its uncertainty and not be overconfident. Standard deep learning models, such as regular deep neural networks (DNN), are based on frequentist learning, which is known to lead to overconfident predictions due to a failure to capture epistemic uncertainty [Guo et al., 2017]. In contrast, probabilistic machine learning methods offer a principled framework to achieve this by having a distribution over the model parameters and its predictions. *Bayesian neural networks* (BNN) are probabilistic neural networks that have the predictive performance of DNNs, but also provide calibrated uncertainty metrics alongside its predictions.

In order to train BNNs, we typically use *variational inference* (VI). In probabilistic machine learning, VI is a foundational method that reframes the task of inference from a marginalization problem into an optimization problem [Jordan et al., 1998]. It does so

by approximating the exact posterior distribution with another (variational) distribution via optimization. But in the federated learning setting, standard VI does not work because it assumes complete access to the entire dataset. Ashman et al. [2022] introduced partitioned variational inference (PVI) to extend VI to the federated learning setting, recovering the same solution as standard VI if it converges and obtaining state-of-the-art results on federated learning benchmarks.

To approximate the intractable distribution over the neural network weights using VI, we choose a simpler variational distribution to approximate with. The form of the simpler approximating distribution is, therefore, integral to the model's predictive performance. It must be complex enough to capture the true distribution over the weights (as to obtain the best predictive performance) but simple enough to be optimized efficiently. A frequently used posterior approximation is the mean-field or fully factorized approximation (MFVI). However, its functional form has several shortcomings, as we will discuss shortly, including not capturing the complete predictive power of BNNs.

This brings us to the main goal of this thesis: to achieve a more powerful predictive BNN in the federated learning setting. To do so, we draw on a posterior approximation for BNNs introduced by Ober and Aitchison [2021], called the *Global Inducing Point* (GI) posterior approximation method. They showed that GI outperforms MFVI in the standard, non-federated VI setting. In this thesis, we integrate the GI method into the PVI framework to extend the posterior approximation method to federated BNNs.

Outline

This thesis is structured as follows. We firstly provide a brief background on the fundamentals of BNNs in chapter 2. We then introduce federated learning and discuss the process of training BNNs in the federated setting using PVI in chapter 3. In chapter 4, we comprehensively review the GI method and provide the theoretical components of this work's contribution: the application of GI to federated BNNs using PVI. We also provide a theoretical analysis of the communication and computation complexity of performing GI in the PVI setting and contrast it to MFVI. Finally, in chapter 5, we empirically show that GI-PVI outperforms the MFVI-PVI approach in predictive performance and communication efficiency on two datasets and under different client data distribution assumptions. We conclude with a summary of the work presented in this thesis and indicate future research directions in chapter 6. The code for the experiments presented in this thesis is available at https://github.com/mbronckers/GI-PVI.

Chapter 2

Background: The Foundations of Bayesian Neural Networks

In this chapter, we provide an introduction to Bayesian Neural Networks. Starting from the task of probabilistic machine learning, we work towards performing variational inference in a Bayesian Neural Network to approximate the posterior distribution over the neural network weights. We discuss the optimization objective, the resulting predictive distribution, as well as the associated predictive uncertainty. We conclude with a discussion of alternative (approximate) inference methods, including the commonly-used mean-field posterior approximation method, which we use as baseline method for experiments later in this work.

§2.1 Probabilistic Machine Learning

In (probabilistic) machine learning, we seek to find the parameter values $\boldsymbol{\theta} \in \Theta$ of a parameterized function f_θ that is most likely to have generated the observations $\mathbf{Y} \in \mathbb{R}^{N \times D_{out}}$ given our inputs $\mathbf{X} \in \mathbb{R}^{N \times D_{in}}$. Via our probabilistic model specification, we define a likelihood distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ that describes how a given input generates an output under parameter setting $\boldsymbol{\theta}$. Depending on our modelling task, we assume different functional forms for the likelihood, i.e. a Gaussian likelihood for regression or a softmax function over the model output vector for classification (equations (2.1)

and (2.2), respectively).

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{y};\ \boldsymbol{f}_\theta(\mathbf{x}), \lambda^{-1} I\right) \tag{2.1}$$

$$p(y = k \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{\exp\left(f_{\theta,k}(\mathbf{x})\right)}{\sum_{k'} \exp\left(f_{\theta,k'}(\mathbf{x})\right)} \tag{2.2}$$

where λ is the precision parameter of the Gaussian distribution.

In probabilistic Bayesian machine learning, we seek to find the posterior distribution over the parameter space Θ given our training dataset $\{\mathbf{X}, \mathbf{Y}\}$, having specified a prior belief over the parameter setting via prior distribution $p(\boldsymbol{\theta})$. To find the most likely parameter setting, we maximize the likelihood distribution $p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta})$, which is the probabilistic model, parameterized by $\boldsymbol{\theta}$, via which the inputs are transformed into outputs.

$$p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathbf{Y} \mid \mathbf{X})} = \frac{p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{\int p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}} \propto p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

The denominator in the posterior, i.e. the model evidence or marginal likelihood $p(\boldsymbol{Y}|\boldsymbol{X})$, is a critical aspect of the posterior as it marginalizes the likelihood over the parameters $\boldsymbol{\theta}$: it effectively averages the possible model parameters, weighted by our prior belief of their probability. However, this marginalization is usually analytically intractable and, as such, we use approximations when needed.

We use the posterior distribution to obtain the predictive distribution by integrating over it. This process of integrating over the model parameters is called *inference* in Bayesian machine learning. Note that this term is also used by the deep learning community at large to refer to evaluation of the model at test time.

$$p\left(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}\right) = \int p\left(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta}\right) p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}) \mathrm{d}\boldsymbol{\theta}$$

§2.2 Variational Inference

Because the marginalization over the parameters in inference is often analytically intractable, we resort to approximate inference. In particular, we introduce a variational approximate distribution $q \in \mathcal{Q}$ that is parameterized by variational parameters ϕ that is tractable. We proceed to approximate the true posterior via q_ϕ and use the approximate variational posterior distribution q_ϕ in lieu of the true posterior p. Technically, any method that uses optimization to approximate the posterior p can be called variational

inference, including expectation propagation and belief propagation, but we focus on Kullback-Leibler variational inference (VI) [Challis and Barber, 2013].

The Kullback–Leibler (KL) divergence is a statistical distance that measures the proximity of one distribution to another. We avoid optimizing the $KL(q\|p)$ directly, because computing the evidence $\log p(x)$ is expensive or impossible. Instead, we maximize the evidence lower bound $\mathcal{L}_{ELBO}(\phi)$ w.r.t. the variational hyperparameters defining q instead. The KL divergence is equal to the negative ELBO plus the evidence, which is independent of q, meaning that minimization of the KL divergence equal is to maximization of the ELBO.

$$\begin{aligned} KL(q(z)\|p(z \mid x)) &\triangleq \mathbb{E}_{z\sim q}\left[\log \frac{q(z)}{p(z \mid x)}\right] \\ &= \mathbb{E}_{z\sim q}\left[\log \frac{q(z)p(x)}{p(x,z)}\right] \\ &= \mathbb{E}_{z\sim q}\left[\log p(x) - \log \frac{p(x,z)}{q(z)}\right] \\ &= \log p(x) - \mathbb{E}_{z\sim q}\left[\log \frac{p(x,z)}{q(z)}\right] \\ &= \log p(x) - \mathcal{L}_{ELBO}(z) \end{aligned}$$

$$\implies \min_z \; KL(q(z)\|p(z \mid x)) \equiv \max_z \; \mathcal{L}_{ELBO}(z)$$

Because optimization of the ELBO is computationally more feasible than marginalization, approximations of q to p in variational inference are possible. Despite this, the maximization of the complete ELBO does not scale well to large data sets and can still be intractable for complex models, including Bayesian Neural Networks as we will discuss in §2.3.1. It is therefore important that the approximate variational distribution family $\mathcal{Q}$ is flexible enough to capture the salient aspects of the posterior, yet also be scalable and tractable for complex models.

§2.3 Bayesian Neural Networks

Bayesian neural networks (BNNs) were first introduced in the 1990s [MacKay, 1992; Neal, 1996]. In contrast to the standard point-estimate neural networks, Bayesian neural networks are probabilistic over their weights by specifying distributions over them. The distributions over the model's weights result in attractive model properties including uncertainty estimates, regularization, and robustness to adversarial samples.

This is achieved by placing a prior across the neural network weights: suppose a BNN is parameterized by $\boldsymbol{\omega} = \{\boldsymbol{W}_\ell\}_{\ell=0}^{L}$, where layer ℓ has weight matrix $\mathbf{W}_\ell$. We often place a standard normal prior distribution over the weights, i.e. $p(\mathbf{W}_\ell) = \mathcal{N}(\mathbf{W}_\ell; 0, \mathbf{I})$. The likelihood specification $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})$ is as described in §2.1. Together, this induces a posterior distribution over the weights, i.e. the parameterization of our neural network, given our dataset $\{\mathbf{X}, \mathbf{Y}\}$:

$$p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})\, p(\boldsymbol{\omega})$$

Because the posterior is generally intractable, we resort to VI with an approximating variational posterior distribution, turning the inference problem into an optimization problem. For a BNN, we denote a variational distribution q defined by hyperparameters ϕ over the weights $\boldsymbol{\omega}$ by $q_\phi(\boldsymbol{\omega})$. Below, we discuss the optimization objective and inference aspects of BNNs in more detail.

2.3.1 Optimization objective

Recall that in VI, we seek to maximize the ELBO via the approximate posterior's variational parameters ϕ (see equation (2.3)). Because the evidence term is independent of the approximate posterior, this reduces to minimizing the KL divergence between the approximate posterior q and the full posterior $p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})$:

$$\begin{aligned} &\max_{\phi}\ \mathcal{L}_{ELBO} = \log p(\mathbf{X}, \mathbf{Y}) - KL(q_\phi(\boldsymbol{\omega}) \,||\, p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})) \\ &\max_{\phi}\ \mathcal{L}_{ELBO} \Leftrightarrow \min_{\phi}\ KL(q_\phi(\boldsymbol{\omega}) \,||\, p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})) \end{aligned} \tag{2.3}$$

Minimizing this KL divergence between q and $p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})$ can be reduced to a maximization of the expected log-likelihood (ELL) minus the prior divergence (see equation (2.4)). The log-likelihood term encourages q to fit to the data by placing density on the weights $\boldsymbol{\omega}$ that explain the observed data well. In contrast, the prior divergence term acts as a regularizer over the weights, representing the divergence between q and

the prior over the weights.

$$\begin{aligned}
\mathrm{KL}\left(q_\phi(\boldsymbol{\omega}) \| p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})\right) &= -\overbrace{\int q_\phi(\boldsymbol{\omega}) \log p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}}^{\text{expected log-likelihood}} + \overbrace{\mathrm{KL}\left(q_\phi(\boldsymbol{\omega}) \| p(\boldsymbol{\omega})\right)}^{\text{prior divergence}} + C \\
&= -\sum_{i=1}^{N} \int q_\phi(\boldsymbol{\omega}) \log p\left(\mathbf{y}_i \mid \mathbf{f}_\omega\left(\mathbf{x}_i\right)\right) \mathrm{d}\boldsymbol{\omega} \; + \; \mathrm{KL}\left(q_\phi(\boldsymbol{\omega}) \| p(\boldsymbol{\omega})\right) \; + \; C \\
\implies \mathcal{L}_{ELBO} &= \sum_{i=1}^{N} \int q_\phi(\boldsymbol{\omega}) \log p\left(\mathbf{y}_i \mid \mathbf{f}_\omega\left(\mathbf{x}_i\right)\right) \mathrm{d}\boldsymbol{\omega} \; - \; \mathrm{KL}\left(q_\phi(\boldsymbol{\omega}) \| p(\boldsymbol{\omega})\right)
\end{aligned} \tag{2.4}$$

where $\mathbf{f}_\omega(\mathbf{x}_i)$ refers to the model output when $\mathbf{x}_i$ is propagated through the BNN that is parameterized by $\boldsymbol{\omega}$.

The ELL term in the $\mathcal{L}_{ELBO}$ requires computations over the entire dataset, which can be computationally prohibitive in case of large N. Instead, we use a mini-batch estimate to approximate the ELL term by computing the ELL over a mini-batch of size $|S|$:

$$\mathcal{L}_{MB} = \hat{\mathcal{L}}_{ELBO} = -\frac{N}{|S|} \sum_{i \in S} \int q_\phi(\boldsymbol{\omega}) \log p\left(\mathbf{y}_i \mid \mathbf{f}^\omega\left(\mathbf{x}_i\right)\right) \mathrm{d}\boldsymbol{\omega} + \mathrm{KL}\left(q_\phi(\boldsymbol{\omega}) \| p(\boldsymbol{\omega})\right) \tag{2.5}$$

This stochastic mini-batch estimate is unbiased, meaning that our objective function remains an unbiased as well, i.e. $\mathbb{E}[\mathcal{L}_{MB}] = \mathcal{L}_{ELBO}$ [Hoffman et al., 2013]. We can therefore use a stochastic optimizer with $\mathcal{L}_{MB}$ to find a (local) optimum set of variational parameters ϕ^* that would also be an optimum to $\mathcal{L}_{ELBO}$. We denote this optimal approximate posterior over the weights by $q^*_\phi(\boldsymbol{\omega})$.

To actually optimize the objective and find $q^*_\phi(\boldsymbol{\omega})$, we need to estimate the ELL's derivatives w.r.t. the parameters of the variational posterior. There exist three main MC estimation techniques for VI with each different characteristics and variances, evaluated in detail by Gal [2016]. In our experiments, we use the pathwise derivative estimator, which is also referred to in other literature as the *re-parameterization trick, infinitesimal perturbation analysis,* and *stochastic backpropagation* [Kingma et al., 2015; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014]. In short, the pathwise derivative estimator uses a change of variables and re-parameterizes $q_\phi(\boldsymbol{\omega})$ to a function $q(\epsilon; \phi, \boldsymbol{\omega})$ in ϵ with $p(\epsilon) = \mathcal{N}(\epsilon; 0, I)$ via a deterministic, differentiable bivariate transformation $g(\phi, \boldsymbol{\omega})$. For more detail, we refer the reader to [Kingma et al., 2015].

2.3.2 Predictive distribution

To be able to perform inference on a model, we need to obtain a predictive distribution. From the approximated posterior distribution over the weights $q^*_\phi(\boldsymbol{\omega})$, we seek to obtain the predictive posterior distribution $p(y^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y})$ using the predictive approximate posterior distribution $q^*_\phi(y^* \mid \mathbf{x}^*)$.

$$\begin{aligned} p\left(y^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}\right) &= \int p\left(y^* \mid \mathbf{x}^*, \boldsymbol{\omega}\right) p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y}) \mathrm{d}\boldsymbol{\omega} \\ &\approx \int p\left(y^* \mid \mathbf{x}^*, \boldsymbol{\omega}\right) q^*_\phi(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \\ &= \mathbb{E}_{\omega \sim q^*_\phi}[p(y^* \mid \mathbf{x}^*, \boldsymbol{\omega})] \\ &\approx \frac{1}{S} \sum_{s=1}^{S} p(y^* \mid \mathbf{x}^*, \boldsymbol{\omega}^{(s)}) \quad \text{where } \boldsymbol{\omega}^{(s)} \sim q^*_\phi(\boldsymbol{\omega}) \end{aligned} \tag{2.6}$$

We estimate this predictive posterior distribution via S stochastic forward passes of $\mathbf{x}^*$. This means that we sample a weight parameterization $\boldsymbol{\omega}$ of the BNN S times from the approximate variational posterior and propagate $\mathbf{x}^*$ for each parameterization to get the associated network outputs. This gives us an MC estimate of the predictive posterior distribution, as laid out in equation (2.6).

2.3.3 Model uncertainty

As mentioned earlier, one of the advantages of BNNs is their natural model uncertainty quantification. This is invaluable to understanding how confident a model is in its predictions given a specific input. We review how we obtain uncertainty metrics from our BNN below.

There are three types of uncertainty in Bayesian modelling: epistemic, aleatoric, and predictive uncertainty. Epistemic uncertainty captures the model's uncertainty (both in parameters and structure), which is also referred to as "reducible" uncertainty as we can decrease it with more observed data. Aleatoric uncertainty captures the inherent variability / noise in our observed data and is also referred to as "irreducible" uncertainty. Predictive uncertainty conveys the model's uncertainty in its output associated with a specific input and is induced by epistemic and aleatoric uncertainty. We typically evaluate a model's predictive uncertainty, as it can be difficult to estimate a model's epistemic and aleatoric uncertainty separately [Depeweg et al., 2018; Kendall and Gal, 2017].

The definition of predictive uncertainty depends on the model task. For regression, we define it as the variance of the predictive distribution. However, this definition does not work for classification because the variance of the resulting output logits from a single forward pass in classification do not represent uncertainty. Moreover, the variance of predicted classes would be dependent on class label magnitude. Because we only consider model uncertainty in our regression experiments in §5.2.1, we show how to estimate regression model uncertainty here. For more information on classification uncertainty quantification, we refer the reader to [Gal, 2016].

