Learn Privacy-friendly Global Gaussian Processes in Federated Learning

by

Haolin Yu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 1, Section 2.1.4, and Chapter 3 are based on this NeurIPS 2022 submission (link), where I am the first author. This research was developed in collaboration with my supervisor Pascal Poupart. I derived all the equations, developed all the algorithms, and wrote all the text in the thesis.

Abstract

In the era of big data, Federated Learning (FL) has drawn great attention as it naturally operates on distributed computational resources without the need of data warehousing. Similar to Distributed Learning (DL), FL distributes most computational tasks to end devices, but emphasizes more on preserving the privacy of clients. In other words, any FL algorithm should not send raw client data, if not the information about them, that could leak privacy. As a result, in typical scenarios where the FL framework applies, it is common for clients to have or obtain insufficient training data to produce an accurate model. To decide whether a prediction is trustworthy, models that provide not only point estimations, but also some notion of confidence are beneficial. Gaussian Process (GP) is a powerful Bayesian model that comes with naturally well-calibrated variance estimations. However, it is challenging to learn a stand-alone global GP since merging local kernels leads to privacy leakage. To preserve privacy, previous works that consider federated GPs avoid learning a global model by focusing on the personalized setting or learning an ensemble of local models.

In this work, we present Federated Bayesian Neural Regression (FedBNR), an algorithm that learns a scalable stand-alone global federated GP that respects clients' privacy. We incorporate deep kernel learning and random features for scalability by defining a unifying random kernel. We show this random kernel can recover any stationary kernel and many non-stationary kernels. We then derive a principled approach of learning a global predictive model as if all client data is centralized. We also learn global kernels with knowledge distillation methods for non-identically and independently distributed (non-i.i.d.) clients. We design synthetic experiments to illustrate scenarios where our model has a clear advantage and provide insights into the rationales. Experiments are also conducted on real-world regression datasets and show statistically significant improvements compared to other federated GP models.

Acknowledgements

First, I would like to thank my supervisor, Professor Pascal Poupart, for being constantly supportive and patient throughout my Master's degree. His valuable experience and suggestions always save me from anxiety towards my research, and his rigorous work attitude has inspired my determination at academic career. Overall, he is the researcher that I respect the most.

I would like to thank my collaborators, Kaiyang Guo, Mahdi Karami, Guojun Zhang, Xi Chen, and Mohsin Hasan for providing precious feedback and ideas and enduring my uncommon work schedule.

I would like to thank my enlightened parents Lixin Niu, Lijun Zhang, my therapist Yang Chen, and my cute Ragdoll for their constant emotional support that makes life lovable.

I would like to thank all my friends that share common topics with me, and all my roommates for not infecting me with COVID-19 yet.

Last but not the least, I would like to thank my partner Renjian Zhang for your continual company, though we are currently on opposite sides of the earth. You helped me through my worst days and convinced me I deserve love. You are my best friend and my anchor of life. I would not have achieved my accomplishments without your support. I wish you succeed in your postgraduate entrance examination and start working on your research interest soon.

Dedication

This is dedicated to my parents and my significant other.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

In Federated Learning (FL) [28], we seek to train a model in a distributed way across several clients without any data leaving the clients to preserve privacy. This is particularly attractive in application domains where each client has insufficient data to train a strong model by itself and therefore could benefit from additional information from other clients. A trusted server is often used to aggregate the client models into a global model that improves upon the local models. Since each client has limited data, its local model is uncertain and therefore there is value in representing this uncertainty to improve the aggregation at the server. Furthermore, uncertainty modeling can be used to derive confidence estimates with respect to predictions.

Although confidence estimates are often desired by managers in the industry, currently most FL algorithms are based on optimization techniques [28, 22, 55, 17, 24, 54, 29, 23]. These non-Bayesian techniques often provide only point estimates for regression tasks, which gives us no information about how confident we should be about their predictions. Even for classification tasks, simply taking the normalized probabilities calculated from the predictive logits as the confidence score empirically yields models that is calibrated worse than Bayesian models [1]. Thus, our work focuses on training a powerful Bayesian model, Gaussian Processes (GPs), which provides not only point estimates but also naturally well-calibrated uncertainty estimates.

GPs with deep kernel learning [61, 59] provide a good balance between expressiveness and complexity to represent model uncertainty. At one end of the spectrum, most models

such as traditional neural networks do not capture any uncertainty, but are simple and scalable. At the other end of the spectrum, most modern Bayesian techniques such as the Bayesian neural networks express a full distribution over all weights of neural networks, but inference tends to be intractable. In between, a GP with a deep kernel consists of a neural feature extractor (also known as deep kernel) and a GP with some conventional kernel, such as the Gaussian kernel, whose inputs are the extracted features, so the resulting composition kernel can benefit both from the scalability of the non-Bayesian neural network and the well-defined uncertainty notion of the simple GP. Besides, if one scarifies the kernel lifting power (i.e. choosing an identity kernel) for simplicity, GPs with deep kernels can degenerate into "neural linear" models [36, 32], where a distribution over the last linear layer that facilitates exact Bayesian Linear Regression (BLR) inference is used to replace the GP. Since the weights of the last layer are the most important for prediction, representing their uncertainty is often sufficient to capture most of the uncertainty of a model.

Several works have explored distributed GPs [8, 64] and federated GPs [1], but they struggle with privacy risks that FL tries to reduce. While distributed GPs are designed to improve scalability, they pose an important privacy risk, since sharing kernels either implies sharing data or sharing pairwise data similarity. In contrast, pFedGP [1] shares only the hyperparameters of deep kernels while learning local GPs that are never shared. This personalized approach reduces privacy risks, but the local GPs do not benefit from other client information (beyond the shared kernel hyperparameters).

In short, we seek an FL algorithm that learns a global GP without sending kernels or features outside clients.

1.2 Contribution

We propose a new federated GP technique called Federated Bayesian neural regression (FedBNR) that can learn a global GP with reduced privacy risks. We avoid kernel sharing by working directly in the feature space and sharing scatter matrices (instead of kernels). We also propose a unifying random kernel (URK) that leverages random features and deep kernels to approximate any stationary kernel and some non-stationary kernels, including infinite kernels. The contributions of this thesis can be summarized as follows:

- New federated GP technique with deep kernel learning called federated Bayesian neural regression (FedBNR). To our knowledge, this is the first federated GP technique that learns a global GP. We describe an exact aggregation technique of the linear

layer that allows inference in a way that is mathematically equivalent to inference with all the data centralized.

- New unifying random kernel (URK) that provides a unifying definition for deep random kernels. URK can approximate any stationary kernel and many non-stationary kernels, including infinite kernels. This kernel has finitely many features and therefore allows us to work directly in the feature space (instead of the dual space).
- Experiments on real world regression datasets where we achieve statistically significant improvements over prior techniques both in terms of predictions and expected calibration error.

Chapter 2

Background and Related Work

In this chapter, we carefully review the FL framework, the GP model, and other techniques based on which we developed our method.

Notation. We will use the following notations throughout the paper. $\mathbf{X}, \mathbf{X}' \in \mathbb{R}^{p \times n}$, $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$ are input matrices or vectors. $\mathbf{y} \in \mathbb{R}^n$ is the target vector. A $*$ subscript indicates the vectors have not been seen by the models. $\sigma \in \mathbb{R}$ is the noise level of Bayesian models. $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ is the zero vector. Their dimensions can be inferred from the context. $\phi : \mathbb{R}^{p \times \bar{a}} \to \mathbb{R}^{p' \times \bar{a}}$ denotes some basis function, and $\mathbf{\Phi} = \phi(\mathbf{X})$ is the corresponding features of $\mathbf{X}$. $k : \mathbb{R}^{p \times \bar{a}} \times \mathbb{R}^{p \times \bar{a}} \to \mathbb{R}^{\bar{a} \times \bar{a}}$ denotes a kernel function, and $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$ is the corresponding kernel matrix. $\bar{a}$ is a placeholder that indicates the function can take in matrices or vectors of any dimension. $\mathbf{w} \in \mathbb{R}^{p'}$ is the weights of the last linear layer in a Bayesian linear regression model. $\mathbf{\Sigma} \in \mathbb{R}^{p' \times p'}$ is its prior covariance matrix. $\mathbb{E}[\cdot], \mathrm{Cov}(\cdot, \cdot)$, $|\cdot|$, and $\mathrm{tr}(\cdot)$ are the expectation, covariance, determinant, and trace function respectively. Any symbol with a c subscript is a local version of the the original symbol, held by some client c.