Regression

For regression, we obtain the model uncertainty from the same stochastic forward passes that we use to construct the predictive posterior. This is because empirical first- and second-moments are unbiased estimators of the predictive distribution's moments Consequently, the empirical predictive mean and variance are unbiased estimators.[1]

$$\begin{aligned} \widetilde{\mathbb{E}}\left[\mathbf{y}^*\right] &:= \frac{1}{S}\sum_{s=1}^{S}\mathbf{f}_{\widehat{\omega}_s}\left(\mathbf{x}^*\right) \underset{S\to\infty}{\longrightarrow} \mathbb{E}_{q_\phi^*(\mathbf{y}^*|\mathbf{x}^*)}\left[\mathbf{y}^*\right] \\ \widetilde{\mathrm{Var}}\left[\mathbf{y}^*\right] &:= \sigma^2\mathbf{I} + \frac{1}{S}\sum_{s=1}^{S}\mathbf{f}_{\widehat{\omega}_s}\left(\mathbf{x}^*\right)^T\mathbf{f}_{\widehat{\omega}_s}\left(\mathbf{x}^*\right) - \widetilde{\mathbb{E}}\left[\mathbf{y}^*\right]^T\widetilde{\mathbb{E}}\left[\mathbf{y}^*\right] \\ &\underset{S\to\infty}{\longrightarrow} \mathrm{Var}_{q_\phi^*(\mathbf{y}^*|\mathbf{x}^*)}\left[\mathbf{y}^*\right] \end{aligned} \tag{2.7}$$

where σ^2 is the variance of the specified model likelihood function $p(y^* \mid \mathbf{x}^*, \boldsymbol{\omega}) = \mathcal{N}\left(\mathbf{y}^*\,;\,\mathbf{f}_\omega\left(\mathbf{x}^*\right), \sigma^2\mathbf{I}\right)$. This means that we empirically estimate the predictive mean and variance via averaging the stochastic forward passes.

§2.4 Mean-Field Variational Approximation

When we perform inference in BNNs as described above, we need to choose a variational approximate family that is both tractable and sufficiently flexible to capture the true posterior distribution over all the weights of the network. One of the standard choices is to make the mean-field assumption on the variational posterior over the weights because it makes computations more efficient and tractable. This is also called Mean-Field Variational Inference (MFVI).

[1] For proof, see Gal [2016, §3.3].

The mean-field variational family assumes that the latent variables are mutually independent and that the posterior factorizes across the latent variables. That is, a mean-field variational posterior over the BNN weights assumes that each weight in the network is distributed independently of all others.

The mean-field approximation does not put a restriction on the parametric form of the individual factors, but we typically assume a Gaussian factor for neural network weights. This means that we aim to find the best q_ϕ^* from the set of fully-factorized Gaussian distributions, such that the variational parameters are just means and variances for each weight. In other words, the mean-field approximation of layer i assumes the weights are sampled from a multivariate Gaussian distribution with a diagonal covariance matrix. As such, the q_{MFVI} factorizes over the layers and the weights as follows:

$$q_{MFVI}(\boldsymbol{\omega}) = \prod_{\ell=0}^{L} q_{MFVI}(\mathbf{W}_\ell) = \prod_{\ell=0}^{L} \mathcal{N}\left(\mathbf{W}_\ell;\ \boldsymbol{M}_\ell, \boldsymbol{\Sigma}_\ell\right) = \prod_{\ell=0}^{L} \prod_{i=0}^{N_\ell} \mathcal{N}\left(\mathbf{w}_{\ell,i};\ \boldsymbol{\mu}_{\ell,i}, \boldsymbol{\sigma}_{\ell,i}\right) \tag{2.8}$$

The advantage of MFVI is that it is significantly computationally less expensive to sample from. In the number of weights per layer N_ℓ, the time complexity is $\mathcal{O}(N_\ell^2)$ and avoids the expensive and numerically-unstable matrix inversion that is necessary for full covariance matrices. The computational space requirement is also reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, where N represents the total number of weights in the BNN.

However, this computational benefit comes at a cost. The factorization assumption of MFVI means that it cannot capture any correlations between layers or between weights in the same layer. Because there can be strong posterior correlations in the parameters, this diagonal approximation for the weights can be a problem [MacKay, 1992]. Though some argue that more complex approximations are unnecessary in deep BNNs [Farquhar et al., 2021], the complexity of MFVI in deep BNNs is still prohibitively large for many real-world settings. Moreover, as the BNN layer width approaches infinity, the empirical evidence suggests that the MFVI posterior converges to the prior predictive distribution, underfitting the data and underestimating predictive uncertainty [Coker et al., 2022]. Finally, it is known that the MFVI tends to overprune its weights, limiting its expressiveness and leading to underfitting [Trippe and Turner, 2018].

§2.5 Related work

In this section, we discuss how the methods of this thesis fit in with other (federated) inference approaches. We also explain the reasoning for choosing to extend the GI method

to federated BNNs through a brief discussion of alternative approaches. We present a more technical presentation of the core methods in subsequent chapters.

In general, Bayesian learning approaches are based either on Markov Chain Monte Carlo (MCMC) sampling or on Variational Inference (VI). While MC methods are slow due to the construction of the sampling chain and are hard to determine convergence of, they admit the true posterior as stationary distribution. In contrast, VI methods are optimization-based approaches that converge more quickly, but have approximate solutions that are limited to the expressiveness permitted by the family of approximations Bishop [2006]. MCMC methods are usually prohibitively expensive for real-world settings. In terms of VI methods, the mean-field method is commonly used, but has many issues, including underfitting and overpruning as discussed in §2.4. For more background on MCMC and MFVI, we refer the reader to [Angelino et al., 2016].

There have been improvements to the mean-field method, but most of them remain computationally challenging. The GI method introduced by Ober and Aitchison [2021] is one of the most promising ones, modeling correlations across the BNN layers while reducing the computational cost by using global inducing points.

Other structured VI posterior approximation methods for BNNs include [Louizos and Welling, 2017], who introduce multiplicative normalizing flows (MNFs), which use pseudo-data along with matrix variate normal distributions. MNFs still factorize across the layers, ignoring layer-wise correlations. Moreover, their method required the number of pseudo-data points to be much smaller than the layer width. This limits the effective dimensionality of the variational parameters and reduces the expressiveness of the MNF method. Alternatively, [Lindinger et al., 2020] use a multivariate normal distribution with structured covariance and inducing points similar to the GI method. However, they use local inducing inputs for every layer and the normal distribution is used for the unconditional posterior across all the layers, rather than the conditional posterior of a single layer as in the GI method. The difference is that the resulting posterior of the GI method is non-Gaussian and thus more flexible. Moreover, Lindinger et al. [2020]'s method scales cubically in complexity w.r.t. the BNN depth, whereas GI scales linearly.

The aforementioned examples are instances of standard, non-distributed Bayesian learning. In contrast, *Federated* Bayesian learning methods seeks to compute a *global / shared* posterior distribution on data that is distributed locally amongst clients. Federated MCMC methods are generally prohibitively slow in convergence and particularly difficult to deploy—making them unpractical for our purposes. A state-of-the-art example of such a method is Distributed Stochastic Gradient Langevin Dynamics (DSGLD) [Ahn et al., 2014; Welling and Teh, 2011], which maintains a number of Markov chains updated via

local Stochastic Gradient Descent while adding Gaussian noise. However, DSGLD is impractically slow for our purposes (see [Ahn et al., 2015]). It is communication inefficient as it operates by passing Markov chains between clients after single-update steps (multiple steps worsen asymptotic accuracy). It also exhibits poor predictive performance under inhomogeneous client data distributions. This is because under inhomogeneous client data partitions, the DSGLD's client gradient estimator is biased and the consequent likelihood contributions will vary vastly across clients [el Mekkaoui et al., 2021]. As we will discuss in chapter 3, the PVI framework that we use in this work naturally regularizes this via its client-local optimization objective.

In terms of distributed VI methods, state-of-the-art results have been obtained with Partitioned Variational Inference [Ashman et al., 2022], which relies on distributed optimization of parametric posteriors. The PVI framework also encapsulates various separate VI methods, including online VI [Ghahramani and Attias, 2000; Sato, 2001] and power EP [Minka, 2004].[2] Alternative distributed VI methods include Distributed Stein Variational Gradient Descent (SVGD) [Kassab and Simeone, 2021], which is a non-parametric framework. Distributed SVGD maintains a number of interacting particles at the server to approximate the current iterate of the global posterior. One of the noteworthy advantages is the ability to vary the communication load by varying the number of particles. However, SVGD is known to work well only in low-dimensional settings as kernel methods struggle in even moderately large dimensionality spaces [Ba et al., 2022].

In short, relative to the aforementioned MCMC and VI methods, GI's form promises great posterior approximation capabilities in a comparatively cheap way. This makes it an interesting posterior approximation to extend to the complex task of federated learning, which we do using the PVI framework in chapter 4.

§2.6 Summary

In this chapter, we reviewed the necessary background for the problem set-up of this work. We briefly described the concept of probabilistic machine learning and variational inference to be able to describe the concept of Bayesian Neural Networks. We discussed the process of inference in BNNs in more detail, including approximating the posterior distribution over the weights via a variational distribution. We showed that the variational optimization objective naturally balances the data fit with a regularizer term via the

[2] See [Ashman et al., 2022, §4] for more details.

expected log-likelihood and KL divergence w.r.t. prior, respectively. We also discussed how we obtain the distribution of interest—the predictive distribution—resulting from the optimized approximate posterior via stochastic forward passes of the input data. The resulting Monte Carlo estimates also naturally provided us with predictive uncertainty estimates, which are one of the benefits of BNNs alluded to in chapter 1. We concluded our background with a discussion of the mean-field posterior approximation, the de facto default variational approximation method—highlighting the trade-off between its computational benefit and its predictive limitations. Finally, we explained how our work fits in with existing literature. Specifically, we discussed alternative (federated) inference approaches and why they are impractical or infeasible for the purposes of our work.

In the following chapters, we will extend BNNs to the federated learning setting, optimizing the approximate posterior in a distributed manner. After that, we will introduce a more powerful posterior approximation method and compare it to the mean-field approximation, showing that it outperforms the standard method in the complex setting of federated learning.

Chapter 3

Federated Learning in Bayesian Neural Networks

In this chapter, we introduce the concept of federated learning and how Bayesian Neural Networks can be used in a federated setting. We focus on using a specific variational inference method to do so, called Partitioned Variational Inference (PVI). PVI is an attractive framework for learning a posterior approximation in a federated setting because it obtains state-of-the-art performance by recovering the standard variational inference solution efficiently. We discuss PVI's algorithmic steps and summarize the properties of different PVI variants.

§3.1 Federated Learning

Federated learning (FL) is a distributed model learning setting in which a single, centralized model is trained by a collection of edge devices ("clients") with coordination via a central server. The key aspect of FL is that it removes direct access to the dataset from the learning procedure, as each client has their own local training dataset, which is never uploaded to the server. This enables clients to collaboratively train and use a shared global model without sharing their data directly with others. The client only computes a local gradient update on the current global model and communicates a parameter update back to the server. This means that the central server never observes the underlying training data directly. This reduces privacy concerns by keeping data on-device and security risks by limiting the possible attack surface to only the client instead of both the client and the server [McMahan et al., 2017]. For a comprehensive overview and taxonomy of existing of federated learning systems, we refer the reader to [Li et al., 2021].

The setting of FL is generally also different from standard distributed learning because of assumptions on the clients and their data characteristics. In contrast to the assumptions in distributed learning of a controlled data center with uniform edge device characteristics, federated learning is characterized by client heterogeneity. Possible issues include:

1. *Inhomogeneous data partitions.* Clients can have differently sized or differently balanced datasets.
2. *Non-IID data.* Clients' data can be reflective of their behavior or interactions and not representative of the population dataset.
3. *Changing datasets.* Clients' datasets change over time and can exhibit covariate shift.
4. *Communication constraints.* Clients can have (varying) latency, bandwidth, and availability constraints.
5. *Compute constraints.* Clients can have (varying) compute bottlenecks that limit optimization and inference ability.

Some of these issues, however, are beyond the scope of this work. In this thesis, we focus on model performance on both homogenous and inhomogeneous data partitions in a controlled server-client environment. We consider a fixed group of clients with reliable communication channels, availability, and sufficient compute ability. We, therefore, do not consider non-IID datasets, changing client datasets over time, or differing client communication constraints. Investigation of model performance in environments with such characteristics is a possible opportunity for future research.

§3.2 Partitioned Variational Inference

Federated learning does not accommodate direct access to the entire dataset because the dataset is sharded amongst the clients. As such, standard VI—also referred to as *global VI*—for Bayesian inference using federated learning does not work because the maximization of the global free-energy requires access to the entire, non-partitioned dataset. Ashman et al. [2022] introduced partitioned variational inference as a general solution to extend VI to the federated learning case, showing that it recovers the global VI solution in a communication-efficient manner and obtaining state-of-the-art results.

Consider a probabilistic model defined by a prior over its parameters, $p(\boldsymbol{\theta})$, and a likelihood function, $p(\mathbf{y}|\boldsymbol{\theta})$. If we partition the dataset $\mathbf{y}$ into K shards $\{\mathbf{y}_0, \ldots, \mathbf{y}_K\}$,

we have a likelihood function that is composed of K data-specific likelihoods: $p(\mathbf{y}|\boldsymbol{\theta}) = \prod_{k=1}^{K} p(\mathbf{y}_k|\boldsymbol{\theta})$. This idea of partitioning the likelihood function into a product of data-shard-specific likelihood terms is the foundation of PVI as an approach to variational inference.

Specifically, PVI decomposes the standard variational posterior approximation into a prior and a product of approximate client likelihood terms $t_k(\boldsymbol{\theta})$, each of which serve to approximate $p(\mathbf{y}_k|\boldsymbol{\theta})$, the true effect of data shard $\mathbf{y}_k$ on the likelihood term.

$$q(\boldsymbol{\theta}) = \frac{1}{\mathcal{Z}_q} p(\boldsymbol{\theta}) \prod_{k=1}^{K} t_k(\boldsymbol{\theta}) \approx \frac{1}{p(\mathbf{y})} p(\boldsymbol{\theta}) \prod_{k=1}^{K} p\left(\mathbf{y}_k \mid \boldsymbol{\theta}\right) = p(\boldsymbol{\theta}|\mathbf{y})$$

To solve the global variational optimization problem, the PVI framework effectively runs a series of client-local optimizations: each client locally optimizes its own approximate likelihood term $t_k(\boldsymbol{\theta})$ on its own data and communicates this back to the server, which in turn recomputes the new global posterior approximation $q(\boldsymbol{\theta})$ from the individual clients' likelihood terms.

We discuss the PVI steps in more detail below. The high-level PVI server and client flow are laid out in algorithms 1 and 2, respectively. An overview of the framework is detailed in algorithm 3.

Algorithm 1: Partitioned Variational Inference - Server Flow

Initialize q to prior p and clients' t to 0.
for $i = 1, 2, \ldots$ until convergence **do**
  Determine set of clients b_i to locally optimize.
  Send out current $q^{(i-1)}(\boldsymbol{\theta})$ to each client in b_i.

  Let each client in b_i locally optimize and send back their updated $t^{(i)}$ to server.

  Update the approximate posterior:

$$q^{(i)}(\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{k \in b_i} t_k^{(i)}(\boldsymbol{\theta}) \prod_{m \notin b_i} t_m^{(i-1)}(\boldsymbol{\theta}).$$

end for

To commence PVI, we initialize the posterior to the prior. At each epoch i, we select a set of clients whose approximate likelihood term we seek to refine via schedule $b_i \in \{1 \ldots K\}$. The clients' approximate likelihood terms are initialized arbitrarily via hyperparameters. Only clients whose likelihood terms have been optimized at least once

are included in the global posterior because we do not include randomly initialized factors.

Algorithm 2: Partitioned Variational Inference - Client (k) Flow

Receive current $q^{(i-1)}(\boldsymbol{\theta})$ from server.
Compute new local approximate posterior q_k via maximization of local VFE:

$$q_k^{(i)^*}(\boldsymbol{\theta}) = \arg\max_{q(\boldsymbol{\theta})\in\mathcal{Q}} \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\mathbf{y}_k|\boldsymbol{\theta})\right] - \mathrm{KL}\left[q(\boldsymbol{\theta})||q_{\setminus k}^{(i-1)}(\boldsymbol{\theta})\right]$$

$$\text{where} \quad q_{\setminus k}^{(i-1)}(\boldsymbol{\theta}) = \frac{q^{(i-1)}(\boldsymbol{\theta})}{t_k^{(i-1)}(\boldsymbol{\theta})} = p(\boldsymbol{\theta}) \prod_{m\neq k} t_m^{(i-1)}(\boldsymbol{\theta}).$$

Send back updated $t_k^{(i)}$ to server.

The client, in turn, receives the global posterior and seeks to optimize it on its own data. The client-local optimization can be done by optimizing $q_k^{(i)}$ w.r.t. the local variational free-energy (VFE) and then computing the new approximate likelihood $t_k^{(i)}$ by division. This is optimization step can be rewritten in a form similar to ELBO maximization, consisting of a expected log-likelihood term and KL term[1]:

$$\arg\max_{q(\boldsymbol{\theta})\in\mathcal{Q}} \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\mathbf{y}_k|\boldsymbol{\theta})\right] - \mathrm{KL}\left[q(\boldsymbol{\theta})||q_{\setminus k}^{(i-1)}(\boldsymbol{\theta})\right]$$

$$\implies t_k^{(i)}(\boldsymbol{\theta}) = \frac{q^{(i)}(\boldsymbol{\theta})}{q^{(i-1)}(\boldsymbol{\theta})} t_k^{(i-1)}(\boldsymbol{\theta})$$

Alternatively but equivalently, the client can optimize the new approximate likelihood $t_k^{(i)}$ directly w.r.t. the local VFE as the only optimizable parameters in q_k are those of the local-client's t_k and the client only sends back the updated likelihood factor.

$$t_k^{(i)}(\boldsymbol{\theta}) = \arg\max_{t_k(\theta)} \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\mathbf{y}_k|\boldsymbol{\theta})\right] - \mathrm{KL}\left[q(\boldsymbol{\theta})||q_{\setminus k}^{(i-1)}(\boldsymbol{\theta})\right]$$

The updates of the client's approximate likelihood factors are communicated back to the server, i.e. the change in factors $t_{k\in b_i}$. The server updates the global approximate posterior by multiplying the previous posterior by the product of the change in factors, or by recomputing the new global posterior from scratch by taking the product of the prior and the likelihood terms.

[1]For proof of the equivalence, see Appendix B of [Ashman et al., 2022].

Different update schedules

The server's approach to select optimizable clients characterizes three variants of PVI:

1. *Sequential* PVI: in each server iteration, only one client is optimized.
2. *Synchronous* PVI: in each server iteration, all clients are optimized.
3. *Asynchronous* PVI: in each server iteration, a subset of clients are optimized (consisting of available clients).

In this work, we evaluate only *sequential* and *synchronous* PVI. Each PVI setting has its advantages and disadvantages, which is discussed in detail by Ashman et al. [2022, §2.4]. To summarize: sequential PVI benefits from automatic global posterior normalization, but generally exhibits poor wall-time efficiency due to its sequential nature (many clients have to wait); synchronous PVI does not have automatically normalized global posteriors and therefore requires likelihoods to lie in the same exponential family as the prior (not a significant drawback in practice).

Furthermore, both PVI settings can demand the need for *damping* of the client approximate likelihoods via a damping factor $\rho \in (0, 1]$:

$$t_k^{(i)}(\boldsymbol{\theta}) \propto \left(\frac{q_k^{(i)}(\boldsymbol{\theta})}{q^{(i-1)}(\boldsymbol{\theta})} \right)^{\rho} t_k^{(i-1)}(\boldsymbol{\theta})$$

The intuitive reason for damping is that the product aggregation of client factors can result in a poor performing global posterior when the clients' factors disagree significantly with each other. Damping effectively decreases the magnitude of changes in a client's likelihood factor after local optimization. In fact, there is no guarantee that the product aggregation in the (a)synchronous PVI case is normalizable and can consequently result in an improper distribution.

Damping decreases the magnitude of changes in the parameters of a client's approximate likelihood, such that the factor remains closer to the previous global iteration. Collectively, this causes smaller updates and leads to worse communication-efficiency in return for stable convergence. It is important to note that the dynamics of damping are not well understood yet. A comprehensive investigation into the effects and requirements of damping are beyond the scope of this work and left for future research.

As we will see in chapter 5, the need for dampening depends on the posterior approximation method and data inhomogeneity. Ashman et al. [2022] found in their experiments that damping of $\rho \propto \frac{1}{N_{clients}}$ was required for sequential MFVI-PVI in inhomogeneous data splits and for synchronous MFVI-PVI in all data splits. We

confirmed this. In contrast, the GI posterior approximation method—which we introduce in the next chapter—did not require dampening in any setting.

Algorithm 3: Partitioned Variational Inference

Input: data partition $\{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$, prior $p(\boldsymbol{\theta})$.

Initialize:

$$t_k^{(0)}(\boldsymbol{\theta}) := 1 \text{ for all } k = 1, 2, \ldots, K.$$
$$q^{(0)}(\boldsymbol{\theta}) := p(\boldsymbol{\theta}).$$

for $i = 1, 2, \ldots$ until convergence **do**

$b_i :=$ set of indices of the next approximate likelihoods to refine.

Communicate previous iteration's posterior $q^{(i-1)}(\boldsymbol{\theta})$ to each client in b_i.

Compute new approximate likelihoods:

for $k \in b_i$ **do**

Compute the new local approximate posterior:

$$q_k^{(i)}(\boldsymbol{\theta}) := \underset{q(\boldsymbol{\theta}) \in \mathcal{Q}}{\arg\max} \underbrace{\int q(\boldsymbol{\theta}) \log \frac{q^{(i-1)}(\boldsymbol{\theta}) p(\mathbf{y}_k|\boldsymbol{\theta})}{q(\boldsymbol{\theta}) t_k^{(i-1)}(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta}}_{\text{local VFE}}.$$

Update the approximate likelihood:

$$t_k^{(i)}(\boldsymbol{\theta}) \propto \frac{q_k^{(i)}(\boldsymbol{\theta})}{q^{(i-1)}(\boldsymbol{\theta})} t_k^{(i-1)}(\boldsymbol{\theta})$$

Communicate $\Delta_k^{(i)}(\boldsymbol{\theta}) \propto \frac{t_k^{(i)}(\boldsymbol{\theta})}{t_k^{(i-1)}(\boldsymbol{\theta})}$ to server.

end for

Update approximate posterior:

$$q^{(i)}(\boldsymbol{\theta}) \propto q^{(i-1)}(\boldsymbol{\theta}) \prod_{k \in b_i} \Delta_k^{(i)}(\boldsymbol{\theta}).$$

end for

---

§3.3 Summary

In this chapter, we discussed the setting of federated learning and how it differs from standard learning as well as distributed learning. Because the data is partitioned across clients in federated learning, we cannot apply standard VI for Bayesian inference. Instead, we use the framework of Partitioned Variational Inference (PVI). PVI effectively decomposes the likelihood function into a product of data-partition specific likelihood factors (one for each client). We discussed the pseudocode of PVI at both the server and client level. We concluded with a summary of the different PVI settings, defined by different client-selection schemes. We will look at how to use PVI in more detail in the next chapter as we introduce a more powerful posterior approximation method and extend it to the PVI framework.

Chapter 4

Global Inducing Point Posterior Approximations for Federated BNNs

In this chapter, we first discuss the Global Inducing Point (GI) posterior approximation method for BNNs introduced by Ober and Aitchison [2021]. After, we provide the theoretical contribution of this thesis: the incorporation of GI into the PVI framework. We explain how the global approximate posterior decomposes and detail the functional form of the pseudo-likelihood factors. Given the layer-conditional form of the GI method, we discuss the pseudocode of the client's pseudo-likelihood factor optimization in more detail. Finally, we provide a theoretical analysis of the communication and computation cost of the GI-PVI method, showing its computational complexity.

§4.1 Global Inducing Points Approximate Posterior

4.1.1 Posterior form

The usage of a factorized approximate posterior, such as a mean-field approximation, can be problematic in neural networks because of the lost correlations between layers. Ober and Aitchison [2021] propose an approximate posterior q for a Bayesian neural network that is correlated over the weights at all layers. They achieve this by constructing the approximate posterior as a product of layer-specific weight posteriors that are conditional

on the weights of all previous layers:

$$\begin{aligned} q(\boldsymbol{\omega}) = q\left(\{\mathbf{W}_\ell\}_{\ell=1}^{L+1}\right) &= q(\mathbf{W}_{L+1}|\mathbf{W}_L, \dots \mathbf{W}_1) \dots q(\mathbf{W}_2|\mathbf{W}_1) q(\mathbf{W}_1) \\ &= \prod_{\ell=1}^{L+1} q\left(\mathbf{W}_\ell \mid \{\mathbf{W}_{\ell'}\}_{\ell'=1}^{\ell-1}\right). \end{aligned} \tag{4.1}$$

The conditional layer-specific weight posteriors are recursively defined by considering the optimal final-layer weight posterior. Consider the final layer $L+1$, where we would want the optimal approximate posterior to be equal to the true final-layer weight posterior, given the previous layers' weights:

$$q\left(\mathbf{W}_{L+1} \mid \{\mathbf{W}_\ell\}_{\ell=1}^{L}\right) = p\left(\mathbf{W}_{L+1} \mid \mathbf{Y}, \mathbf{X}, \{\mathbf{W}_\ell\}_{\ell=1}^{L}\right)$$

Using Bayes' theorem and the conditional independencies of a BNN model, we can decompose the conditional layer-specific weight posteriors into a prior and likelihood. Specifically, we can decompose them into a prior over that layer's weights $\mathbf{W}_{L+1}$ and a likelihood of that layer's outputs given the weights $\mathbf{W}_{L+1}$ and the previous layer's outputs $\mathbf{F}_L$.

$$\begin{aligned} q\left(\mathbf{W}_{L+1} \mid \{\mathbf{W}_\ell\}_{\ell=1}^{L}\right) &= p\left(\mathbf{W}_{L+1} \mid \mathbf{Y}, \mathbf{X}, \{\mathbf{W}_\ell\}_{\ell=1}^{L}\right) \\ &\propto p\left(\mathbf{Y} \mid \mathbf{W}_{L+1}, \mathbf{F}_L\right)\ p\left(\mathbf{W}_{L+1}\right) \end{aligned} \tag{4.2}$$

Because the previous layer's outputs $\mathbf{F}_L$ are incorporated into the next layer, this decomposition naturally incorporates correlations between layers into the approximate posterior.

However, we can only decompose the final layer this way because we have associated outputs $\mathbf{Y}$ with the inputs $\mathbf{X}$. Since intermediate network layers do not have observed outputs, Ober and Aitchison [2021] introduce a likelihood function $\mathcal{N}\left(\mathbf{v}_\ell;\ \mathbf{F}_\ell, \boldsymbol{\Lambda}_\ell^{-1}\right)$ for each layer ℓ using variational parameters, where $\mathbf{v}_\ell$ represents noisy "pseudo-observed outputs" of the layer ℓ's true outputs, $\mathbf{F}_\ell$, distributed according to a normal distribution with precision parameter $\boldsymbol{\Lambda}_\ell$. These variational parameters $\phi = \{\mathbf{v}_\ell, \boldsymbol{\Lambda}_\ell\}_{\ell=1}^{L+1}$ can be viewed as pseudo-observed outputs and pseudo-noise parameters for the intermediate layers, respectively.

4.1.2 Inducing points

Ober and Aitchison [2021]'s approximate posterior method relies on so-called "global inducing points", a concept that is common in the literature of Gaussian Processes

[Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005; Wilson and Nickisch, 2015]. As such, we refer to the posterior approximation as the global inducing point (GI) method.

Inducing points are used to mitigate the computational cost of optimization, a problem in using BNNs with large datasets. Consider a BNN with weights $\{\mathbf{W}^\ell\}_{\ell=1}^{L+1}$ with $\mathbf{W}^\ell \in \mathbb{R}^{D_{\ell-1} \times D_\ell}$ and activation function $\phi(\cdot)$. Rather than propagating the entire dataset $\mathbf{X} \in \mathbb{R}^{N \times D_0}$, we can instead propagate only $M << N$ global inducing inputs $\mathbf{U} \in \mathbb{R}^{M \times D_0}$, which are defined only at the input layer and propagated through the network to obtain inducing inputs at subsequent layers:

$$\mathbf{U}_1 = \mathbf{U}\mathbf{W}_1, \quad \mathbf{U}_\ell = \phi\left(\mathbf{U}_{\ell-1}\right)\mathbf{W}_\ell \quad \forall\, \ell = 1, \ldots, L$$

These global inducing inputs $\mathbf{U}$ are part of the variational parameters that we optimize with respect to the VI objective. Instead of having to compute the gradients w.r.t. the entire training batch of size N, they only have to be computed for the M global inducing points. Note that the usage of *global* inducing inputs differs from other methods such as Salimbeni and Deisenroth [2017], which use a different set of inducing points for each layer.

When we assume a Gaussian prior over the weights $\mathrm{p}(\mathbf{w}_\lambda^\ell) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{S}_\ell)$, the conditional approximate posterior is Gaussian and analytic (equation (4.3)). However, the resulting unconditional approximate posterior is not Gaussian and, thus, able to approximate more complex true posteriors than factorized approximate posteriors. The Gaussian form of the conditional approximate posterior is useful because it enables optimization of the variational parameters via standard reparameterized VI using an optimizer such as Adam [Kingma and Ba, 2017]. As we will discuss below, this functional form makes GI also attractive for PVI.

$$\begin{aligned} q\left(\mathbf{W}_\ell \mid \{\mathbf{W}_{\ell'}\}_{\ell'=1}^{\ell-1}\right) &\propto \prod_{\lambda=1}^{N_\ell} \mathrm{p}\left(\mathbf{w}_\lambda^\ell\right) \mathcal{N}\left(\mathbf{v}_\lambda^\ell; \phi\left(\mathbf{U}_{\ell-1}\right)\mathbf{w}_\lambda^\ell, \boldsymbol{\Lambda}_\ell^{-1}\right) \\ &= \prod_{\lambda=1}^{N_\ell} \mathcal{N}\left(\mathbf{w}_\lambda^\ell \mid \boldsymbol{\Sigma}_\ell^{\mathbf{w}}\, \phi\left(\mathbf{U}_{\ell-1}\right)^T \boldsymbol{\Lambda}_\ell \mathbf{v}_\lambda^\ell, \boldsymbol{\Sigma}_\ell^{\mathbf{w}}\right) \\ &\text{where } \boldsymbol{\Sigma}_\ell^{\mathbf{w}} = \left(N_{\ell-1}\mathbf{S}_\ell^{-1} + \phi\left(\mathbf{U}_{\ell-1}\right)^T \boldsymbol{\Lambda}_\ell \phi\left(\mathbf{U}_{\ell-1}\right)\right)^{-1} \end{aligned} \tag{4.3}$$

The ELBO estimation for the GI method is computed iteratively as we propagate the inducing inputs throughout the network. Because the layer-conditional posterior is Gaussian, we sample weights from the posterior given the inducing inputs and use these to propagate the inducing points forward. We also compute the KL of the layer-conditional

posterior over the weights with respect to the prior. After having sampled weights at every layer, we use the cached weight samples to propagate through the training batch and compute the expected log likelihood of the data (conditioned on the sampled weights). From the computed KL and expected log likelihood, we can compute our ELBO estimate.

§4.2 Unification of PVI and GI

Unfortunately, we cannot directly apply Ober and Aitchison [2021]'s GI method as posterior approximation in federated BNNs because of the global VI assumption: access to the entire dataset and a single set of variational parameters $\phi = (\mathbf{U}_0, \{\mathbf{V}_\ell, \mathbf{\Lambda}_\ell\}_{\ell=0}^L)$.

To use the GI method as posterior approximation for federated BNNs, we adapt its form to fit the PVI framework. Intuitively, the GI method can be viewed as a product of layer-specific priors and pseudo-likelihood factors (equation (4.2)). To conform to PVI's posterior form requirements, we decompose GI's layer-specific pseudo-likelihood factor into a product of K client-local pseudo-likelihood terms to obtain a layer-conditional posterior (equation (4.4)). This modified approximate layer-conditional posterior can then be optimized via PVI as detailed in algorithm 3.

$$
\begin{aligned}
q\left(\mathbf{W}^\ell \mid \left\{\mathbf{W}^{\ell'}\right\}_{\ell'=1}^{\ell-1}\right) &\propto \prod_{d=1}^{D^\ell} p\left(\mathbf{w}_d^\ell\right) \prod_{k=1}^{K} t_k\left(\mathbf{w}_d^\ell \mid \left\{\mathbf{W}^{\ell'}\right\}_{\ell'=1}^{\ell-1}\right), \\
\text{where} \quad & t_k(\mathbf{w}_d^\ell \mid \{\mathbf{W}^{\ell'}\}_{\ell'=1}^{\ell-1}) = \mathcal{N}\left(\mathbf{v}_{k,d}^\ell \,;\, \mathbf{M}_k^\ell,\, \mathbf{\Sigma}_k^\ell\right), \\
& \mathbf{M}_k^\ell = \mathbf{\Sigma}_k^\ell \, \phi\left(\mathbf{U}_k^{\ell-1}\right)^T \mathbf{\Lambda}_k^\ell \, \mathbf{V}_\ell \\
& \mathbf{\Sigma}_k^\ell = \left(\phi\left(\mathbf{U}_k^{\ell-1}\right)^T \, \mathbf{\Lambda}_k^\ell \, \phi\left(\mathbf{U}_k^{\ell-1}\right)\right)^{-1}
\end{aligned}
\tag{4.4}
$$

where $t_k(\mathbf{w}_d^\ell \mid \{\mathbf{W}^{\ell'}\}_{\ell'=1}^{\ell-1})$ is client k's pseudo-likelihood contribution for the l^{th} layer at that client's inducing points $\mathbf{U}_k$. This decomposition means that each client maintains a set of variational parameters for their own inducing points and pseudo-likelihood factors, i.e. the variational parameters of client k are $\phi_k = (\mathbf{U}_k, \{\mathbf{V}_k^\ell, \mathbf{\Lambda}_k^\ell\}_{\ell=0}^L)$.