2.1 Gaussian Processes

2.1.1 Bayesian Linear Regression

Bayesian linear regression (BLR) is the foundation of GPs. It assumes the true underlying function $f(\cdot)$ is linear in some latent feature space induced by the basis functions $\phi(\cdot)$, and

the observations y are noisy:

$$y = f(\mathbf{x}) + \epsilon = \phi(\cdot)^\top \mathbf{w} + \epsilon, \text{ where } \epsilon \sim N(0, \sigma^2) \tag{2.1}$$

In contrast to non-Bayesian linear regression, BLR assumes the linear weights $\mathbf{w}$ is a random vector, and its probability distribution, called the **prior**, should be specified before training based on prior knowledge. For simplicity and tractability, the prior is often assumed Gaussian with zero mean and some covariance matrix $\mathbf{\Sigma}$:

$$\Pr(\mathbf{w}) = N(\mathbf{0}, \mathbf{\Sigma}) = |2\pi\mathbf{\Sigma}|^{-1/2} \exp\left(-(\mathbf{w}-\mathbf{0})^\top \mathbf{\Sigma}^{-1}(\mathbf{w}-\mathbf{0})/2\right) \tag{2.2}$$

Then the **likelihood** of observed data naturally follows as:

$$\Pr(\mathbf{y}|\mathbf{X}, \mathbf{w}) = N(\mathbf{\Phi}^\top \mathbf{w}, \sigma^2 \mathbf{I}) \tag{2.3}$$

To fit a BLR model, we estimate the **posterior** of $\mathbf{w}$ given the inputs and targets $\mathbf{X}, \mathbf{y}$, using Bayes' theorem:

$$\begin{aligned}
\Pr(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto \Pr(\mathbf{w}) \Pr(\mathbf{y}|\mathbf{X}, \mathbf{w}) \\
&\propto \exp\left(-\mathbf{w}^\top \mathbf{\Sigma}^{-1}\mathbf{w}/2\right) \exp\left(-(\mathbf{y}-\mathbf{\Phi}^\top\mathbf{w})^\top(\mathbf{y}-\mathbf{\Phi}^\top\mathbf{w})/(2\sigma^2)\right) \\
&= \exp\left(-\mathbf{w}^\top \mathbf{\Sigma}^{-1}\mathbf{w}/2 - (\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{\Phi}^\top\mathbf{w} + \mathbf{w}^\top\mathbf{\Phi}\mathbf{\Phi}^\top\mathbf{w})/(2\sigma^2)\right) \\
&\propto \exp\left(-(\mathbf{w}^\top(\sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^\top + \mathbf{\Sigma}^{-1})\mathbf{w} - 2\sigma^{-2}\mathbf{y}^\top\mathbf{\Phi}^\top\mathbf{w})/2\right) \\
&= \exp\left(-(\mathbf{w}^\top\mathbf{A}\mathbf{w} - 2(\sigma^{-2}\mathbf{y}^\top\mathbf{\Phi}^\top\mathbf{A}^{-1})\mathbf{A}\mathbf{w})/2)\right) \\
&\propto \exp\left(-(\mathbf{w}^\top\mathbf{A}\mathbf{w} - 2\bar{\mathbf{w}}^\top\mathbf{A}\mathbf{w})/2)\right) \\
&\propto \exp\left(-(\mathbf{w}-\bar{\mathbf{w}})^\top\mathbf{A}(\mathbf{w}-\bar{\mathbf{w}})/2\right)
\end{aligned}$$

where $\bar{\mathbf{w}} = \sigma^{-2}\mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y}$ and $\mathbf{A} = \sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^\top + \mathbf{\Sigma}^{-1}$. Since the multiplication of two Gaussian probability density distributions (pdfs) is still a Gaussian pdf, we have:

$$\Pr(\mathbf{w}|\mathbf{X}, \mathbf{y}) = N(\bar{\mathbf{w}}, \mathbf{A}^{-1}) \tag{2.4}$$

To provide more than point estimations, the **prediction** probability distribution can be inferred from Equation 2.1:

$$(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \phi(\cdot)^\top(\mathbf{w}|\mathbf{X}, \mathbf{y}) + \epsilon \tag{2.5}$$

Here $(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ and $(\mathbf{w}|\mathbf{X}, \mathbf{y})$ are two new conditional random vectors that both have Gaussian distributions. Since ϵ is independent of $\mathbf{w}$, by the linear combination rules of multivariate Gaussian random variables:

$$\Pr(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = N(\phi(\mathbf{x}_*)^\top\bar{\mathbf{w}}, \sigma^2 + \phi(\mathbf{x}_*)^\top\mathbf{A}^{-1}\phi(\mathbf{x}_*)) \tag{2.6}$$

The complexity of BLR is linear in the amount of data, but cubic in the number of features due to the inversion of $\mathbf{A}$. If ϕ contains any hyperparameters, they can be learnt by maximizing the log marginal likelihood. According to Bishop's formulation, Equation (3.86) in [3]:

$$\log \Pr_{\text{BLR}}(\mathbf{y}|\mathbf{X}) \propto \log \int \Pr(\mathbf{y}|\mathbf{X}, \mathbf{w}) \Pr(\mathbf{w}) d\mathbf{w} \tag{2.7}$$

$$= -n \log \sigma^2 - \log |\mathbf{\Sigma}| - \log |\mathbf{A}| - \mathbf{y}^\top \mathbf{y}/\sigma^2 + \mathbf{w}^\top \mathbf{A} \mathbf{w} \tag{2.8}$$

2.1.2 Kernel Trick

It is possible to overcome the cubic complexity in the number of features by modifying the order of matrix multiplications. For simplicity, we choose $\mathbf{\Sigma} = \lambda^2 \mathbf{I}$ in the following derivation, since it is the only case we will work with later when deriving our approach. See Appendix A.1 for details. We first introduce two matrix identities.

Lemma 1 (push-through identity [14])**.** $\forall\ \mathbf{U} \in \mathbb{R}^{n \times k}, \mathbf{V} \in \mathbb{R}^{k \times n}, \lambda \in \mathbb{R}$ *such that the inversions exist:*

$$(\lambda^2 \mathbf{I} + \mathbf{U}\mathbf{V})^{-1} \mathbf{U} = \mathbf{U} (\lambda^2 \mathbf{I} + \mathbf{V}\mathbf{U})^{-1}$$

Lemma 2 (Woodbury matrix identity [62])**.** $\forall\ \mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{C} \in \mathbb{R}^{k \times k}, \mathbf{U} \in \mathbb{R}^{n \times k}, \mathbf{V} \in \mathbb{R}^{k \times n}$ *such that the inversions exist:*

$$(\mathbf{A} + \mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}$$

Then we can rewrite the prediction mean and variance in Equation 2.6 as follows:

$$\phi(\mathbf{x}_*)^\top \bar{\mathbf{w}} = \phi(\mathbf{x}_*)^\top \sigma^{-2} (\sigma^{-2} \mathbf{\Phi}\mathbf{\Phi}^\top + \lambda^{-2} \mathbf{I})^{-1} \mathbf{\Phi} \mathbf{y} \tag{2.9}$$

$$= \phi(\mathbf{x}_*)^\top \mathbf{\Phi} (\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda^{-2} \sigma^2 \mathbf{I})^{-1} \mathbf{y}, \text{ by Lemma 1} \tag{2.10}$$

$$= \lambda^2 \phi(\mathbf{x}_*)^\top \mathbf{\Phi} (\lambda^2 \mathbf{\Phi}^\top \mathbf{\Phi} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} \tag{2.11}$$

$$\phi(\mathbf{x}_*)^\top \mathbf{A}^{-1} \phi(\mathbf{x}_*) = \phi(\mathbf{x}_*)^\top (\sigma^{-2} \mathbf{\Phi}\mathbf{\Phi}^\top + \lambda^{-2} \mathbf{I})^{-1} \phi(\mathbf{x}_*) \tag{2.12}$$

$$= \sigma^2 \phi(\mathbf{x}_*)^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda^{-2} \sigma^2 \mathbf{I})^{-1} \phi(\mathbf{x}_*) \tag{2.13}$$

$$= \lambda^2 \phi(\mathbf{x}_*)^\top (\mathbf{I} - \mathbf{\Phi}(\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda^{-2} \sigma^2 \mathbf{I})^{-1} \mathbf{\Phi}^\top) \phi(\mathbf{x}_*), \text{ by Lemma 2} \tag{2.14}$$

$$= \lambda^2 (\phi(\mathbf{x}_*)^\top \phi(\mathbf{x}_*) - \phi(\mathbf{x}_*)^\top \mathbf{\Phi} (\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda^{-2} \sigma^2 \mathbf{I}))^{-1} \mathbf{\Phi}^\top \phi(\mathbf{x}_*) \tag{2.15}$$

$$= \lambda^2 \phi(\mathbf{x}_*)^\top \phi(\mathbf{x}_*) - (\lambda^2 \phi(\mathbf{x}_*)^\top \mathbf{\Phi})(\lambda^2 \mathbf{\Phi}^\top \mathbf{\Phi} + \sigma^2 \mathbf{I})^{-1} (\lambda^2 \mathbf{\Phi}^\top \phi(\mathbf{x}_*)) \tag{2.16}$$

Notice in the new formulations, features of either the training or testing data always appear in the form of inner products $\phi^\top\phi$. We can further simplify the equations by defining $\phi'(\cdot) = \lambda\phi(\cdot)$, and a kernel function $k(\cdot,\cdot) = \phi'(\cdot)^\top\phi'(\cdot)$. Specially, we call $\mathbf{K} = k(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{n\times n}$ the kernel matrix or the gram matrix. Then,

$$\phi(\mathbf{x}_*)^\top\bar{\mathbf{w}} = k(\mathbf{x}_*,\mathbf{X})(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} \tag{2.17}$$

$$\phi(\mathbf{x}_*)^\top\mathbf{A}^{-1}\phi(\mathbf{x}_*) = k(\mathbf{x}_*,\mathbf{x}_*) - k(\mathbf{x}_*,\mathbf{X})(\mathbf{K}+\sigma^2\mathbf{I})^{-1}k(\mathbf{X},\mathbf{x}_*) \tag{2.18}$$

Instead of calculating $\phi'(\cdot)^\top\phi'(\cdot)$ explicitly, it is sufficient to know just the results of the kernel function evaluated on pairs of inputs. That is, we can directly specify the function $k(\cdot,\cdot)$ without knowing what features it induces as long as it meets the following condition.