The Gaussian form of GI's conditional posterior makes it attractive for PVI because the resulting global layer-conditional $q\left(\mathbf{W}^\ell \mid \left\{\mathbf{W}^{\ell'}\right\}_{\ell'=1}^{\ell-1}\right)$ from multiplying all the clients' factors remains Gaussian if a Gaussian prior is used (a property of the exponential family of distributions).

Client optimization

When a client is selected for optimization by the server, it follows the pseudocode as described in algorithm 4. As prescribed by PVI, it receives the current posterior $q^{(i-1)}$ from the server and optimizes the new local posterior $q^{(i)}$ with respect to the local variational-free energy until convergence. As discussed in §3.2, the local variational free energy optimization is equivalent to variational KL optimization, which is of similar form to the ELBO:

$$q_k^{(i)^*}(\boldsymbol{\theta}) = \underset{q(\boldsymbol{\theta})\in\mathcal{Q}}{\arg\max}\, \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\mathbf{y}_k|\boldsymbol{\theta})\right] - \mathrm{KL}\left[q(\boldsymbol{\theta})||q_{\setminus k}^{(i-1)}(\boldsymbol{\theta})\right] \tag{4.5}$$

where $q_{\setminus k}^{(i-1)}(\boldsymbol{\theta}) = \frac{q^{(i-1)}(\boldsymbol{\theta})}{t_k^{(i-1)}(\boldsymbol{\theta})} = p(\boldsymbol{\theta})\prod_{m\neq k} t_m^{(i-1)}(\boldsymbol{\theta})$ is also referred to the *cavity distribution* and is equal to the current posterior *without* client k's contribution. Intuitively, the expected log-likelihood term in equation (4.5) ensures that the local posterior fits to the client's local data, while the KL term regularizes how far a client's local factor moves from the other clients. This aims to prevent disagreement amongst the client factors, which could lead to a poor performing global posterior.

Because of GI's layer-conditional posterior structure, the local posterior is computed iteratively over the layers. For each layer, the client k computes the local layer ℓ-specific posterior $q_{k,\ell}$ by multiplying the prior by all clients' factors, which results in a Gaussian distribution as described in equation (4.4). After sampling S weights from this distribution, the client computes the layer-specific KL divergence with respect to the cavity distribution $q_{\setminus k}^{(i-1)}(\boldsymbol{\theta})$, and propagates forward both its training inputs and inducing points. After the local posterior has converged, the client computes the new local pseudo-likelihood factor $t_k^{(i)}$ and communicates this back to the server.

§4.3 Computational Analysis

The amending of GI to PVI as laid out above perhaps disguises the computational cost of the method. This section seeks to shine additional light on this and dives deeper into the computation and communication that is required at both the server and client level. We discuss the computational and communication complexity of the significant GI-PVI steps in the order of the algorithm (see algorithms 1 to 3).

Algorithm 4: GI-PVI - Client (k) Flow

Receive current $q^{(i-1)}(\boldsymbol{\theta})$ from server.
for $i = 1, 2, \ldots$ until convergence **do**
  for $\ell \in \{1, \ldots, L\}$ **do**
    Compute mean and precision of local layer-conditional posterior $q_{k,\ell} = p_\ell(\boldsymbol{\theta})\ t_{k,\ell}^{(i)}(\boldsymbol{\theta})\ \prod_{m \neq k} t_{m,\ell}^{(i-1)}(\boldsymbol{\theta})$.
    Sample weights from $q_{k,\ell}$
    Compute layer-specific KL w.r.t. cavity prior
    Propagate the training batch and inducing points using sampled weights
  end for
  Compute expected log likelihood $\mathbb{E}_{q_k(\boldsymbol{\theta})}\left[\log p(\mathbf{y}_k|\boldsymbol{\theta})\right]$
  Optimize $(\mathbf{U}, \{\mathbf{V}^\ell, \boldsymbol{\Lambda}^\ell\}_{\ell=0}^L)$ w.r.t local VFE
end for
Send back updated t_k to server.

Communication load

In terms of communication, recall that the server needs to send out the current posterior to the clients being optimized in the current global iteration. Because of the layer-conditional GI posterior form, the server needs to send out the components that make up the posterior for each layer of the network. Therefore, to each client being optimized, the server has to send out the prior for each layer and every client's set of variational parameters, i.e. the global inducing points as well as the pseudo-observations and pseudo-noise for every layer: $\{\phi_i = (\mathbf{U}_i, \{\mathbf{V}_i^\ell, \boldsymbol{\Lambda}_i^\ell\}_{\ell=0}^L) \mid i \in 1 \ldots N_{clients}\}$. This means that the downstream communication load from server to a *single* client scales not only with the number of network layers, but also with the total number of clients. Fortunately, the upstream communication load per server-client channel does not increase with the number of clients. A client has to only communicate back its *own* set of variational parameters ϕ_k to the server after concluding client-local optimization.

In contrast, MFVI-PVI only needs to send out the sufficient statistics for each layer-specific posterior because the layer-posteriors do not depend on the realized weight samples as in GI-PVI. This means that the downstream communication load only scales with the number of layers and not with the number of total clients as in GI-PVI. The upstream communication load in MFVI-PVI is the client's set of variational parameters $\phi_k = \{\mathbf{M}_k^\ell, \boldsymbol{\Sigma}_k^\ell\}_{\ell=0}^L$.

Computational load

The GI method has a computationally expensive posterior approximation because the posterior is conditional on the previous layer's weights. This means that the posterior over the weights for a specific layer can only be constructed *after* samples from the previous layer have been drawn. Both the server and the client need to construct the posterior for inference and optimization. The difference is that the server only needs to construct the posterior when it wants to perform inference (e.g. once every global iteration) whereas the client needs to do so *every* client-local optimization iteration to find the optimal client-local posterior. We discuss the computation load of doing so in more detail below.

As discussed in §4.2, the client needs to compute the optimal local posterior $q_k^{(i)}$ by optimizing the variational parameters of t_k w.r.t. the local variational free energy. The client can directly optimize the posterior via the local factor $t_k^{(i)}$ (see section 3.2). The costs of this optimization are bound by the number of iterations the clients has to perform inference:

1. The number of iterations needed to converge on q_k^*
2. Cost of propagation
3. Cost of computing the local layer-conditional posterior
4. Cost of sampling weights from the local layer-conditional posterior

(1). Because the number of iterations depends on the optimization process, which varies with the dataset and hyperparameter settings, we do not theoretically analyze and include the bound in our complexity cost analysis, but we inspect the empirical optimization convergence of GI versus MFVI in the experiments in chapter 5.

(2). The complexity of propagation is determined by the number of layers in the BNN and their dimensionality. If a layer ℓ has input and output dimensionality D_{in}, D_{out}, respectively and the number of training points is N, then propagation of a layer ℓ takes $\mathcal{O}(N \times D_{in,\ell} \times D_{out,\ell})$ from the matrix multiplication. This propagation complexity is identical for GI and MFVI.

However, GI requires more propagation of points than MFVI due to the global inducing points. In both GI and MFVI, we need to propagate the batch of N_{batch} training points to estimate the expected log likelihood for the ELBO estimate. Additionally, in the standard GI method, the M inducing points need to be propagated because every layer is conditioned on the previous layer's weights and the inducing outputs. But in GI-PVI, we need to propagate *all* $N_{clients}$ clients' inducing points to compute the layer-specific

posterior for a single iteration of client-local optimization. So, for every layer, GI-PVI needs to propagate through all clients' inducing points in addition to the training inputs. In contrast, MFVI needs to only propagate through the training inputs. This means that, for each layer in the BNN, GI-PVI has to compute an additional $N_{clients}$ matrix multiplications of cubic complexity. Relative to MFVI, This results in an additional propagation cost per BNN layer for GI-PVI that is equal to:

$$\mathcal{O}(\overbrace{N_{clients} \times M}^{\text{propagate clients inducing points}} \times D_{in} \times D_{out})$$

(3). Computing the local layer-conditional posterior q^ℓ involves multiplying the prior by the product of all clients' pseudo-likelihood factors conditioned on the previous layers' weights. In our implementation, we represent the normal distribution $\mathcal{N}(\mu, \Sigma)$ using the natural parameterization $(\eta_1 = \Sigma^{-1}\mu, \eta_2 = \Sigma^{-1})$. Because the resulting layer-conditional client factors are normal distributions (see equation (4.4)), this enables an efficient computation of the layer-conditional posterior because the product of two naturally parameterized distributions reduces to just a single matrix addition operation for each of the sufficient statistics. The complexity of the matrix addition operation is $\mathcal{O}(D_{out} \times M)$. Multiplying all client factors, therefore, is $\mathcal{O}(N_{clients} \times D_{out} \times M)$. This means that the cost of computing for a single layer-conditional posterior is

$$\mathcal{O}(q^\ell) = \mathcal{O}(\overbrace{N_{clients} \times T}^{\text{compute client factors}} + \overbrace{N_{clients} \times (D_{out} \times M)}^{\text{multiply client factors}})$$

where $N_{clients}$ is the number of clients and T is the cost of computing a *single* client's likelihood factor t conditioned on the previous layers' weights. We discuss the cost of T next.

To compute a single client's likelihood factor for a layer conditioned on the previous layers' sampled weights, we need to compute the sufficient statistics of the resulting normal distribution as originally laid out in equation (4.4). The factor's distribution and its statistics are restated here in equation (4.6).

$$\begin{aligned} t_k(\mathbf{w}_d^\ell \mid \{\mathbf{W}^{\ell'}\}_{\ell'=1}^{\ell-1}) &= \mathcal{N}\left(\mathbf{v}_{k,d}^\ell \,;\, \mathbf{M}_k^\ell,\, \mathbf{\Sigma}_k^\ell\right), \\ \text{where} \quad \mathbf{M}_k^\ell &= \mathbf{\Sigma}_k^\ell \overbrace{\phi\left(\mathbf{U}_k^{\ell-1}\right)^T \mathbf{\Lambda}_k^\ell \, \mathbf{V}_\ell}^{\text{“XLY”}} \quad \longrightarrow \quad XLY \in \mathbb{R}^{D_{out} \times M} \\ (\mathbf{\Sigma}_k^\ell)^{-1} &= \overbrace{\left(\phi\left(\mathbf{U}_k^{\ell-1}\right)^T \mathbf{\Lambda}_k^\ell \; \phi\left(\mathbf{U}_k^{\ell-1}\right)\right)}^{\text{“XLX”}} \quad \longrightarrow \quad XLX \in \mathbb{R}^{D_{out} \times M \times M} \end{aligned} \tag{4.6}$$

To lighten notation, we use t^ℓ to denote a client's ℓth layer pseudo-likelihood factor conditioned on the previous layers' weights. The mean of t^ℓ is the covariance matrix $\mathbf{\Sigma}$ multiplied by a matrix-product we refer to as XLY. In turn, the covariance matrix $\mathbf{\Sigma}$ is the inverse of a matrix-product we refer to as XLX. Here, X refers to the previous layer's inducing outputs $\phi(\boldsymbol{U}^{\ell-1}) \in \mathbb{R}^{M \times D_{in}}$; L refers to the precision of the pseudo-observations $\mathbf{\Lambda}^\ell \in \mathbb{R}^{D_{out} \times M \times M}$; and Y to the pseudo-observations $\boldsymbol{V}^\ell \in \mathbb{R}^{D_{out} \times M}$. Assuming a naïve cubic-complexity of the matrix-product operation, the complexity of computing XLX and XLY for a layer is $\mathcal{O}(M^2 \times D_{in})$ and $\mathcal{O}(M \times D_{in} + M^2)$, respectively.

Recalling that the natural parameterization is related to the default parameterization via $(\eta_1 = \Sigma^{-1}\mu, \eta_2 = \Sigma^{-1})$, the natural parameterization of a t^ℓ is given by $\mathcal{N}(\eta_1 = XLY, \eta_2 = XLX)$. Therefore, the cost of computing a *single* client's layer-specific pseudo-likelihood factor conditioned on the previous weights is equal to the cost of computing XLX and XLY:

$$\mathcal{O}(T) = \mathcal{O}(\overbrace{M^2 \times D_{in}}^{\text{compute } XLX} + \overbrace{M \times D_{in} + M^2}^{\text{compute } XLY}) \quad = \mathcal{O}(M^2 D_{in})$$

(4). To sample weights from the layer-conditional posterior, we sample from the resulting multivariate normal distribution. Because of the re-parameterization trick (§2.3.1), we use the common method of sampling ϵ from a standard normal and multiplying this by the Cholesky of our target distribution's covariance matrix, to which we add the target distribution mean:

$$\hat{\mathbf{w}} = \mathbf{M} + \overbrace{\mathbf{C}\epsilon}^{dW}, \quad \text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1}),\ \epsilon \in \mathbb{R}^{[1 \times D_{out}]} \text{ and } \mathbf{\Sigma} = \boldsymbol{C}^T\boldsymbol{C}$$

We have to compute both terms regardless of posterior method. Because of the naturally-parameterized normal distribution in our implementation, we use the backsubstitution algorithm rather than naïve matrix product and matrix inversion operations for both $\mathbf{M}$ and dW:

$$\text{solve } \mathbf{U}\mathbf{M} = XLY \text{ for } \mathbf{M}$$

$$\text{solve } \mathbf{U}\ dW = \epsilon \text{ for } dW \qquad \text{where } \mathbf{U} = \texttt{chol(precision)} \iff \mathbf{U}^T\mathbf{U} = XLX = \mathbf{\Sigma}^{-1}$$

The backsubstitution algorithm is more efficient and more numerically stable than directly inverting the precision matrix. The computational complexity of backsubstitution is $\mathcal{O}(N^2)$. However, we cannot avoid the cubic complexity (as with direct matrix inversion) because we have to provide a triangular matrix for the backsubstitution algorithm. To

do so, we compute the Cholesky decomposition of the precision, which is $\mathcal{O}(N^3)$. This means that sampling from the layer-conditional GI posterior takes $\mathcal{O}(D_{out}^3)$ and is cubic in the layer output.

Though sampling from a MFVI posterior also entails computing both terms, the posterior form allows for faster operations. Because the MFVI posterior is factorized, we can invert just the precision diagonal directly, which is $\mathcal{O}(N)$ rather than having to resort to computing Cholesky decompositions and backsubstitution operations. The subsequent matrix multiplications and additions for $\mathbf{M}$, dW and $\hat{\mathbf{w}}$ are $\mathcal{O}(N^2)$. This means that sampling a layer posterior in MFVI takes $\mathcal{O}(D_{out}^2)$ and is quadratic in layer output.

To summarize, the computational cost of inference in the GI method is high relative to the MFVI. This due to the layer-conditional nature of GI's posterior, which forces the server or client to construct a posterior iteratively. This complexity is multiplied by the fact that it has to be repeated $N_{clients}$ times to construct each client's likelihood factor after propagating through *each* client's inducing points. In contrast, a client using MFVI directly receives the local-posterior distribution. In total, the computational complexity of constructing GI's posterior for a single layer ℓ is provided in equation (4.7).

$$\begin{aligned}\mathcal{O}_{GI}(q_\ell) &= \mathcal{O}(N_{clients} \times (\overbrace{M^2 \times D_{in}}^{\text{compute client factors}} + \overbrace{D_{out} \times M}^{\text{multiply client factors}} + \overbrace{M \times D_{in} \times D_{out}}^{\text{propagate inducing points}})) \\ &= \mathcal{O}(N_{clients} \times (M^2 D_{in} + M D_{in} D_{out}))\end{aligned} \tag{4.7}$$

An optimized implementation can parallelize the computation of all the client factors as well as the propagation of the inducing points, reducing the empirical time-complexity significantly depending on the computational constraints. Finally, the sampling from a GI posterior is cubic in the layer width whereas MFVI is quadratic because GI's precision matrix is not a diagonal matrix as in MFVI:

$$\begin{aligned}\mathcal{O}_{GI}(\hat{\mathbf{W}}_\ell) &= D_{out}^3 \\ \mathcal{O}_{MFVI}(\hat{\mathbf{W}}_\ell) &= D_{out}^2\end{aligned}$$

As we will see in chapter 5, these two aspects of the GI method will lead to considerably worse wall-time efficiency in non-parallelized implementations relative to MFVI.

§4.4 Summary

In this chapter, we introduced Ober and Aitchison [2021]'s GI posterior approximation method and extended it to the federated learning setting using Ashman et al. [2022]'s PVI framework. A key feature of GI is the ability to model cross-layer dependencies via global inducing points, which propagate through the network. The cross-layer dependencies enable GI to capture a larger family of posterior distributions, while the use of inducing points reduces the optimization cost. We amended GI to the PVI framework by decomposing GI's pseudo-likelihood into a product of client-local pseudo-likelihood factors, explaining the resulting client-local optimization procedure.