Theorem 1 (Characterisation of kernels [41])**.** *A function $k : X \times X \to \mathbb{R}$ can be decomposed into the inner product of feature mappings $k(\mathbf{x},\mathbf{x}') =< \phi(\mathbf{x}),\phi(\mathbf{x}') >$ if and only if it is a symmetric function for which the matrices formed by restriction to any finite subset of the space X are positive semi-definite.*

In short, Theorem 1 states that a function is a valid kernel function if and only if it is positive semi-definite. A kernel is called stationary if it solely depends on the distance between the pair of inputs $\exists f : k(\mathbf{x},\mathbf{x}') = f(\mathbf{x}-\mathbf{x}')$, otherwise non-stationary. A popular stationary kernel is the Radial Basis Function (RBF) kernel $k(\mathbf{x},\mathbf{x}') = \exp(-||\mathbf{x}-\mathbf{x}'||^2/(2\sigma^2))$, where $\sigma \in \mathbb{R}$ is a hyperparameter.

With this kernel trick, the total training complexity becomes cubic in the number of data points but independent of the number of features, allowing us to compute kernels corresponding to potentially infinite features without paying a price. We refer to the original BLR as working in the primal space, while Equation 2.17 - 2.18 as working in the dual space. As we will show next, this method gives results identical to GPs, which originates from a different model interpretation than BLRs.

2.1.3 Function Space View

A GP can be informally viewed as an infinite dimensional Gaussian distribution over functions $f(\cdot)$. With a finite set of points of interest $\mathbf{X}$ on the support, a GP boils down to a multi-dimensional Gaussian distribution $f(\mathbf{X})$, providing mean and variance estimates at these places. A noisy GP model is similar to BLR, but assumes no parameterization of the true underlying function:

$$y = f(\mathbf{x}) + \epsilon \text{ ,where } \epsilon \sim N(0,\sigma^2) \tag{2.19}$$

Instead of assuming a prior over the weights, GP assumes a Gaussian **prior** over the function itself, whose mean is usually zero and covariance is some positive semi-definite function (thus also a kernel function).

$$\Pr(f(\cdot)) = N(\mathbf{0}, k(\cdot,\cdot)) \tag{2.20}$$

Similarly to BLR, the **likelihood**, **posterior**, and **prediction** can be computed as:

$$\begin{aligned}
\Pr(\mathbf{y}|\mathbf{X}, f(\cdot)) &= N(f(\mathbf{X}), \sigma^2\mathbf{I}) && (2.21)\\
\Pr(f(\cdot)|\mathbf{X}, \mathbf{y}) &\propto \Pr(f(\cdot))\Pr(\mathbf{y}|\mathbf{X}, f(\cdot)) && (2.22)\\
&= N(\bar{m}(\cdot), k'(\cdot,\cdot)) && (2.23)\\
\Pr(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= N(\bar{m}(\mathbf{x}_*), \sigma^2 + k'(\mathbf{x}_*, \mathbf{x}_*)) && (2.24)
\end{aligned}$$

where $\bar{m}(\cdot) = k(\cdot, \mathbf{X})(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$ and $k'(\cdot,\cdot) = k(\cdot,\cdot) - k(\cdot, \mathbf{X})(\mathbf{K} + \sigma^2\mathbf{I})^{-1}k(\mathbf{X}, \cdot)$.

As we have established through Equation 2.9 - 2.18, a GP always has an equivalent BLR (but the BLR may not be feasible due to its complexity). The main difference lies in the kernel function. Popular kernels tend to contain infinite features, providing significant model capacity compared to BLR. Hyperparameters of $k(\cdot,\cdot)$ can be learnt similarly by maximizing the log marginal likelihood:

$$\begin{aligned}
\log \Pr_{\text{GP}}(\mathbf{y}|\mathbf{X}) &\propto \log \int \Pr(\mathbf{y}|f(\cdot), \mathbf{X}) \Pr(f(\cdot)) df && (2.25)\\
&= -\mathbf{y}^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y} - \log|\mathbf{K} + \sigma^2\mathbf{I}| && (2.26)
\end{aligned}$$

For further details, check Chapter 2 of [57].

2.1.4 Random Features

As mentioned above, the complexity of GP is cubic in the amount of data due to the inversion of $\mathbf{K} + \sigma^2\mathbf{I}$. Thus in practice, full GPs are often intractable when the size of dataset is large, and approximations are needed for scalability. Random features [35] is a kind of approximation that allows working in the primal space despite the full GP having infinitely many features. The idea is to find randomized basis functions $\mathbf{z}$ such that:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top\phi(\mathbf{x}') = \mathbb{E}[\mathbf{z}(\mathbf{x})^\top\mathbf{z}(\mathbf{x}')] \approx \left(\frac{\mathbf{s}^m(\mathbf{x})}{\sqrt{m}}\right)^\top \left(\frac{\mathbf{s}^m(\mathbf{x}')}{\sqrt{m}}\right) \tag{2.27}$$

When $\frac{\mathbf{s}^m(\mathbf{x})}{\sqrt{m}}$, the normalized concatenation of m samples of $\mathbf{z}(\mathbf{x})$, has much lower dimensionality than $\phi(\mathbf{x})$ and the amount of data, the cubic cost in the number of features becomes negligible.

The most renowned random feature approach is random Fourier features (RFF) [35] that can approximate any stationary kernel, based on Bochner's theorem:

Theorem 2 (Bochner [37])**.** *A continuous kernel $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') = k(\delta)$ on $\mathbb{R}^p$ is positive definite if and only if $k(\delta)$ is the Fourier transform of a non-negative measure.*

If the kernel is real-valued and properly scaled, its inverse Fourier transform $p(\omega)$ is also real-valued and is a proper probability distribution. Then a valid mapping is $\mathbf{z}_\omega(\mathbf{x}) = [\cos(\omega^\top \mathbf{x}), \sin(\omega^\top \mathbf{x})]^\top$, since

$$k(\delta) = \int_{\mathbb{R}^p} p(\omega) \cos(\omega^\top \delta) d\omega = \mathbb{E}_\omega[\cos(\omega^\top \delta)] = \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] \tag{2.28}$$

The true expectation is then approximated by the empirical mean of multiple samples from $\mathbf{z}_\omega(\mathbf{x})$, which makes it possible to recover an infinite kernel with a finite set of features, and enables working directly in the primal space.

2.2 Federated Learning

2.2.1 Framework Setting

In Federated Learning (FL), there is always a collection of clients $c \in S$ that wish to collaborate, and each client holds their own data $\mathcal{D}_c = \{\mathbf{X}_c, \mathbf{y}_c\}$ locally. We seek to train some machine learning model $\mathcal{M}_\theta$ with parameters θ on these client data. Conventionally, we would centralize all the data $\mathcal{D} = \cup_{c \in S} \mathcal{D}_c$, and choose some algorithm $\mathcal{A}$ and parameter initialization θ_0 to learn the model:

$$\mathcal{M}_\theta = \mathcal{A}(\mathcal{D}, \theta_0) \tag{2.29}$$

This simple approach becomes infeasible if the clients cannot directly share their data due to privacy concerns. FL is designed to specifically tackle this problem. In typical scenarios, a trusted central server is allowed to receive and send perturbed matrices that only contain limited information about raw client data such as model parameters, but any message that could easily leak the original data is still prohibited. Most FL algorithms

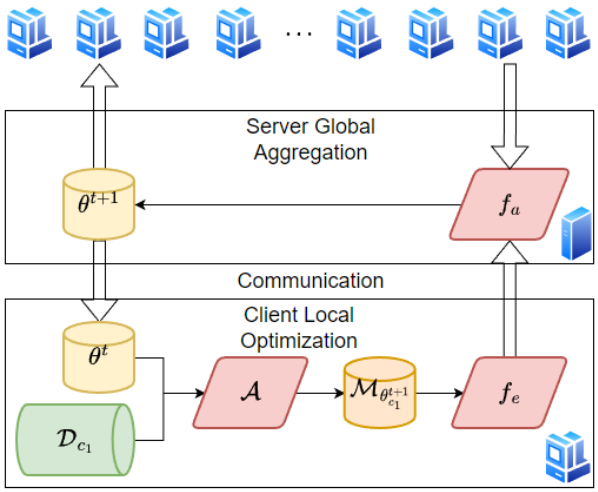

Figure 2.1: Diagram of regular iterative Federated Learning. Clients train their models locally, and send information about the updated model to the server. The server aggregates the information into global consensus parameters, sends them back to clients for a new iteration.

iterate between local optimization and global aggregation, as shown in Figure 2.1. During local optimization at t^{th} iteration, all the clients (or a proportion of the clients in some cases) start with the global consensus parameters θ^t, and run the algorithm $\mathcal{A}$ to update the local models:

$$\mathcal{M}_{\theta_c^{t+1}} = \mathcal{A}(\mathcal{D}, \theta^t) \tag{2.30}$$

Then, clients extract information from their updated model, and send this information instead of data to the central server so that it can be aggregated into the global consensus parameters for next iteration θ^{t+1}. We denote the function for information extraction f_e and for aggregation f_a. Then this process can be formulated as:

$$\theta^{t+1} = f_a(f_e(\mathcal{M}_{\theta_{c_1}^{t+1}}), \cdots, f_e(\mathcal{M}_{\theta_{c_k}^{t+1}})),\ c_1, \cdots, c_k \in S \tag{2.31}$$

Intuitively, θ^{t+1} should serve as a better approximation to the parameters θ in Equation 2.29 than θ^t, so does the corresponding global model $\mathcal{M}_{\theta^{t+1}}$ to $\mathcal{M}_\theta$ (if such a global model is available). The very first FL algorithm, FedAvg [28], chooses the following instances and

illustrates that this framework is effective:

$$\mathcal{A}(\mathcal{D}_c, \theta) : \text{finite iterations of gradient updates} \tag{2.32}$$

$$f_e(\mathcal{M}_{\theta_c}) = \theta_c \tag{2.33}$$

$$f_a(\theta_{c_1}, \cdots, \theta_{c_k}) = \sum_c \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\theta_c \tag{2.34}$$

In words, FedAvg sends all the model parameters to the server, and assigns their average to the new global parameters. The choice of f_a is then widely used and studied in the FL literature. We will refer to it as the FedAvg heuristic later.

2.2.2 Challenges

FL faces a few unique challenges. First, since practically the server has no control over how $\mathcal{D}$ is distributed to all the clients, it is very likely that we have non-identically-and-independently-distributed (non-i.i.d.) clients. That is, if we assume all the client data are sampled from their corresponding latent distribution $\mathcal{D}_c \sim \Pr_c(\mathbf{x}, y)$:

$$\exists c_i, c_j \in S, \Pr_{c_i}(\mathbf{x}, y) \neq \Pr_{c_j}(\mathbf{x}, y) \tag{2.35}$$

To be more specific, mainly two types of non-i.i.d. data are considered [66]:

1. *Label distribution skew*: $\Pr_c(y)$ are different, but $\Pr_c(\mathbf{x}|y)$ may be identical.
2. *Label preference skew*: $\Pr_c(y|\mathbf{x})$ are different, but $\Pr_c(\mathbf{x})$ may be identical.

Most works [18, 42, 29] dealing with non-i.i.d. clients focus on the label distribution skew case, though these two cases may occur at the same time. When the first case arises, using the FedAvg heuristic can cause the global consensus parameters to converge slower or even diverge from the optimum, and it is in general non-trivial to find a good aggregation function f_a. When the second case arises, any uni-model $\mathcal{M}$, which assumes a single true underlying function $f(\cdot)$, cannot capture different label preferences of clients. Methods developed to deal with this case include personalization [25, 21], clustering [11, 19], and adding contextual features [20]. We only deal with the label distribution skew in this thesis, since one GP is a uni-model. More information about non-i.i.d. data can be found in this survey [66].

Besides non-i.i.d. clients, FL also faces the challenge that it is not always safe to build a global model $\mathcal{M}_\theta$ with only the global consensus parameters θ available, due to privacy concerns. Note here the algorithm $\mathcal{A}$ cannot be applied since the server has no access to $\mathcal{D}$. This thesis specifically tackles this challenge. For example, if we assume a stand-alone global GP model and use the FedAvg heuristic, we can have a personalized FL algorithm, like pFedGP [1]:

$$\theta: \text{ hyperparameters of GP kernel} \tag{2.36}$$
$$\mathcal{M}_{\theta_c} = \Pr(y_*|\mathbf{x}_*, \mathcal{D}_c) \text{ by Equation 2.24} \tag{2.37}$$
$$\mathcal{M}_\theta = \Pr(y_*|\mathbf{x}_*, \mathcal{D}) \text{ by Equation 2.24} \tag{2.38}$$
$$\mathcal{A}(\mathcal{D}_c, \theta): \text{ Local updates by Equation 2.26} \tag{2.39}$$
$$f_e(\mathcal{M}_{\theta_c}) = \theta_c \tag{2.40}$$
$$f_a(\theta_{c_1}, \cdots, \theta_{c_k}) = \sum_c \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\theta_c \tag{2.41}$$

This is a valid FL algorithm in the sense that clients collaborate to learn the kernel hyperparamters, and only limited information about the raw data are sent to the server. However, building the global model $\mathcal{M}_\theta$ requires the kernel distance between data points, making it inevitably violating the privacy of clients. This statement applies to any GP approximation that works in the dual space such as the inducing points approximation [34].

On the other hand, FedLoc [64] that assumes an ensemble global model, the distributed product-of-GP-experts model [8], and applies the Proximal ADMM (pxADMM) [63] algorithm from the field of Distributed Learning, can be specified as:

$$\theta: \text{ hyperparameters of GP kernel and pxADMM} \tag{2.42}$$
$$\mathcal{M}_{\theta_c} = \Pr(y_*|\mathbf{x}_*, \mathcal{D}_c) \text{ by Equation 2.24} \tag{2.43}$$
$$\mathcal{M}_\theta = \Pr(y_*|\mathbf{x}_*, \mathcal{D}) = \Pi_c(\mathcal{M}_{\theta_c})^{\beta_c} \tag{2.44}$$
$$\mathcal{A}(\mathcal{D}_c, \theta): \text{ Local updates by pxADMM} \tag{2.45}$$
$$f_e(\mathcal{M}_{\theta_c}) = \theta_c \tag{2.46}$$
$$f_a(\theta_{c_1}, \cdots, \theta_{c_k}): \text{ Global consensus by pxADMM} \tag{2.47}$$

β_c is a hyperparameter used to balance the local models in generalized product-of-experts [5]. In order to obtain the global model $\mathcal{M}_\theta$, the server must have access to all the client models $\mathcal{M}_{\theta_c}$, due to Equation 2.44. To preserve privacy, we should not collect these models to any entity, since it requires access to the client data to calculate the kernel distance by Equation 2.24. We should not send data points to clients for their local predictions either,

since these data points can also be confidential (e.g. any of the clients seeking prediction with the global model). Thus it is not safe to learn a global GP with FedLoc either.

2.3 Summary of Related Work

Deep kernel learning and GP approximations. There have been many works that committed to increase the model capacity of GPs by incorporating deep neural networks (DNNs). [16, 4] either pretrains a deep belief network or directly trains it with a GP to extract first-step features before sending the data into the GP with conventional kernels. [61, 59], building on top of [58, 60, 48, 15, 30], extend this idea with approximations for scalability and stochastic variational inference for classification tasks. Then [49] studies the variance estimations of deep kernel learning models and propose to use Monte-Carlo Dropout [9] for better calibration, and [31] proposes to use Bayesian Neural Networks instead of deterministic DNNs to prevent over-fitting. Besides, [10] designed a special architecture that makes it possible for DNNs to simulate GP behaviors. [7] forms GPs into a deep architecture that corresponds to a deep belief network based on GP mappings. To make GPs practical, one popular method is the inducing point approximation, where the joint GP prior of training points and inducing points are approximated [34]. Variants includes SoD [34], SoR [44], DTC [40], FITC [45], and PITC [39]. Later, KISS-GP [58] gave another interpretation that inducing point approximations are equivalent to global GP interpolation, and it can exploit Kronecker structures [38]. However, these methods work in the dual space and will pose privacy risks under the FL framework. Another approximation method is random features [35, 43, 33] that use randomized basis functions to approximate kernels. More information about scalable GPs can be found in this survey [26].

Distributed and Federated Gaussian Processes (GPs). Closely related to our work is the literature on distributed and federated GPs. Distributed GPs [8, 64] were initially proposed to improve scalability by partitioning the data into several machines since GPs that operate in the dual space scale cubically with the amount of data in the worst case. The product-of-experts framework has emerged as a popular technique to aggregate local GPs, including generalized product of experts [5] and robust Bayesian committee machines [8]. Distributed optimization of hyperparameters in GPs has also been explored [63]. While those techniques do not ensure data privacy, recent work about federated GPs reduce privacy risks while training in a distributed way. This includes pFedGP [1], which optimizes the hyperparameters of a global deep kernel, while training local GPs. In another line of work, GPs have also been used to estimate correlations between clients in FL in order to

actively select independent clients for aggregation [46]. Our work differs from previous distributed and federated GPs by learning a global GP while reducing privacy risks.

Bayesian FL. Beyond federated GPs, other Bayesian models have been explored to represent distributions over models and predictions in FL. The challenge is in the aggregation of the local posteriors. Various techniques have been proposed including posterior averaging [2], online Laplace approximation [27], Thompson sampling [6], MCMC [53].

Chapter 3

FedBNR

3.1 Algorithm

We now describe our approach. First we extend RFF to non-stationary kernels. Then we show how we can learn a stand-alone global GP in a principled way by updating the model in two phases.

3.1.1 Unifying Random Kernel

Although conventional stationary kernels have been specifically popular due to their distance awareness property, deep kernel learning [61, 59] pointed out that incorporating DNNs with stationary kernels further increases the model capacity and makes it more suitable for modern machine learning tasks. However, the common architecture that a DNN is plugged in before the kernel to extract first-step features usually results in non-stationary kernels and also constrains the architecture of the combined kernel. Thus, we wish to extend RFF to non-stationary kernels and provide a unifying definition for random kernels with DNNs, with which people can design any architecture freely.

Let $\omega \in \mathbb{R}^{d'}$ be any random variable or vector, and $g : \mathbb{R}^{d'} \times \mathbb{R}^{p \times \bar{a}} \to \mathbb{R}^{d \times \bar{a}}$ be any function that extracts d features out of each input with some random weights ω. Then we construct the random basis functions $\mathbf{z}$ as $\mathbf{z}_\omega(\mathbf{x}) = g(\omega, \mathbf{x})$. We define the true underlying

kernel and its approximation, the unifying random kernel (URK) as:

$$k(\mathbf{x},\mathbf{x}') = \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] = \mathrm{tr}(\mathrm{Cov}_\omega(\mathbf{z}_\omega(\mathbf{x}),\mathbf{z}_\omega(\mathbf{x}'))) + \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})]^\top \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x}')] \tag{3.1}$$

$$\approx \mathrm{URK}_{\omega,g}(\mathbf{x},\mathbf{x}') = \left(\frac{\mathbf{s}_\omega^m(\mathbf{x})}{\sqrt{m-1}}\right)^\top \left(\frac{\mathbf{s}_\omega^m(\mathbf{x}')}{\sqrt{m-1}}\right), \tag{3.2}$$

where $\mathbf{s}_\omega^m(\mathbf{x})$ is the concatenation of m samples of $\mathbf{z}_\omega(\mathbf{x})$.