Subsequently, we analyzed the communication and computational complexity of GI-PVI and compared it to that of MFVI in a federated learning setting. We showed that the cost of inference for GI is high relative to MFVI in a multi-client setting because of GI's layer-conditional posterior form; in a federated setting, each client has to recompute all clients' pseudo-likelihood factor for every layer. This means that GI's client computation load scales with the total number of clients, whereas for it is constant for MFVI. Moreover, as with any structured covariance matrix, the cost of sampling for GI is cubic in the layer width, whereas MFVI is quadratic.

In the following chapter, we will evaluate the method's performance and will investigate whether the computational cost is alleviated by the fact that clients converge more quickly in GI than in MFVI or compensated by other predictive performance benefits.

Chapter 5

Experimental Evaluation

In this chapter, we evaluate the GI method against the MFVI method on multiple datasets with various PVI settings. We evaluate both posterior approximation methods in a single-client setting (*global VI*) and multi-client *sequential* and *synchronous* PVI setting. We report the marginal log-likelihood and accuracy on the test set to quantitatively evaluate the model fit and out-of-sample predictive performance. We firstly consider the methods on a one-dimensional toy regression dataset, followed by an evaluation of performance on two classification datasets.

Experiment details

We discuss the details and configurations of the experiments below. We use a BNN with two hidden layers followed by a ReLU activation function for all experiments. For the regression task, we use 20 units per hidden layer. For the classification task, we use 50 units per hidden layer.

Prior: It is standard practice to choose a standard normal prior across the weights (*StandardPrior*), independent of the number of layers or weights. However, in wide or deep networks, this leads to large variance in function space. Following Foong et al. [2020]; Ober and Aitchison [2021], we consider a layer-specific prior that scales the variance by the number of input weights to that layer: $\mathcal{N}(\mathbf{0}, \frac{1}{D_{in}}\mathbf{I})$. This prior is also referred to as the *NealPrior* by Ober and Aitchison [2021].

Posterior initialization: For MFVI, the layers' means are initialized randomly from $\mathcal{N}(0, 1)$. For GI, the inducing points for the first layer and pseudo-observations of the output layer are initialized to the first batch of training data. If the number of inducing points $M <$ batch size, a random subset of training data is selected to initialize

the inducing points to. If $M >$ batch size, the additional inducing points are randomly initialized. Intermediate layers' pseudo-observations are also randomly initialized from $\mathcal{N}(0, 1)$. For GI and MFVI, we initialize the precisions of the pseudo-observations and weights, respectively, to the same value: $10^3 - D_{in}$. When using the NealPrior, this means that the posterior's precisions in both methods are tightly initialized to 10^3.

Optimization: During training, we use $S = 2$ weight samples from the posterior for local VFE optimization. For inference, we draw $I = 50$ weight samples. For the KL term, we use the analytical KL rather than Monte Carlo estimate based on the sampled weights. For client-local optimization, we use Adam as optimizer [Kingma and Ba, 2017]. We use a learning rate of 10^{-2} and 10^{-3} and a batch size of 40 and 256 for the regression and classification tasks, respectively.

PVI related settings: For all multi-client PVI settings, we use 10 clients ($N_{clients}$). In synchronous PVI, we apply *no* damping to GI and damping of $\rho = 0.2$ to the MFVI model. This ρ setting was also used by Ashman et al. [2022] on the UCI datasets. The damping is implemented by multiplying the change in likelihood factor's parameters after the client-local optimization with ρ before the client communicates its factor to the server.

Number of communications: Because a single global iteration implies a different number of optimized clients for each VI/PVI setting, we optimize the model for an equal number of *communications* across PVI settings. In both posterior methods, a single client optimization requires 2 communications between client and server: one for the sending the current posterior and one for communicating back the updated client likelihood factor.

In the regression case, we optimize for $4 \times N_{clients}$ communications, meaning we optimize every client twice. In the classification setting, we optimize for $20 \times N_{client}$ communications, meaning that we optimize every client 10 times. This is so that we can evaluate both methods in reasonable computational time. We evaluate the global posterior after every global iteration.

We optimize the client(s) with each $10,000$ local iterations and no early stopping in the regression experiment. In the classification experiment, we optimize the client(s) with each $20,000$ local iterations and apply early stopping after having optimized every client once.

We check for convergence (early stopping) by checking whether the most recent VFE has improved relative to the past 20 reported VFEs. We do not check this every iteration due to the natural stochasticity of mini-batching. Since we only report client VFE and

check for early stopping every 50 iterations, this approximately means that a client has converged when its VFE has not improved in a 1000 gradient updates. Using these settings, we noticed that clients typically converged after $4,000$ iterations or less.

§5.1 Toy Regression

In this section, we evaluate GI and MFVI in multiple settings on the toy regression dataset used by Ober and Aitchison [2021]: $y = x^3 + \epsilon$, $\epsilon \sim \mathcal{N}(0, 9)$, normalizing the y values ($N = 40$). The toy regression experiment serves two roles. Firstly, it shows the dynamics of the sequential and synchronous PVI settings relative to the global VI setting. Secondly, because it is a one-dimensional regression problem, it provides a *visual* canvas to compare the GI method to MFVI by plotting the predictive distribution. In particular, we highlight the improved predictive uncertainty of GI in parts of the domain that were not seen during training and show the inducing point behavior of the GI method.

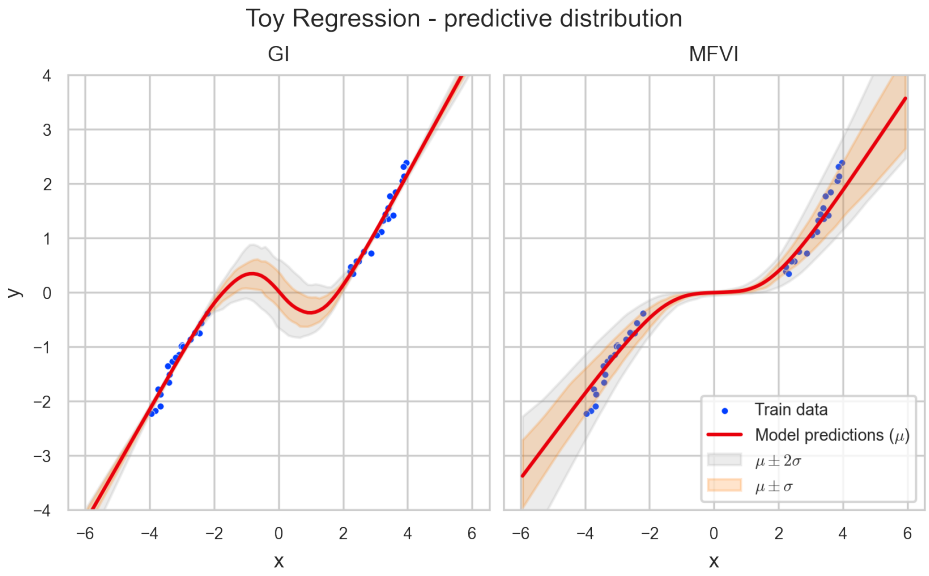


Figure 5.1: Illustration of the predictive uncertainty of both posterior methods for $x \in [-2, 2]$. GI is able to capture predictive uncertainty across parts of the domain not seen during training, whereas MFVI collapses in between training data.

In figure 5.1 we illustrate the improved predictive uncertainty of GI over MFVI, confirming the examples shown by Ober and Aitchison [2021]. GI is able to represent predictive uncertainty in the part of the domain between the training data. In contrast, MFVI's predictive distribution collapses and is overly confident for $x \in [-2, 2]$. We note that because of the small dataset size ($N = 40$), the prior has a more pronounced

effect on the predictive uncertainty; the NealPrior induces a relatively small predictive uncertainty of GI outside the training data distribution on the toy-regression dataset.

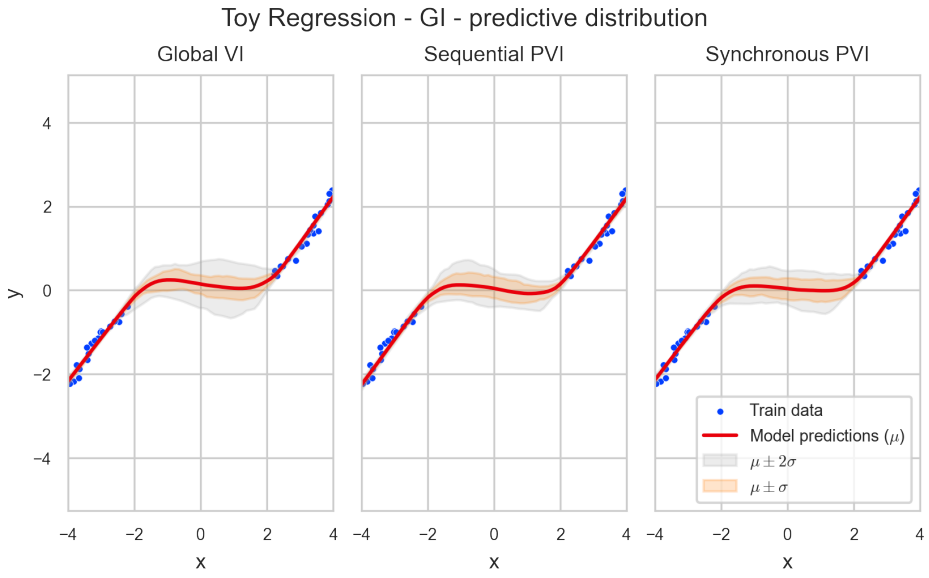


Figure 5.2: Illustration of the posterior convergence of the GI-PVI method. As expected, the GI-PVI method converges to the same predictive distribution as the single-client global VI setting.

Figure 5.2 shows the convergence of the predictive distribution of the GI method under sequential and synchronous PVI to the global VI solution. Ashman et al. [2022] prove that any fixed point solution of the PVI framework is also a local optimum of the global VI algorithm and empirically show that MFVI-PVI recovers the global VI solution. For the GI method, the posterior for sequential and synchronous PVI will converge to, but not be identical to, the global VI solution. This is because every client has their own set of inducing points rather than a single set as in the global VI setting. This is supported by the results in figure 5.2 as the differences between in predictive distributions are minute.[1]

Because the GI method constructs a likelihood factor t via the matrix products XLY and XLX as discussed in §4.3, the number of inducing points M is an effective dimensionality bottleneck on the variational parameters as $\mathbf{V}^\ell \in \mathbb{R}^{D^\ell_{out} \times M}$, $\mathbf{\Lambda}^\ell \in \mathbb{R}^{D^\ell_{out} \times M \times M}$, and $\mathbf{U} \in \mathbb{R}^{M \times D_{in}}$. Because the number of inducing points also has a large impact on the computational cost, it is natural to evaluate the effect of varying M on the predictive distribution. We consider the GI method with $M \in \{1, 5, 10, 40\}$ on the toy regression

[1]Note that the confidence interval is sensitive to the $I = 50$ sampled weights, meaning that multiple draws from the same posterior would also result in slightly different confidence intervals.

dataset and the same $N = 40$ training points. Figure 5.3 shows the inducing points and predictive distribution after training the BNN. We empirically find that on the toy-regression dataset $M \geq 2$ was sufficient to capture the true posterior, but that $M = 1$ causes the predictive distribution to break. Moreover, we note that the number of inducing points empirically does not affect the converged predictive distribution, as evident from the bottom row in figure 5.3. As M increases beyond an arbitrary (dataset-related) threshold, we observe that a large proportion of the inducing points move away from the training data towards $x = 0$, meaning that the inducing input *locations* are set (close) to zero. When an inducing input is 0, it effectively does not contribute to the subsequent layer's output and brings the posterior closer to the prior, satisfying the KL term inside the client's variational objective. This behavior of VI reduces the complexity penalty without incurring a large penalty for increasing the variance in predictions [Trippe and Turner, 2018].

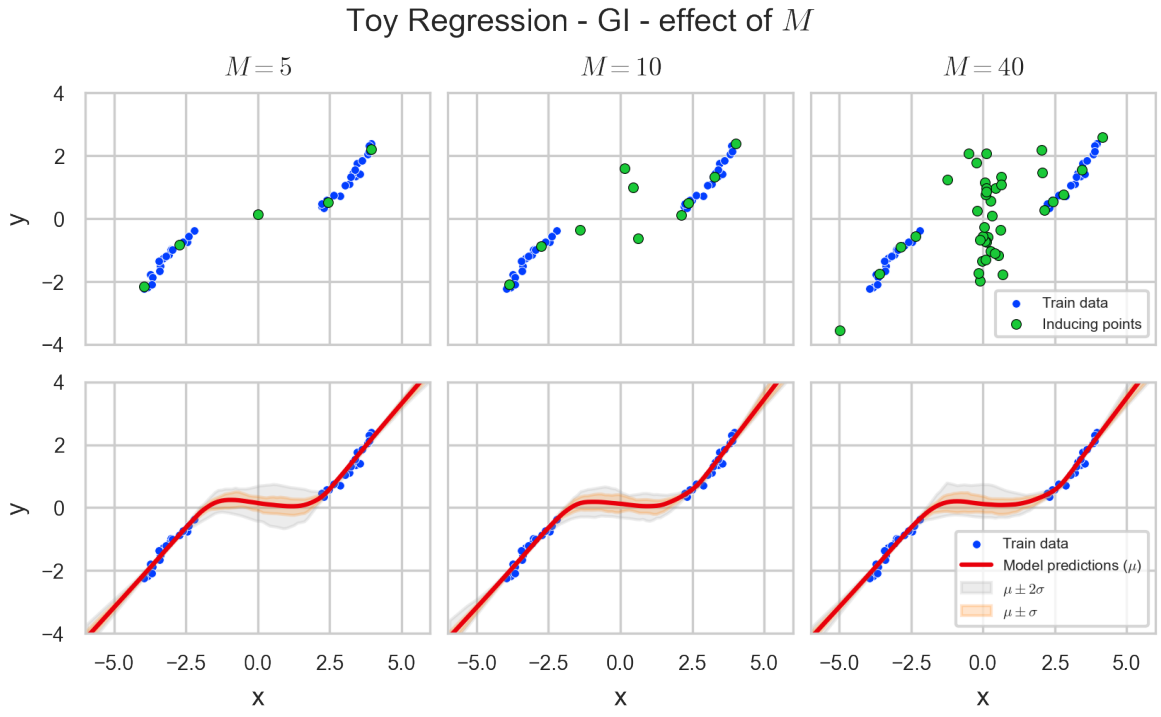


Figure 5.3: Visualization of the inducing point behavior and predictive distribution with varying the number of inducing points M. The predictive distribution appears to be independent of M. As M increases, more inducing points move towards $x = 0$ to satisfy the prior term in the local VFE optimization.

We investigated the distribution of the pseudo-observations's variance at the final layer across the M configurations (figure 5.4), which confirms this behavior. We notice

that as M increases, there are more pseudo-observations with very large variances. These are associated with the inducing locations that are set to zero as $M = 5$ has no large variances. These inducing points effectively sit at the prior and do not contribute (much) to the overall posterior. As we will discuss at the end of this chapter (§5.3), the number of inducing points is an interesting hyperparameter—being an effective bottleneck on the posterior which can affect convergence dynamics in multi-client settings.

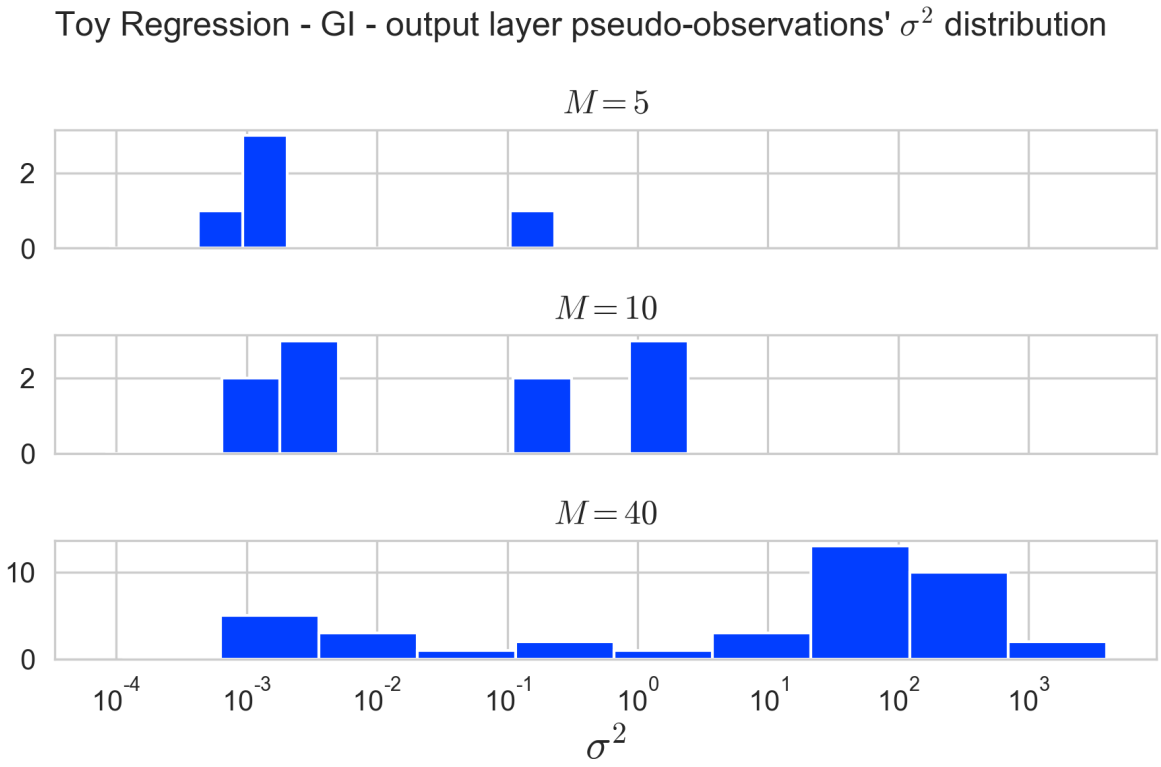


Figure 5.4: Visualization of the distribution of pseudo-observation variance $\mathbf{\Lambda}_{\ell=L}^{-1}$ in the final GI-BNN layer with varying the number of inducing points M. As we increase M, we notice that more pseudo-observations have very large variances, which are associated with the inducing input locations that move towards $x = 0$. These pseudo-observations effectively do not contribute to the posterior and reduce the complexity penalty.