Kernel→URK URK can recover any stationary kernel since RFF is a special case of it:

Theorem 3. *Given any properly scaled stationary kernel $k(\mathbf{x},\mathbf{x}')$ on $\mathbb{R}^p$ and its inverse Fourier transform $p(\omega)$, $\exists \omega \sim p(\omega), g(\omega,\mathbf{x}) = [\cos(\omega^\top \mathbf{x}), \sin(\omega^\top \mathbf{x})]^\top$ s.t. $\lim_{m\to\infty} URK_{\omega,g}(\mathbf{x},\mathbf{x}') = k(\mathbf{x},\mathbf{x}')$.*

Proof. Following Theorem 2 and Equation 2.28, the construction of ω and $\mathbf{z}_\omega$ is equivalent to RFF, so $k(\mathbf{x},\mathbf{x}') = \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')]$. By Equation 3.2, $\mathrm{URK}_{\omega,g}(\mathbf{x},\mathbf{x}')$ is an unbiased estimator of $\mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')]$, so $\lim_{m\to\infty} \mathrm{URK}_{\omega,g}(\mathbf{x},\mathbf{x}') = k(\mathbf{x},\mathbf{x}')$. □

To provide some insights into possible non-stationary kernels expressed by URK, we give two example constructions of non-stationary kernels below.

Theorem 4. *Let $\omega \sim N(\mathbf{0},\mathbf{I}), g(\omega,\mathbf{x}) = \exp(\omega^\top \mathbf{x})$, then $k(\mathbf{x},\mathbf{x}') = \lim_{m\to\infty} URK_{\omega,g}(\mathbf{x},\mathbf{x}')$ has infinite features.*

Proof.

$$\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] \\
&= \mathbb{E}_\omega[\exp\left(\omega^\top(\log\bar{\mathbf{x}} + \log\bar{\mathbf{x}}')\right)] \\
&= M_\omega(\log\bar{\mathbf{x}} + \log\bar{\mathbf{x}}'), \text{ the moment generating function of } \omega \\
&= \exp((\mathbf{x}+\mathbf{x}')^\top(\mathbf{x}+\mathbf{x}')/2) \\
&= \exp((\mathbf{x}^\top\mathbf{x} + \mathbf{x}'^\top\mathbf{x}')/2)\exp(\mathbf{x}^\top\mathbf{x}') \\
&= \exp((\mathbf{x}^\top\mathbf{x} + \mathbf{x}'^\top\mathbf{x}')/2)\sum_{l=0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{x}')^l}{l!}, \text{ by the Maclaurin series of } \exp(x) \\
&= \sum_{l=0}^{\infty}\left(\frac{\exp(\mathbf{x}^\top\mathbf{x}/2)}{\sqrt{l!}}(\mathbf{x}^\top\mathbf{x}')^l\frac{\exp(\mathbf{x}'^\top\mathbf{x}'/2)}{\sqrt{l!}}\right) \\
&= \phi(\mathbf{x})^\top\phi(\mathbf{x}')
\end{aligned}$$

□

Additionally, we show URK can recover the popular polynomial kernel, a non-stationary kernel beyond RFF's capability.

Theorem 5. *Let* $c_{poly}, n_{poly} \in \mathbb{R}$. *Define* $\mathbf{p}_{poly} = [\frac{1}{2}, \frac{1}{2p}, \frac{1}{2p}, \cdots, \frac{1}{2p}]^\top \in \mathbb{R}^{p+1}$, $\omega \sim Multi(n_{poly}, \mathbf{p}_{poly})$, *the multinomial distribution,* $\bar{\mathbf{x}} = [\sqrt{2c_{poly}}, \sqrt{2p}\mathbf{x}^\top]^\top \in \mathbb{R}^{p+1}$, $g(\omega, \mathbf{x}) = \exp(\omega^\top \log \bar{\mathbf{x}})$, *then* $k(\mathbf{x}, \mathbf{x}') = \lim_{m\to\infty} URK_{\omega,g}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c_{poly})^{n_{poly}}$

Proof. In the following proof, any subscript i means the i_{th} entry of the vector. By definition,

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] \\
&= \mathbb{E}_\omega[\exp\left(\omega^\top(\log \bar{\mathbf{x}} + \log \bar{\mathbf{x}}')\right)] \\
&= M_\omega(\log \bar{\mathbf{x}} + \log \bar{\mathbf{x}}'), \text{ the moment generating function of } \omega \\
&= \left(\sum_{i=1}^{p+1} \mathbf{p}_{poly,i} \exp(\log \bar{\mathbf{x}}_i + \log \bar{\mathbf{x}}'_i)\right)^{n_{poly}} \\
&= \left(\frac{1}{2}\exp(2\log\sqrt{2c_{poly}}) + \sum_{i=1}^{p} \frac{1}{2p} \exp(\log\sqrt{2p}\mathbf{x}_i + \sqrt{2p}\log \mathbf{x}'_i)\right)^{n_{poly}} \\
&= \left(c_{poly} + \sum_{i=1}^{p} \mathbf{x}_i \mathbf{x}'_i\right)^{n_{poly}} \\
&= (\mathbf{x}^\top \mathbf{x}' + c_{poly})^{n_{poly}}
\end{aligned}
$$

□

URK→Kernel More importantly, note that in the definition of URK we do not rely on the inverse Fourier transformation or the Bochner's theorem to find a valid distribution for ω. Instead, any $p(\omega)$ and g can give us a valid kernel:

Theorem 6. *Given any proper probability distribution* $p(\omega)$ *on* $\mathbb{R}^{d'}$ *and function* g *on* $\mathbb{R}^{d'} \times \mathbb{R}^{p\times\bar{a}} \to \mathbb{R}^{d\times\bar{a}}$, *the corresponding kernel matrix* $k(\mathbf{X}, \mathbf{X}') = \lim_{m\to\infty} URK_{\omega,g}(\mathbf{X}, \mathbf{X}')$ *is positive semi-definite.*

Proof. Following Equation 3.1, we have

$$k(\mathbf{X}, \mathbf{X}') = \textstyle\sum_{i=1}^{d}(\mathrm{Cov}_\omega(\mathbf{z}_\omega(\mathbf{X})_i, \mathbf{z}_\omega(\mathbf{X}')_i)) + \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{X})]^\top \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{X}')].$$

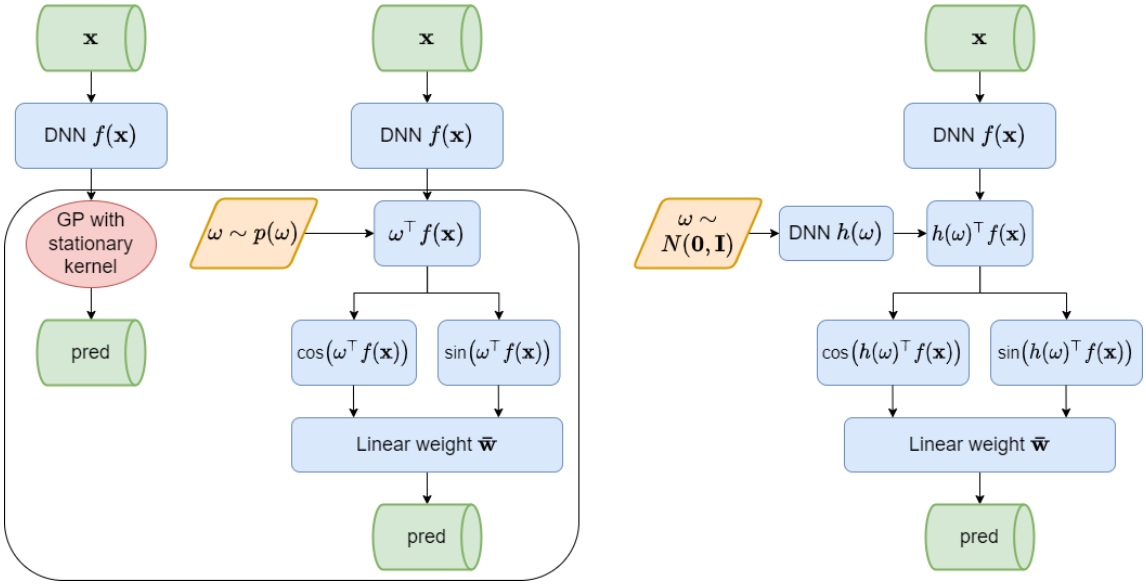

Figure 3.1: From left to right: a standard deep kernel learning algorithm with conventional stationary kernel GPs; the corresponding URK architecture; a more general architecture enabled by URK for convenient latent stationary kernel learning.

$\mathbf{z}_\omega(\mathbf{X})_i$ denotes the i^{th} row of the $d \times n$ matrix $\mathbf{z}_\omega(\mathbf{X})$. The first term is the addition of d covariance matrices and is always positive definite; the second term is symmetrical and essentially adds one feature to the first term. Thus $k(\mathbf{X}, \mathbf{X}')$ is always positive semi-definite. □

Since g is an arbitrary function, we can assign it any DNN with any architecture. We call it a Kernel Neural Network (KNN) since its weights, θ, are essentially the kernel hyperparameters. Theorem 6 allows us to train kernels with optimization methods similarly to training DNNs from a much richer hypothesis set than conventional kernels.

3.1.2 Greater Expressiveness with URK

We expand on new architectures of deep random kernels enabled by the definition of URK in this section to show that URK is more flexible than common heuristics in deep kernel learning. Minimal arguments and evidence are provided below since the ultimate goal of this thesis is still to propose a Bayesian FL algorithm that can learn a global GP, not a random feature algorithm that provides better GP approximation.

As illustrated in Figure 3.1, URK can recover any standard deep kernel combined with conventional stationary GPs easily. Further more, we can exploit the flexibility of URK and define a distribution shifter h that transforms ω. We can start from a standard normal distribution, which is very easy to sample from, and send the samples through h to simulate a much more complex distribution with minimal computation resources required. If we choose h carefully so that the identity function is in its hypothesis set, we will presumably learn a kernel at least as good as the Gaussian kernel.

If we take a step further beyond DNNs, the function g in URK can be assigned some replication policy that creates randomized versions of $\mathbf{x}$ given different ω such as multiplying or adding random Gaussian noise to the input. Combined with the idea of being distance-aware in some latent space, we present another architecture as the leftmost diagram in Figure 3.2. We train a GP with URK of these two architectures on a step function and show their predictions in Figure 3.2, where f is a very small DNN. The green curve shows the predictions. The light red area is a 95% confidence interval based on variance. The blue points are the training data. The replicate policy (the upper right one), although introduces no additional parameters, further increases the model capacity, and its prediction is more reasonable than just using a DNN and a stationary kernel.

3.1.3 Two-phase Update

The training procedure of FedBNR can be divided into 2 phases, as illustrated in Fig. 3.3. In the first phase, we train the KNN by optimization methods. In the second phase, we calculate the weights of the last linear layer that maps the random features to the output space, by a closed-form formula inferred from Equation 2.1 - 2.6. The pseudocode of FedBNR is summarized in Algorithm 1.

In phase 1, we follow a standard training procedure under the FL framework. We assume there is a central server holding the shared global KNN weights θ and global hyperparameters σ, λ that denote the noise level and the prior variance respectively. We assume there is a set of clients $c \in S$ holding local KNN weights θ_c, local hyperparameters σ_c, λ_c, local inputs $\mathbf{X}_c$, and local targets $\mathbf{y}_c$. We use $\theta(\mathbf{x})$ to denote the result of sending $\mathbf{x}$ through the KNN. In the beginning of each aggregation round, the server first sends a copy of aggregated or initialized θ, σ, λ to all the clients. Then all the clients $c \in S$ first update their local model for a fixed number of iterations. Then they send θ_c, σ_c, and λ_c back to the server for aggregation and start another aggregation round. The local loss function is

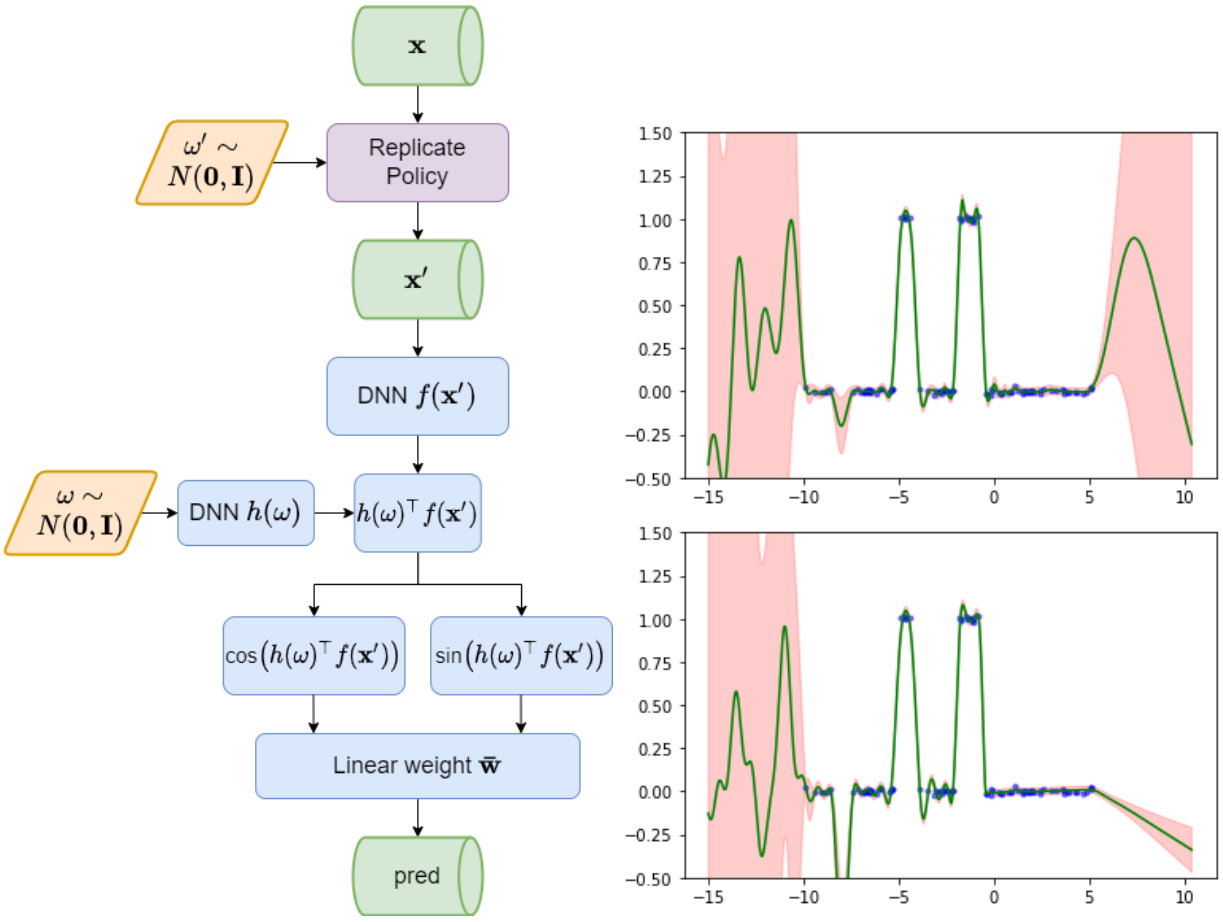

Figure 3.2: Left: another URK architecture. Right: Train a GP with URK on a step function. Top right: results of the leftmost architecture in Figure 3.2. Bottom right: results of the rightmost architecture in Figure 3.1. Blue points: training data; green curve: point estimation; red range: 95% confidence interval.

Algorithm 1 Federated Bayesian Neural Regression

(θ: the global KNN, σ: noise level, λ: prior covariance of the linear layer, ω: a set of random numbers, ζ: local learning step, ζ': knowledge distillation step)

Phase 1: Kernel Learning

Initialize shared kernel $\theta \leftarrow \theta^0$, hyperparameters $\sigma \leftarrow \sigma^0, \lambda \leftarrow \lambda^0$

for each aggregation round $t \leftarrow 0, 1, 2, \cdots$ **do**

  for each client $c \in S$ **do**

    $\theta_c^t, \sigma_c^t, \lambda_c^t \leftarrow \theta^t, \sigma^t, \lambda^t$

    for each local update round $k \leftarrow 0, 1, 2, \cdots$ **do**

      $\theta_c^t, \sigma_c^t, \lambda_c^t \leftarrow -\zeta \nabla \mathcal{L}_c^{ML}$ according to Equation 3.3

  if FedAvg **then**

    $\theta^{t+1}, \sigma^{t+1}, \lambda^{t+1} \leftarrow mean(\theta_S^t), mean(\lambda_S^t), mean(\lambda_S^t)$

  else if Knowledge Distillation **then**

    for each knowledge distillation round $k' \leftarrow 0, 1, 2, \cdots$ **do**

      $\theta^{t+1}, \sigma^{t+1}, \lambda^{t+1} \leftarrow -\zeta' \nabla \mathcal{L}^{KD}$ according to Equation 3.4

Phase 2: Update the Global Linear Layer

Server: send $\theta, \sigma, \lambda, \omega$ to all clients

for each **Client** $c \in S$ **do**

  compute the random features $\mathbf{\Phi}_c \leftarrow \theta(\omega, \mathbf{X}_c)$ and send $\mathbf{\Phi}_c \mathbf{\Phi}_c^\top$ to the server

Server: send $\mathbf{A}^{-1} \leftarrow (\sigma^{-2} \sum^c (\mathbf{\Phi}_c \mathbf{\Phi}_c^\top) + \lambda^{-2} \mathbf{I})^{-1}$ to all clients

for each **Client** $c \in S$ **do**

  $\bar{\mathbf{w}}_c \leftarrow \sigma^{-2} \mathbf{A}^{-1} \mathbf{\Phi}_c \mathbf{y}_c$ and send $\bar{\mathbf{w}}_c$ to the server

Server: $\bar{\mathbf{w}} \leftarrow \sum_{c \in S} \bar{\mathbf{w}}_c$

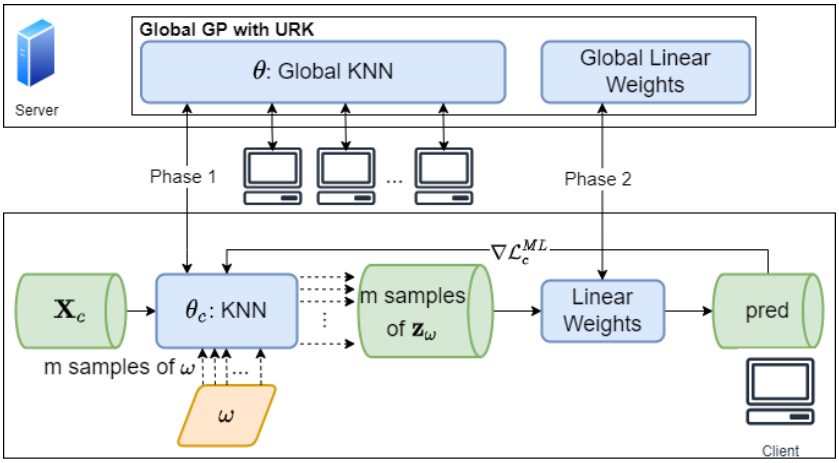

Figure 3.3: FedBNR learns a global federated GP in two phases: kernel learning with FL optimization and last layer updating with exact Bayesian inference. Though we work in the primal space, URK allows us to approximate an infinite kernel in the dual space with finite features in the primal space.