§5.2 Classification

In this section, we evaluate the performance of GI-PVI relative to MFVI-PVI on small-scale classification tasks under both homogeneous and inhomogeneous client data distributions. We use two binary classification datasets from the UCI database: *Adult* and *Bank*. *Adult* has a feature dimensionality of 108 and the task is to determine whether yearly income exceeds 50K USD based on census data. *Bank* has a feature dimensionality of 51 and the task is to determine whether a client will subscribe to a term deposit. The purpose of this experiment is to assess the predictive performance and convergence of said methods under various federated learning settings and client data distributions.

Client data distribution

For each dataset, we use an 80/20 split for training and test data. In the standard VI setting, we only have 1 client and assign all the training data to that client. Therefore, the global VI setting only has a homogeneous data partition. In the federated learning setting, we evaluate two separate data distribution settings among the 10 clients. In partition split A, we homogeneously partition the data in size and label balance. In partition split B, we heterogeneously partition the data amongst the 10 clients identically to the data distribution scheme used by Ashman et al. [2022] for UCI datasets. The degree of heterogeneity is determined by two parameters affecting the label and size imbalance (k and β). When having 10 clients with data split B, these imbalance settings assign half of the clients to have $\beta = 0.95$ of the data and $k = 0.9$ of the positive class labels.

5.2.1 Results

For every experiment, we report the marginal log-likelihood (mll) and accuracy (acc) on the test set to quantitatively evaluate the model fit and out-of-sample predictive performance. In the figures and tables, we refer to the global VI setting as *GVI*, sequential PVI as *SEQ*, and synchronous PVI as *SYNC*.

We first discuss the results of experiments using the homogeneous data partition (split A). Then, we consider the results under the inhomogeneous data partition (split B). We report the marginal log-likelihoods and aggregate run-times across all experiments in table 5.1 and table 5.2, respectively.

Remark (MFVI results)**.** *Many aspects of our experiment settings are inspired by [Ashman et al., 2022]'s investigations of PVI on the UCI datasets. However, it is important to note that their MFVI results are not directly comparable with ours because they use a logistic regression model, which has a noticeably smoother convergence due to the simpler approximate posterior. They note this in their MIMIC-III experiments as well. We also observed smoother convergence and different marginal log-likelihood values when evaluating single-hidden layer BNN models. Moreover, our test sets are not exactly identical to theirs, meaning that small differences in test log likelihoods and accuracies are expected.*

Table 5.1: Marginal log-likelihoods of experiments on test set. Log-likelihoods are reported in nats. Best performing methods are marked in bold.

		split A		split B	
		MFVI	GI	MFVI	GI
Adult	GVI	-0.356	**-0.315**	-	-
	Seq	-0.331	**-0.324**	-0.365	**-0.328**
	Sync	-0.438	**-0.356**	-0.422	**-0.353**
Bank	GVI	-0.250	**-0.213**	-	-
	Seq	-0.235	**-0.232**	-0.239	**-0.225**
	Sync	-0.392	**-0.241**	-0.325	**-0.253**

Homogeneous data split

In figure 5.5, we show the predictive performance of GI and MFVI on the homogeneous client dataset split A. As can be seen from the figure, there is a significant difference in test log likelihood between GI and MFVI in the global VI setting on both datasets after 20 communications (i.e. 200K mini-batch updates across entire dataset). This confirms the predictive performance benefits indicated by Ober and Aitchison [2021].

In the sequential PVI setting, the difference in predictive performance between the two methods is smaller, mainly because sequential GI does not recover its global VI performance. This is something we discuss in §5.3. GI does have a slightly improved communication efficiency over MFVI in the sequential setting, converging in 20 communications versus MFVI's 40 to 50 communications. However, as reported in table 5.2, its wall-clock runtime is about 3x slower than MFVI's—removing any practical benefits in a sequential PVI setting. Practically speaking, this means that sequential MFVI attains

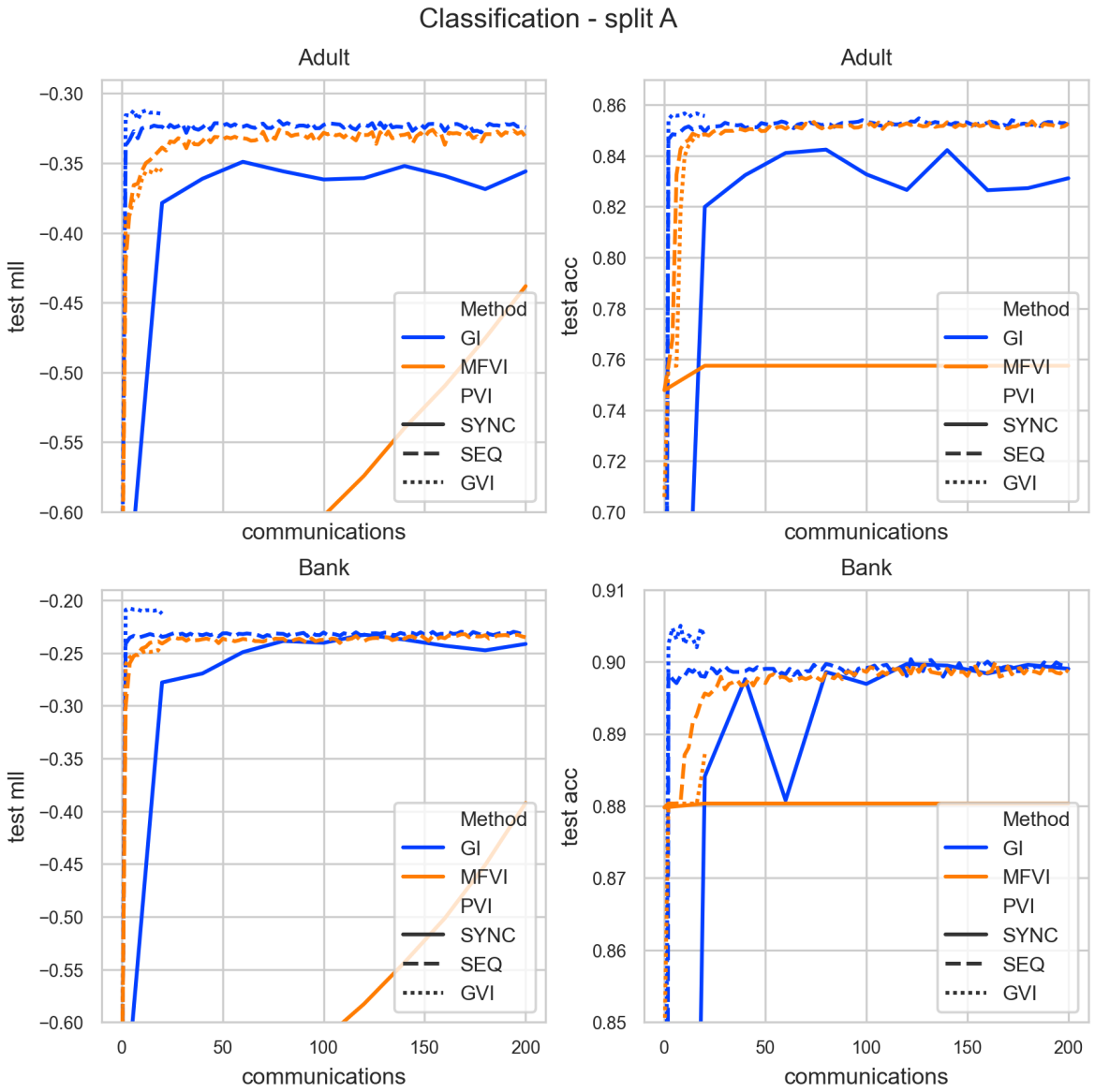


Figure 5.5: Illustration of the predictive performance of BNNs with GI and MFVI approximate posterior methods on *homogeneously* split UCI datasets.

approximately the same level of predictive performance as sequential GI when the data distribution amongst clients is homogeneous.

We do observe predictive performance benefits of GI over MFVI in the synchronous PVI setting, despite GI not entirely recovering even its sequential performance. As can be seen in figure 5.5, synchronous MFVI does not converge to GI's predictive performance in 200 communications: the test set accuracy does not improve beyond 0.76 and 0.88 for Adult and Bank, respectively, and the log likelihoods have not even converged yet. This

is due to the damping necessary in synchronous MFVI (even on split A) which forces client's factor updates to be less pronounced, preventing large disagreement between client factors. As Ashman et al. [2022] note, damped updates lead to a greater number of updates necessary to attain convergence and mention that damping was necessary for their BNN with MFVI in the synchronous PVI setting. We found that synchronous MFVI without damping exhibited very large oscillations in log likelihoods, leading to divergent behavior. As such, damping is a necessary evil in synchronous MFVI. It introduces an additional hyperparameter that requires careful tuning: too conservative damping leads to diverging behavior, while too aggressive leads to much longer wall-clock times and worse communication efficiency. We used the same dampening factor $\rho = 0.2$ as Ashman et al. [2022] for their logistic-regression MFVI model.

The main benefit of synchronous GI over MFVI in the homogeneous split is that it does not require damping and achieves good predictive performance in just a few global iterations. As can be seen from figure 5.5, on the Bank dataset, synchronous GI recovers the same level of predictive performance as sequential GI in 80 communications (4 global iterations). On Adult, the predictive performance is slightly worse than sequential GI, but it converges in 60 communications. In short, the lack of damping improves communication efficiency drastically. This, in combination with the improved (real-world)[2] time-efficiency of synchronous updates, makes synchronous GI a contending method in low-communication environments where we want to quickly train a shared model using just a few updates from every client.

Client dynamics

In 5.6, we show the optimization dynamics of `client0` across all the local optimization steps (*total iterations*). As expected, the behavior of sequential and synchronous PVI on the first client are identical. In the first global iteration, GI converges slightly later and achieves a worse local VFE than MFVI because of the initial KL term. The initial KL term scales with M as the intermediate layers' pseudo-observations and precisions $\mathbf{V}^\ell, \mathbf{\Lambda}^\ell$ are randomly initialized, meaning that the terms inside matrix products XLX and XLY scale with the number of inducing inputs. As such, the resulting means and variances of the initial posterior are much larger (see equation (4.6)) and the initial KL divergence w.r.t the prior is larger.

[2]Note that because we did not parallelize our synchronous PVI models, we cannot speak to the empirical time-efficiency of synchronous GI.

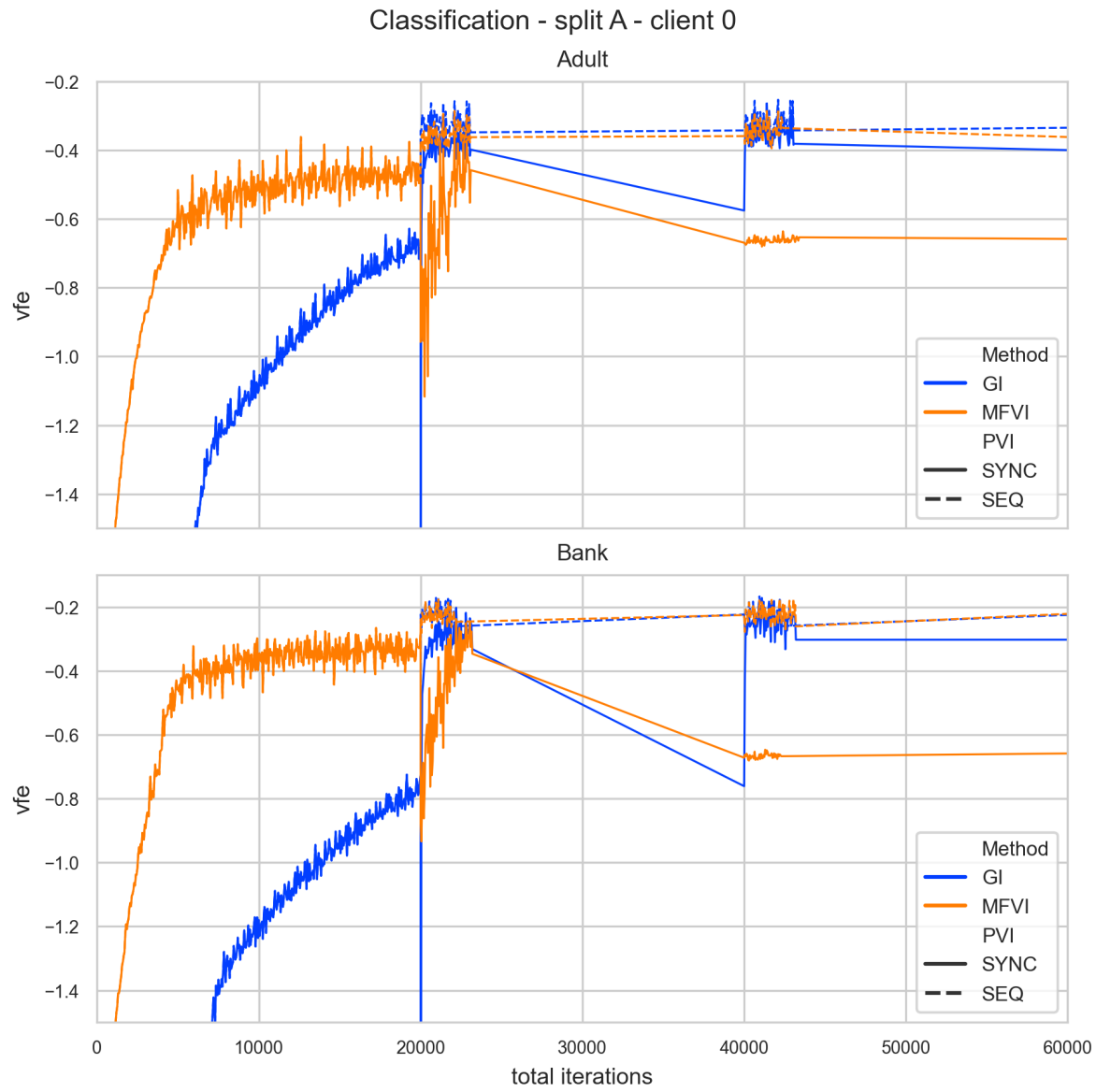


Figure 5.6: Illustration of client-local optimization dynamics when $M = 100$. Every 20K total iterations denotes the start of a new `client0` local optimization. GI appears resilient to clients' factor disagreement post-aggregation. GI has a worse initial local VFE due to its initial KL term that scales with M. Synchronous MFVI, despite the applied damping, does suffer from oscillations after aggregating all clients' factors and takes much longer to recover the local optimum.

Moreover, the number of iterations for a single client optimization needed for GI and MFVI is similar under our early stopping parameters, as can be seen in the figure. This depends on the early stopping parameters, but this suggests that there is no client optimization efficiency improvement of GI relative to MFVI.

Finally, we observe the idiosyncrasies of synchronous MFVI discussed above: where synchronous GI recovers the sequential performance almost immediately, synchronous MFVI needs much longer to recover due to the necessary damping. We note the slight drop in VFE for synchronous GI after the second global iteration, but notice that it immediately recovers the old level of VFE. Synchronous GI quickly recovers from any oscillation after aggregating all clients' factors without the need for damping, showing its resilience as method to client factor disagreement.

Inhomogeneous data split

We show the predictive performance of both posterior methods on the inhomogeneous data split in figure 5.7. As explained in §5.2, the inhomogeneous data split B means that half of the clients collectively have 0.95 of the data with 0.9 of the positive class labels such that there is significant label and size imbalance amongst the clients. Ashman et al. [2022] note that the order in which clients are updated will affect the convergence rate of sequential PVI. In the results below, the first five optimized clients are the "small" clients with collectively 0.05 of the training data with 0.1 of the positive labels; this explains why both sequential methods only converge after having seen a large client. We did not evaluate different client optimization orders as it does not affect the converged predictive performance of either method.