the local log marginal likelihood:

$$\mathcal{L}_c^{ML} = \log \Pr_{\text{BLR}}(\mathbf{y}_c|\theta_c(\mathbf{X}_c); \sigma_c, \lambda_c) = -n_c \log \sigma_c^2 - \log(|\lambda_c^2 \mathbf{I}||\mathbf{A}_c|) - \mathbf{y}_c^\top \mathbf{y}_c/\sigma_c^2 + \mathbf{w}_c^\top \mathbf{A}_c \mathbf{w}_c \tag{3.3}$$

One commonly used method for aggregation is the FedAvg [28] heuristic, where the new global model parameters are assigned the average of all client model parameters $\theta = \sum_{c\in S} \theta_c/|S|$. However, multiple works [18, 42, 29] have pointed out the quality and the convergence rate of this heuristic can suffer from non-i.i.d. clients. To account for this, we propose to adapt kernel knowledge distillation [12] to aggregate the KNNs. We assume the server holds a relatively small dataset $\mathbf{X}_{kd}, \mathbf{y}_{kd}$ and tries to minimize the following knowledge distillation loss with respect to this dataset:

$$\mathcal{L}^{KD} = \mathcal{L}_{kd}^{ML} + \alpha * \text{MSE}\left(\theta(\mathbf{X}_{kd})^\top \theta(\mathbf{X}_{kd}) - \sum_{c\in S} \theta_c(\mathbf{X}_{kd})^\top \theta_c(\mathbf{X}_{kd})/|S|\right) \tag{3.4}$$

Here, $\mathcal{L}_{kd}^{ML}$ is the global log marginal likelihood loss $\mathcal{L}^{ML}$ with respect to the knowledge distillation dataset $\mathbf{X}_{kd}, \mathbf{y}_{kd}$. α is a common hyperparameter in knowledge distillation methods to adjust the ratio between the log marginal likelihood loss and the mean squared error (MSE) loss. The MSE loss factor forces the global kernel $\theta(\mathbf{X}_{kd})^\top \theta(\mathbf{X}_{kd})$ to simulate the mean of all client kernels $\sum_{c\in S} \theta_c(\mathbf{X}_{kd})^\top \theta_c(\mathbf{X}_{kd})/|S|$, which is akin to concatenating all

the features of the client kernels. Ideally, if the global kernel successfully learns to do so, it should not perform worse on any of the clients, while the FedAvg heuristic has no similar guarantees.

In phase 2, we fix the kernel hyperparameters and learn $\mathbf{A}^{-1}$, the matrix for covariance prediction, and $\bar{\mathbf{w}}$, the weights of the last linear layer, in an exact way as if all client data are centralized. To understand the procedure, first notice that we can decompose $\mathbf{A}$ and $\bar{\mathbf{w}}$ as follows:

$$\mathbf{A} = \sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda^{-2}\mathbf{I} = \sigma^{-2}\sum_{c\in S}\mathbf{\Phi}_c\mathbf{\Phi}_c^\top + \lambda^{-2}\mathbf{I} \tag{3.5}$$

$$\bar{\mathbf{w}} = \sigma^{-2}\mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y} = \sum_{c\in S}\sigma^{-2}\mathbf{A}^{-1}\mathbf{\Phi}_c\mathbf{y}_c \tag{3.6}$$

Here $\mathbf{\Phi}_c = \theta(\mathbf{X}_c)$ denotes the random features of local inputs extracted by the global KNN. The server first broadcasts the global model to all the clients, and asks them to return the scatter matrices $\mathbf{\Phi}_c\mathbf{\Phi}_c^\top$, and then the server can calculate $\mathbf{A}$ following Equation 3.5. Next, the server broadcasts $\mathbf{A}^{-1}$ and asks clients for the intermediate weights $\sigma^{-2}\mathbf{A}^{-1}\mathbf{\Phi}_c\mathbf{y}_c$. Finally, the server can calculate $\bar{\mathbf{w}}$ according to Equation 3.6 and broadcasts the whole model again to all the clients.

We claim that FedBNR protects privacy of clients at least as well as FedAvg and other federated learning algorithms that send client models to the server. In phase 1, the aggregation only requires client model parameters. In phase 2, we send information twice outside each client: the scatter matrices and the intermediate weights. Sending these matrices and vectors are safer than sending the features $\mathbf{\Phi}_c$ directly since they have limited sizes that are completely independent of the training data size n_c, meaning they must only contain limited information about the raw data. Specifically, the scatter matrices are of size $md \times md$, and the intermediate weights are of size md, where m is the number of samples from $\mathbf{z}$ and d is the output dimension of the KNN. Although there have been critiques that part of the client data can be recovered from just gradients (model parameter changes) in FedAvg, it is possible to further make FedAvg and our model more secured by trivially applying differential privacy to public messages, since these messages are essentially summed at the server to a single term.

3.2 Experiments

3.2.1 Synthetic Experiment

Although personalized federated learning (PFL) is usually viewed as an advanced version of the plain FL framework since it learns a fine-tuned local model for each client and can automatically handle label preference skew [66] (i.e. $\Pr(\mathbf{y}|\mathbf{X})$ varies for clients), it is noteworthy that if all the clients can agree on a single global model in the hypothesis set, PFL with no global model may not be the best choice due to the trade-off between generalization and personalization. For example, a hospital that only collected data for lung cancer may also want their model to help diagnosing COVID-19, but personalization would prevent the model from generalizing to other diseases. Moreover, when a new client comes in with few data points, the quality of prediction will suffer compared to other clients, even only querying its own range.

We designed the following synthetic experiments to support these arguments. We first decide a true underlying function, sample 200 points uniformly from the range $[-5, 5]$, and add Gaussian noise with $\sigma = 0.5$. We then learn the kernel hyperparameters in a centralized fashion to eliminate any impact from imperfect kernels later on. Details about the kernel sizes and architectures can be found in Appendix B. We then assign the first 100 points in range $[-5, 0]$ to client 1, and the rest to client 2. We train pFedGP and FedBNR with the learnt kernel hyperparameters fixed and query for prediction in range $[-5, 5]$. The top left graph of Figure 3.4 shows the result of pFedGP, and the top right one shows the result of FedBNR. The blue curve shows the true underlying function, and the green curve shows the predictions. The light red area is a 95% confidence interval based on variance.

Since both algorithms share the same kernel hyperparameters, the only difference lies in whether they leverage data of both clients for the predictive model. As shown in the pFedGP graph, the personalized model of client 1 only sees its own data (red dots), so it does not generalize well to the range of client 2, while the global model learnt by FedBNR can leverage all the data (blue dots) and generalizes to the full range.

Next, we introduce two new clients into the system. The first client holds data uniformly sampled from range $[-5, 5]$, and the second from range $[5, 15]$. We again fix the kernel hyperparameters learnt centrally and train pFedGP, FedLoc, and FedBNR on these new clients seperated. For testing, we still query the range $[-5, 5]$. The bottom graphs of Figure 3.4 shows the MSE loss as the size of data of both new clients grows. As expected, for pFedGP, the quality of prediction is massively impacted when there are few points in

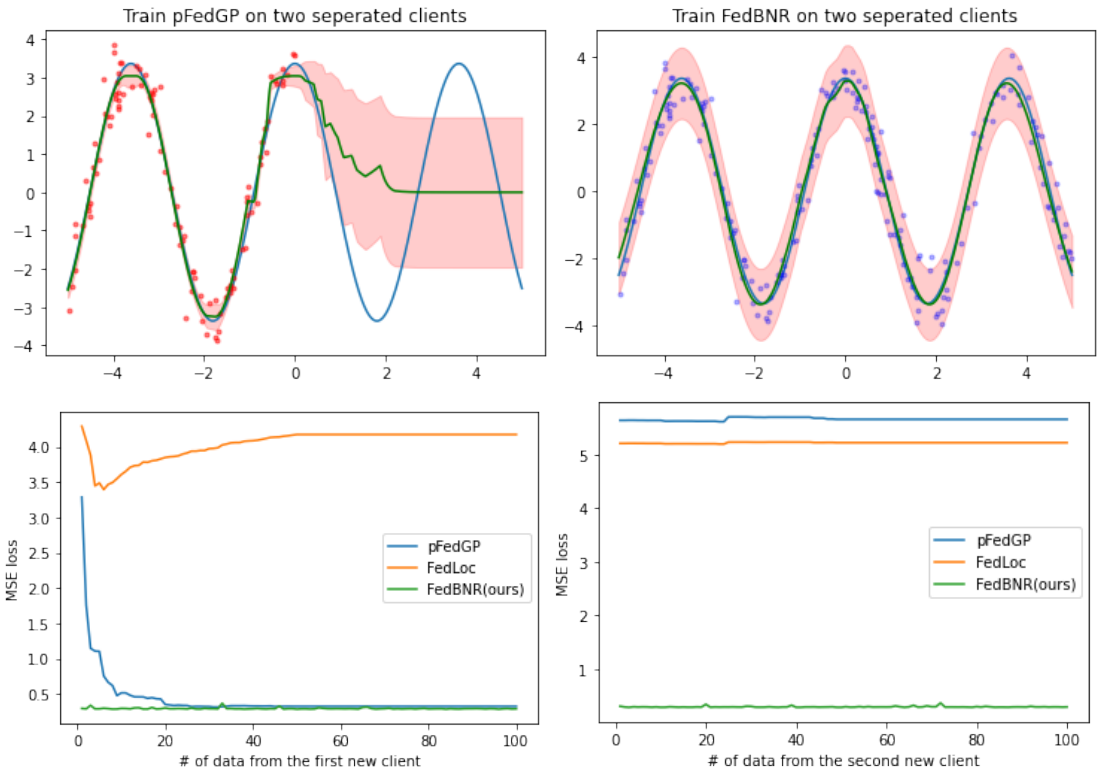

Figure 3.4: Top graphs: (i) left: prediction of a pFedGP model trained on two non-overlapping clients; (ii) right: global prediction of FedBNR. Blue curve: underlying truth; green curve: point estimation; red range: 95% confidence interval; dots: training data leveraged by the model. Bottom graphs: MSE of predictions tested on the same range v.s. number of training data from a new client; (i) left: client with training range [-5, 5]; (ii) right: client with training range [5, 15]

the first case, and for FedBNR the loss remains approximately a straight line. For FedLoc, the loss is also impacted due to zero-out effects of non-overlapping client models. Even worse, when training data and testing data are not in the same range for the second new client, the MSE loss of both pFedGP and FedLoc never drops back to the level of previous clients.