Despite being relatively simple datasets, the results in figure 5.7 demonstrate the comparatively poor performance of the MFVI method in the presence of inhomogeneity across clients. The magnitude of improvements naturally depend on the dataset, but the significance is apparent. In just a 20 communications, GI achieves a 0.859 and 0.838 accuracy on Adult in the sequential and synchronous PVI setting, respectively, in contrast to MFVI's best accuracy of 0.81. On Bank, GI converges in 40 communications and achieves an accuracy of 0.90 versus MFVI's 0.88.

Moreover, sequential MFVI exhibits oscillations due to the inhomogeneity and lack of damping, aligning with the observations made by Ashman et al. [2022]: "sequential [MFVI-PVI] can perform poorly in the presence of inhomogeneous partitions, and can even fail to converge when M is large unless damping is used". This is because the approximate posterior in the sequential case is biased towards the most recently updated client. As a result, sequential MFVI requires around 100 communications on both datasets to converge in log likelihoods, while still achieving worse predictive performance than GI.

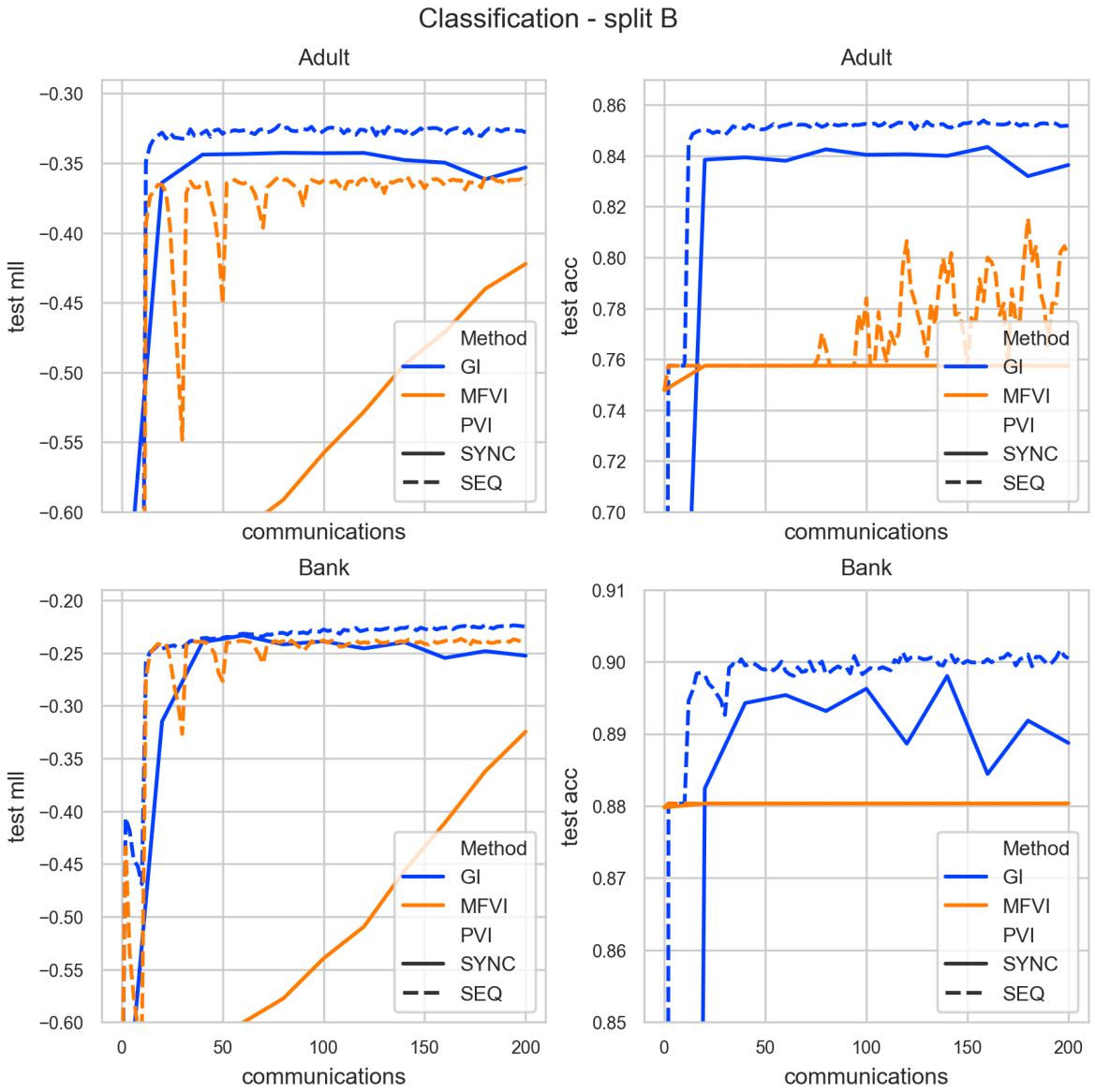


Figure 5.7: Illustration of the predictive performance of BNNs with GI and MFVI approximate posterior methods on *inhomogeneously* split UCI datasets.

Finally, while synchronous MFVI exhibits the same slow convergence behavior due to the damping as in split A, synchronous GI exhibits almost no oscillation in log likelihoods across datasets and recovers the predictive performance of sequential GI almost entirely—without any need for damping.

The performance of GI in presence of inhomogeneity is noteworthy for three reasons. One, GI converges in the sequential setting after just a *single* global iteration. Two, synchronous GI converges, again, without damping and exhibits great performance

after only *two* global iterations. This is significant because we expect client factor disagreement to be more pronounced in light of data inhomogeneity. Three, GI recovers its homogeneous split predictive performance in both PVI settings. Together, this shows the flexibility and effectiveness of GI in the presence of inhomogeneity across clients.

Empirical computational complexity

We ran our classification experiments on a single NVIDIA A100 GPU with two AMD EPYC 7763 64-Core 1.8GHz CPUs (128 cores in total). The GPU has 19.5 TFLOPS of FP32 precision. We report aggregate wall-clock times of the experiments in table 5.2. Please note that the synchronous experiments were not parallelized and are not indicative of parallelized models' time-efficiency in a real-world synchronous PVI setting.

Table 5.2: Wall-clock time of experiments. Time is reported in *(h:mm)*. GI takes only 2x as long as MFVI in a single-client setting, but around 3.5 to 4x in multi-client settings. (Note that synchronous models were not parallelized).

		MFVI	GI	Multiple factor
Adult	GVI	0:08	0:14	1.8x
	Seq	1:17	4:43	3.6x
	Sync	0:57	3:52	4.0x
Bank	GVI	0:07	0:14	2.1x
	Seq	1:12	4:09	3.4x
	Sync	0:56	3:34	3.8x

The multiple factors in table 5.2 reveal the computational complexity of GI. In a single-client setting, GI is only 2x as computationally expensive as MFVI; 20,000 client-local gradient updates with $M = 100$ on Bank took 5.5 minutes, whereas MFVI took 3 minutes. As the number of clients increase, however, this factor increases to 3.5x. As explained in §4.3, this is caused by the additional computation necessary as $N_{clients}$ increases to compute the layer-conditional posterior at the client and server level.

§5.3 Discussion

In the previous section §5.2.1, we detailed the empirical observations from evaluating a BNN with the GI and MFVI method under different settings. In this section, we discuss the implications of our empirical findings.

Let us start with the attractive aspects of GI as approximate posterior method. Firstly, **GI exhibited stable convergence** in both homogeneous and inhomogeneous data partitions and proved resilient to client factor disagreement without the need for damping. This is an extremely desirable model property in federated learning, as non-IID data is ubiquitous in real-world settings and can worsen predictive accuracy in federated point-estimate models by up to 55% [Zhao et al., 2018].

The first explanation for GI's resilience to non-IID data is that the effective number of parameters in GI enable clients to find better local posteriors that fit *both* theirs and other clients' data. The second hypothesis is that superfluous inducing points (beyond what is necessary to fit the client's data) move to satisfy the cavity KL term in the local VFE objective—thereby reducing client factor disagreement and improving convergence. We observed that a larger number of inducing points $M = 100$ improved convergence both in stability and in communication efficiency in the classification task.[3] Increasing M increases the family of posteriors that GI can capture as M is an effective dimensionality bottleneck on the variational parameters. This more flexible approximate posterior enables the client to find a local posterior that fits well to their data *and* agrees with other clients' factors—thereby improving convergence.

The additional possible explanation is that the additional inducing points move to converge the posterior towards the prior. In the regression experiment, we saw that most of the inducing points move to satisfy the KL term in the local VFE. In a multi-client setting, this term is the divergence between the local posterior and the cavity distribution. Satisfying this term means reducing the divergence between the client's factor and other clients' factors, thereby mitigating client factor disagreement. However, it is hard to verify this hypothesis as KL divergence terms are not cross-comparable across various M settings.

Secondly, **GI exhibited high communication efficiency**, recovering converged predictive performance after only a few rounds of communications. In the sequential setting, GI converged after a *single* global iteration, having seen every client exactly once. Synchronous GI only needed *two* to *four* global iterations to achieve convergence.

[3] When using fewer inducing points ($M = 10$), we observed slower convergence and slight oscillations in the first few global iterations of synchronous PVI. See the plots in Appendix A for more detail.

Interestingly enough, synchronous GI needed more communications on homogeneous data than it did on inhomogeneous data. We believe this is optimization-setting related rather than structurally indicative of synchronous GI performance on homogenous data partitions.

Thirdly, **GI showed better predictive performance than MFVI** in both log likelihoods and accuracies, albeit it less pronounced under homogeneous data partitions. Particularly under inhomogeneous data partitions, we observed GI's robustness contrasting MFVI's fragility. In scenarios where client data is heavily inhomogeneously distributed, GI is a solid choice for practitioners, despite its complexity, as MFVI breaks down without dampening.

This brings us to the single disadvantage of GI: its computational cost. GI scales poorly with respect to many hyperparameters, but in particular to the number of clients (discussed in §4.3): GI took 2x as long as MFVI when $N_{clients} = 1$ and 3.5x times as long when $N_{clients} = 10$. This shows how poorly GI scales with the number of clients, because in almost every local update, the client has to recompute *all* the other clients' factors and propagate their inducing points for *every* BNN layer. Theoretically, the client could parallelize the computation of all the clients' factors as well as the inducing point propagation as they are independent of each other—but even this is hard to do when $N_{clients}$ is large. In essence, **the GI method is limited to a small number of clients**.

Despite this, GI-PVI can be a strong method in environments where we want to quickly train a shared model using just a single / few update(s) from every client. Particularly synchronous GI is of interest because of its improved time efficiency in sequential GI. Primarily because of the synchronous updates, but secondarily because of the first global iteration: the first local optimization of *every* client is independent of all other clients' factors—which in sequential setting only holds for the first client. This is because the cavity distribution in the first global iteration merely consists of the prior.

In short, GI is an attractive posterior approximation method in certain settings. Because of the method's complexity, GI is only a viable method when either the computational constraints are relaxed or the number of clients and input dimensionality are small. In these scenarios, GI is a well-performing, robust method that improves over MFVI—particularly under inhomogeneous client data distributions. Especially in synchronous PVI settings, GI offers converged predictive performance after just a few iterations in a time-efficient manner as the computation in the first iteration is independent of the number of clients.

5.3.1 Limitations

Hyperparameter optimization is a well-known and hard problem in federated learning [Kairouz et al., 2021; Khodak et al., 2021, §3.4.1]. Because of the significant wall-time of experiments, it is difficult to tune the hyperparameters as to obtain the best possible performance. Moreover, the complexity of learning a shared posterior in Bayesian federated learning means that the effect of hyperparameters can be even more important. The lack of hyperparameter tuning is most likely also the reason that the GI method did not attain the same level of predictive performance in multi-client settings as it did in the single-client global VI case. Still, we observed the relative outperformance of GI under our—perhaps sub-optimal—hyperparameter settings. Hyperparameter optimization in (Bayesian) federated learning is a computationally expensive task and poor hyperparameter settings can limit the ability for the posterior approximation to capture the true posterior, thereby reducing the predictive performance.

Another inherent limitation of our experiments is the usage of synthetic datasets and client data distributions. It is a standard approach of existing work to take synthetic datasets and create non-IID partitions with distribution skew by partitioning based on labels and size. However, real-world federated datasets likely contain a multitude of non-IID characteristics extending beyond mere label and size imbalance, such as covariate shift, concept shift, and concept drift [Kairouz et al., 2021, §3.1]. Caldas et al. [2019]'s LEAF collection of datasets is a good attempt at establishing a suite of realistic federated datasets, but its adoption is not wide-spread and the underlying datasets are still limited in their representation of real-world federated learning tasks. The lack of open-source, real-world, federated non-IID datasets inherently limits the ability to evaluate posterior approximation methods and assessing their resilience to real-world client heterogeneity.

§5.4 Summary

In this chapter, we presented the main results of our thesis: an empirical evaluation of GI against MFVI as posterior approximation method in the federated learning setting. We compared both methods on a toy-regression task and two classification datasets, evaluating their performance under both homogeneous and inhomogeneous client data splits w.r.t. label and size imbalance. In each experiment, we considered the sequential and synchronous federated learning setting and provided the single-client global VI setting as baseline. We discussed the implications and limitations of our empirical findings. We concluded that GI is a well-performing federated posterior approximation method that is

robust to client heterogeneity. Due to its computational complexity, its practicality is restricted to settings where the number of clients and input dimensionality are small or the computational constraints of clients are relaxed.

Chapter 6

Conclusion

In this work, we set out to extend and evaluate the richer GI posterior approximation method in BNNs under the federated learning setting. To do so, we provided a theoretical computational complexity analysis and an empirical evaluation of the predictive performance and contrasted it to the default mean-field posterior approximation method (MFVI).

Our theoretical analysis highlighted that GI's ability to model cross-layer dependencies via its inducing points increased the communication and computation load relative to MFVI considerably; GI's layer-conditional posterior form means that a client has propagate through *every* clients' inducing points and recompute *all* clients' pseudo-likelihood factors for *every* layer. This means that the downstream communication load and computation load scale with the total number of clients, whereas it is constant for MFVI. Finally, the complexity of sampling a layer's weights for GI is cubic in the layer width (as with any structured covariance matrix) while it is quadratic for MFVI.

Our empirical evaluation of both posterior approximations consisted of a regression task and two classification tasks. On the regression task, we visualized GI's improved ability to model predictive uncertainty in between the training data, as MFVI was overly confident in regions of the domain not seen during training. We also investigated the inducing point behavior of GI as we varied the number of inducing points. We observed that, although the predictive distribution did not change on the toy-regression dataset, more inducing points moved to satisfy the prior KL divergence as we increased the number of inducing points.

In the classification experiments, we evaluated the predictive performance and convergence on two medium-scale datasets from the UCI repository. We considered both homogeneous and inhomogeneous client data partitions and both sequential and syn-

chronous PVI settings. To succinctly summarize our findings, GI improved over MFVI in every aspect but empirical time-efficiency. Across all client data splits and PVI settings, GI demonstrated:

- stable convergence, without the need for damping.
- high communication efficiency, converging after only a few rounds of communications.
- better predictive performance, in terms of both log-likelihoods and accuracy on the test set.

We also observed the rather poor scaling of the GI method with respect to the number of clients. For the same number of communications, GI took 2x and 3.5x as long as MFVI in single- and ten-client settings, respectively. However, GI's improved communication efficiency means that the time-efficiency is better than that. Particularly in a synchronous PVI setting, GI can offer excellent predictive performance in a time-efficient manner due to its synchronous updates and the fact that the computation in the first global iteration is independent of the number of clients.

In general, federated learning is a challenging problem setting, and even more so from a Bayesian perspective. The task of approximating the true posterior remains computationally expensive, independent of the approximating method. As we showed in this work, the default mean-field posterior approximation method underperforms or breaks down under client heterogeneity. Particularly in real-world settings, we suspect MFVI to be inviable as client heterogeneity is likely more significant than we were able to capture in our experiments. Considering this, the robustness and excellent predictive performance of GI-PVI makes it a very attractive method for federated BNNs. At the same time, GI's computational complexity possibly limits its practicality to smaller-scale environments.

§6.1 Future opportunities

Concluding our findings, we discuss the future research directions concerning the GI method that we believe are fruitful. We believe that future opportunities lie in reducing the computational cost of deploying GI in a BNN without sacrificing the method's performance.

One option would be to use the GI method only in the final layer of a federated BNN. For example, Ober and Aitchison [2021] report the scores of a BNN with factorized

posteriors in the early layers followed by a GI layer. In their UCI experiments, this model has very competitive log likelihoods and RMSEs on the regression datasets—often outperforming the factorized and global inducing models. This model would limit the computational cost of the GI method to just a single layer, rather than every layer.

Another setting in which we believe the GI method can be deployed effectively is the domain of personalized federated learning (PFL). In short, PFL methods seek to combine the benefits of a shared global model with client-locally trained models. For more information on PFL, a complete taxonomy is provided in Tan et al. [2022]. Inspired by Zhang et al. [2022]'s approach of using a trained global model as prior distribution for client-local model optimization, GI can be deployed as posterior approximation method at the client optimization level since the number of clients would be reduced to just one. A cheap global model could be trained on all the clients using MFVI and could be used as prior for the client-local posterior approximation method. Together, the expectation is that the GI method refines the shared global model around the client's data while limiting the computational cost.

Future directions related to computational improvements include reducing the client-computation load from scaling with the total number of clients. One possible direction is to investigate the effect of restricting the number of included clients in every communicated posterior by the server. By subsampling the client factors that are included in the communicated posterior and cavity distribution, we limit the client computation only to scale with the number of included clients in the posterior. Intuitively, this means that each client in a global iteration update would only optimize its factor with respect to its own data and the clients included in the cavity distribution (and the prior). By randomly subsampling for every global iteration, we still ensure that the client's factor does not move far away from all the other clients' factors. At the cost of likely increasing the number of communications necessary to converge, this would significantly reduce the client computation load and scale the GI-PVI method up to a larger number of clients.

References

Ahn, S., Korattikara, A., Liu, N., Rajan, S., and Welling, M. (2015). Large-Scale Distributed Bayesian Matrix Factorization using Stochastic Gradient MCMC.

Ahn, S., Shahbaba, B., and Welling, M. (2014). Distributed Stochastic Gradient MCMC. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1044–1052. PMLR.

Angelino, E., Johnson, M. J., and Adams, R. P. (2016). Patterns of Scalable Bayesian Inference.

Ashman, M., Bui, T. D., Nguyen, C. V., Markou, S., Weller, A., Swaroop, S., and Turner, R. E. (2022). Partitioned Variational Inference: A Framework for Probabilistic Federated Learning. *arXiv:2202.12275 [cs, stat]*.

Ba, J., Erdogdu, M. A., Ghassemi, M., Suzuki, T., Sun, S., Wu, D., and Zhang, T. (2022). Towards Characterizing the High-dimensional Bias of Kernel-based Particle Inference Algorithms. In *Second Symposium on Advances in Approximate Bayesian Inference*.

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

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2019). LEAF: A Benchmark for Federated Settings. *arXiv:1812.01097 [cs, stat]*.

Challis, E. and Barber, D. (2013). Gaussian Kullback-Leibler approximate inference. *The Journal of Machine Learning Research*, 14(1):2239–2286.

Coker, B., Bruinsma, W. P., Burt, D. R., Pan, W., and Doshi-Velez, F. (2022). Wide Mean-Field Bayesian Neural Networks Ignore the Data. *arXiv:2202.11670 [cs, stat]*.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2018). Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning.

el Mekkaoui, K., Mesquita, D., Blomstedt, P., and Kaski, S. (2021). Federated stochastic gradient Langevin dynamics. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 1703–1712. PMLR.

Farquhar, S., Smith, L., and Gal, Y. (2021). Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations.

Foong, A., Burt, D., Li, Y., and Turner, R. (2020). On the Expressiveness of Approximate Inference in Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 15897–15908. Curran Associates, Inc.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Ghahramani, Z. and Attias, H. (2000). Online variational bayesian learning. In *Slides from Talk Presented at NIPS Workshop on Online Learning*.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks.

Hoffman, M., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic Variational Inference.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). An Introduction to Variational Methods for Graphical Models. In Jordan, M. I., editor, *Learning in Graphical Models*, pages 105–161. Springer Netherlands, Dordrecht.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021). Advances and Open Problems in Federated Learning.

Kassab, R. and Simeone, O. (2021). Federated Generalized Bayesian Learning via Distributed Stein Variational Gradient Descent.

Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Khodak, M., Tu, R., Li, T., Li, L., Balcan, M.-F., Smith, V., and Talwalkar, A. (2021). Federated Hyperparameter Tuning: Challenges, Baselines, and Connections to Weight-Sharing.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational Dropout and the Local Reparameterization Trick.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., and He, B. (2021). A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

Lindinger, J., Reeb, D., Lippert, C., and Rakitsch, B. (2020). Beyond the Mean-Field: Structured Deep Gaussian Processes Improve the Predictive Uncertainties. In *Advances in Neural Information Processing Systems*, volume 33, pages 8498–8509. Curran Associates, Inc.

Louizos, C. and Welling, M. (2017). Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2218–2227. PMLR.

MacKay, D. J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data.

Minka, T. (2004). Power EP. Technical Report MSR-TR-2004-149, Microsoft.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY.

Ober, S. W. and Aitchison, L. (2021). Global inducing point variational posteriors for Bayesian neural networks and deep Gaussian processes. *arXiv:2005.08140 [cs, stat]*.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models.

Salimbeni, H. and Deisenroth, M. (2017). Doubly Stochastic Variational Inference for Deep Gaussian Processes. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. (2022). Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–17.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly Stochastic Variational Bayes for non-Conjugate Inference. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1971–1979. PMLR.

Trippe, B. and Turner, R. (2018). Overpruning in Variational Bayesian Neural Networks. *arXiv:1801.06230 [stat]*.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, Madison, WI, USA. Omnipress.

Wilson, A. G. and Nickisch, H. (2015). Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP).

Zhang, X., Li, Y., Li, W., Guo, K., and Shao, Y. (2022). Personalized Federated Learning via Variational Bayesian Inference.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. (2018). Federated Learning with Non-IID Data.

Appendix A

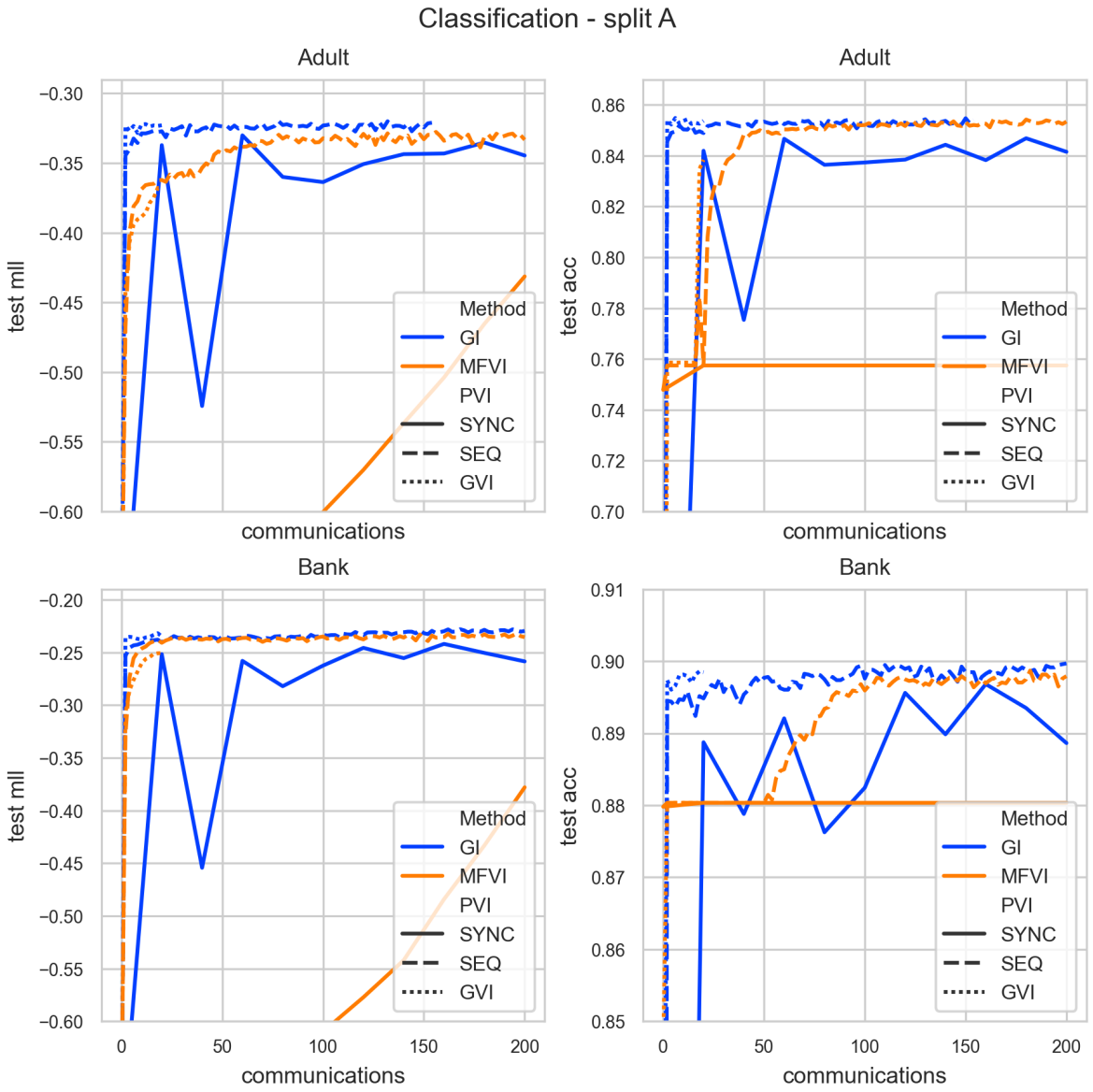


Figure A.1: Illustration of the predictive performance of BNNs with GI and MFVI approximate posterior methods on *homogeneously* split UCI datasets. GI model has $M = 10$ inducing points and $10,000$ maximum local updates. While still outperforming MFVI, GI exhibits more oscillations and slower convergence than with $M = 100$.

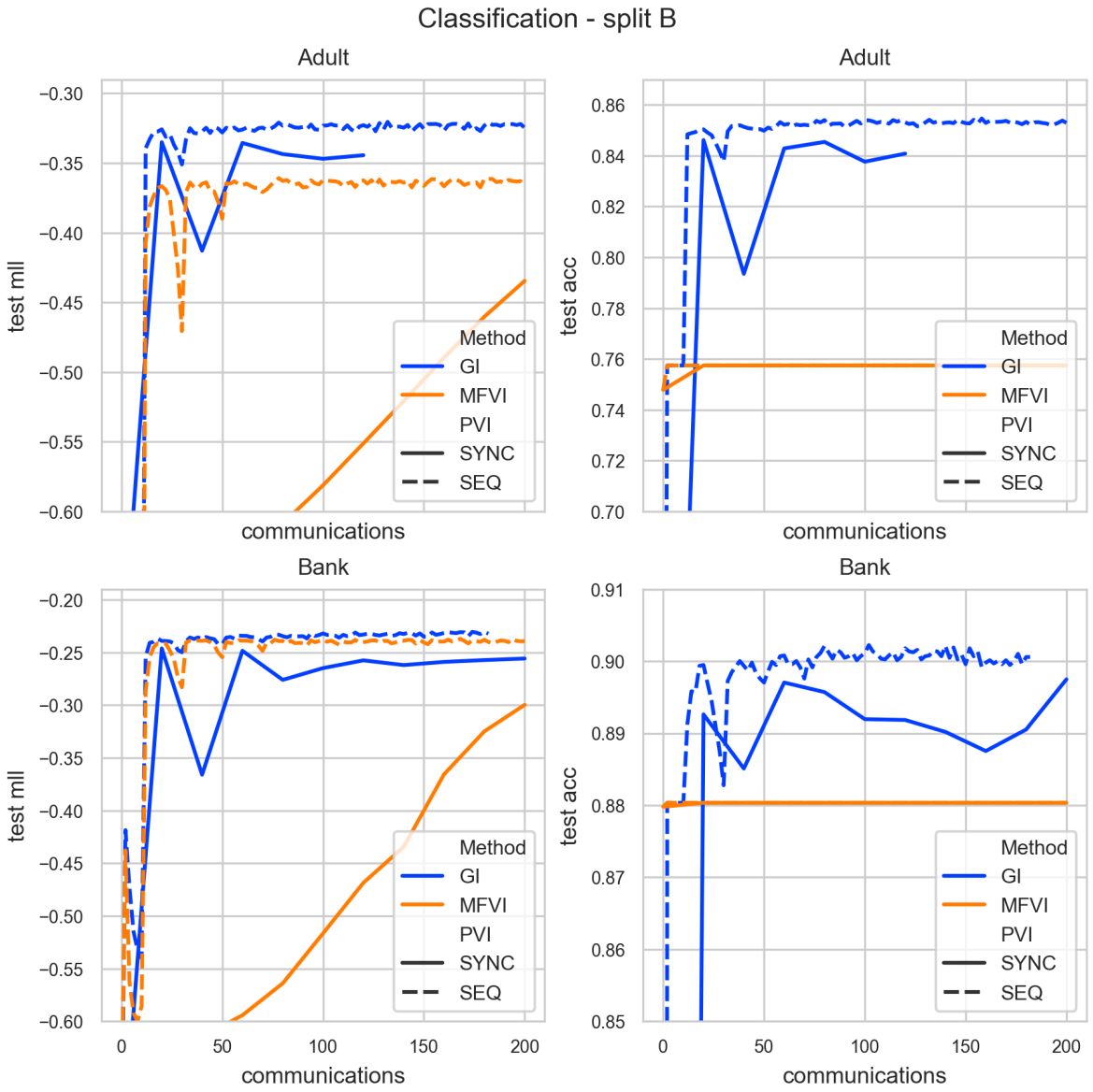


Figure A.2: Illustration of the predictive performance of BNNs with GI and MFVI approximate posterior methods on *inhomogeneously* split UCI datasets. GI model has $M = 10$ inducing points and $10,000$ maximum local updates. While still outperforming MFVI, GI exhibits more oscillations and slower convergence than with $M = 100$.

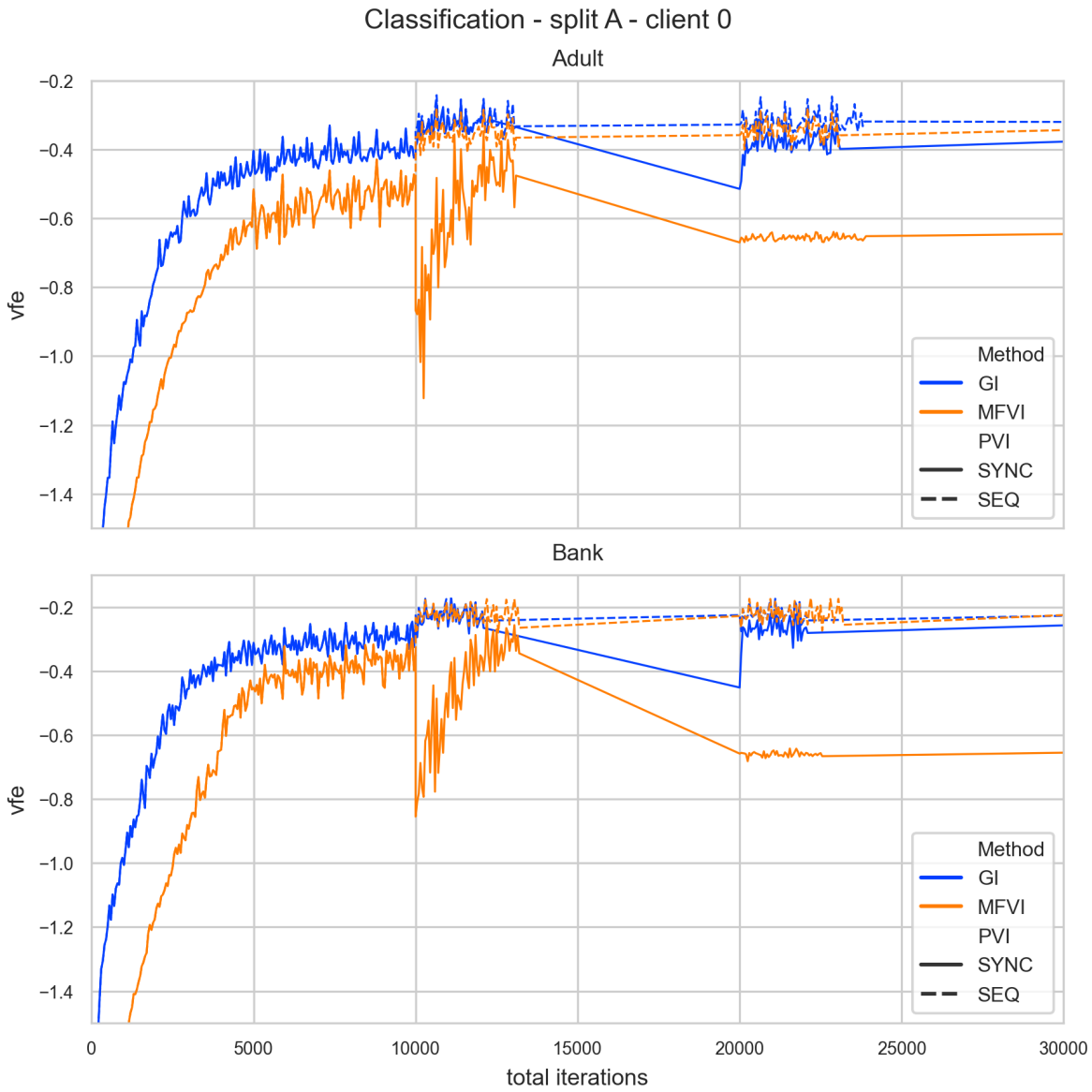


Figure A.3: Illustration of client-local optimization dynamics when $M = 10$. Every 10K total iterations denotes the start of a new `client0` local optimization. GI converges to a better local VFE and appears resilient to clients' factor disagreement post-aggregation. Synchronous MFVI, despite the applied damping, does suffer from oscillations after aggregating all clients' factors and takes much longer to recover the local optimum.