3.2.2 UCI Regression Datasets

We conducted comprehensive experiments on five UCI regression datasets under ten cases. Results of two variants are reported: i) *FedBNR* that performs the FedAvg heuristic at aggregation; and ii) *FedBNR-KD* that performs the knowledge distillation method at aggregation.

Two groups of baselines are compared to our method for RMSE error: i) ablation study that contains local+local, local+global, avg+local, and kd+global, in the format of kernel learning method + last linear weight learning method, where we remove the global aggregation of either phase 1 or phase 2 from our methods; ii) previous works that contains (1) FedAvg [28], a standard non-Bayesian FL algorithm that has a global model; (2) FedProx [22], a non-Bayesian FL algorithm that adds a proximal loss to FedAvg to prevent client models from getting too far from the global model; (3) pFedGP [1], a Bayesian PFL method that learns a local GP with a shared deep kernel for each client; and (4) pFedGP [64], a Bayesian FL method that directly applies distributed GP methods [63] without deep kernel learning. We also compare to pFedGP and FedLoc for calibration errors since they are Bayesian models that have a notion of confidence. Besides, we report results of a centralized GP equipped with URK as a casual reference of the testing error lower-bound.

We used fully connected neural networks to extract first-step features for FedAvg, FedProx, and pFedGP. We used Gaussian kernels for the GPs in pFedGP and FedLoc. For fairness, KNNs used by our methods have similar architectures to the combined kernel of pFedGP. For scalability, FITC [45] approximations are implemented for the other GPs as described in pFedGP. We used 50 random samples for KNN and 50 inducing points for pFedGP and FedLoc. Further details are in Appendix A.

Each dataset is uniformly divided into 8:1:1 training:testing:validation sets globally. The training data is sorted by the feature that has the largest absolute correlation coefficient with the output, and divided into multiple chunks. Each client randomly takes two chunks so that their data distributions are heterogeneous. The larger the absolute correlation coefficient is, the more significant the distribution skew is. Before training, we tune

Table 3.1: UCI regression datasets, RMSE reported. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft [47]		SML [65]		Parkinsons [50]		Bike [52]		CCPP [51]	
train/test size	2670 334		3309 414		4700 587		7008 876		7654 957	
corr-coef	-0.660		0.783		0.410		0.539		-0.948	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.95		0.21		3.58		0.37		4.02	
local+local	$1.26^{\Uparrow}$	$1.48^{\Uparrow}$	$1.49^{\Uparrow}$	$2.32^{\Uparrow}$	$10.9^{\Uparrow}$	$10.6^{\Uparrow}$	$0.79^{\Uparrow}$	$0.92^{\Uparrow}$	$14.6^{\Uparrow}$	$19.3^{\Uparrow}$
local+global	$1.08^{\Uparrow}$	$1.22^{\Uparrow}$	$1.00^{\Uparrow}$	$1.65^{\Uparrow}$	$6.42^{\Uparrow}$	$7.42^{\Uparrow}$	$0.59^{\Uparrow}$	$0.73^{\Uparrow}$	$5.62^{\Uparrow}$	$7.03^{\Uparrow}$
avg+local	$1.05^{\Uparrow}$	$1.06^{\Uparrow}$	$0.61^{\Uparrow}$	$0.81^{\Uparrow}$	$9.84^{\Uparrow}$	$8.80^{\Uparrow}$	$0.45^{\Uparrow}$	$0.54^{\Uparrow}$	$8.32^{\Uparrow}$	$13.4^{\Uparrow}$
kd+local	$1.04^{\Uparrow}$	$1.28^{\Uparrow}$	$0.86^{\Uparrow}$	$1.34^{\Uparrow}$	$6.15^{\Uparrow}$	$6.97^{\Uparrow}$	$0.51^{\Uparrow}$	$0.61^{\Uparrow}$	$10.0^{\Uparrow}$	$17.1^{\Uparrow}$
FedAvg [28]	$1.00^{\uparrow}$	$1.03^{\Uparrow}$	$0.36^{\Uparrow}$	$0.71^{\Uparrow}$	$6.83^{\Uparrow}$	$7.38^{\Uparrow}$	$\mathbf{0.38}^{\downarrow}$	0.42	4.45	4.44
FedProx [22]	0.98	$1.05^{\Uparrow}$	$0.34^{\uparrow}$	$0.62^{\Uparrow}$	$6.32^{\Uparrow}$	$7.38^{\Uparrow}$	0.39	0.42	4.43	4.47
pFedGP [1]	$0.99^{\uparrow}$	$1.15^{\Uparrow}$	$0.75^{\Uparrow}$	$1.34^{\Uparrow}$	$9.36^{\Uparrow}$	$8.91^{\Uparrow}$	$0.45^{\Uparrow}$	$0.46^{\Uparrow}$	$14.2^{\Uparrow}$	$18.2^{\Uparrow}$
FedLoc [64]	$1.08^{\Uparrow}$	$4.15^{\Uparrow}$	$2.83^{\Uparrow}$	$5.43^{\Uparrow}$	$8.40^{\Uparrow}$	$11.5^{\Uparrow}$	$0.64^{\Uparrow}$	$0.75^{\Uparrow}$	$20.6^{\Uparrow}$	$44.1^{\Uparrow}$
ours										
FedBNR	0.98	**0.97**	**0.25**	**0.44**	**3.10**	5.42	0.39	**0.42**	4.40	4.51
FedBNR-KD	$\mathbf{0.96}^{\Downarrow}$	$0.98^{\uparrow}$	$0.55^{\Uparrow}$	$0.55^{\Uparrow}$	$4.58^{\Uparrow}$	$\mathbf{4.67}^{\downarrow}$	$0.43^{\Uparrow}$	$0.48^{\Uparrow}$	**4.38**	$\mathbf{4.38}^{\downarrow}$

hyperparameters that cannot be learnt by gradient descent with grid searching on the validation set. Specially, FedBNR-KD uses 80% of the validation set for knowledge distillation, and the rest 20% for validation. All the datasets are then ran for at most 50 local epochs times 100 aggregation rounds in full batches. Validation and testing error are recorded for each aggregation round. For methods that contain only local models, these errors are defined as the mean error of all local models with respect to the testing/validation set. If the validation error has not improved for 5 rounds, the training process is terminated. We report the average minimum testing RMSE for 10 random seeds for each case in Table 3.1. We also measured the statistical significance of the results compared to FedBNR with one-tailed Wilcoxon signed-rank tests [56]. We then report expected calibration errors in Table 3.2 and perform the Wilcoxon test compared to FedBNR-KD. Additional metrics, including the maximum calibration error, the Brier score, and further details of Table 3.1 are listed below in Section 3.2.3.

The results show: i) in terms of RMSE, our methods are statistically better than the Bayesian models pFedGP and FedLoc in all the cases and better than the non-Bayesian

Table 3.2: Expected calibration error (ECE) reported. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.02		0.09		0.27		0.11		0.24	
pFedGP [1]	$0.43^{\Uparrow}$	$0.38^{\Uparrow}$	$0.45^{\Uparrow}$	$0.45^{\Uparrow}$	$0.49^{\Uparrow}$	$0.48^{\Uparrow}$	$0.42^{\Uparrow}$	$0.41^{\Uparrow}$	$0.49^{\Uparrow}$	$0.49^{\Uparrow}$
FedLoc [64]	$0.32^{\Uparrow}$	$0.44^{\Uparrow}$	$\mathbf{0.12}^{\Downarrow}$	$0.27^{\Uparrow}$	$0.32^{\Uparrow}$	$0.43^{\Uparrow}$	$0.26^{\Uparrow}$	$0.16^{\Uparrow}$	$\mathbf{0.23}^{\Downarrow}$	$0.50^{\Uparrow}$
ours										
FedBNR	**0.05**	$0.20^{\Uparrow}$	$0.39^{\Uparrow}$	$0.37^{\Uparrow}$	$0.36^{\uparrow}$	$0.40^{\Uparrow}$	**0.07**	$0.04^{\Uparrow}$	$0.24^{\Downarrow}$	$\mathbf{0.20}^{\Downarrow}$
FedBNR-KD	**0.05**	**0.06**	0.20	**0.21**	**0.29**	**0.30**	0.08	**0.09**	0.30	0.31

models FedAvg and FedProx in most of the cases, especially when the training set at each client is significant smaller than the whole set; ii) compared with the ablation study methods, our methods always perform better, so the global aggregation at both phases are essential; iii) FedLoc without deep kernel learning has a especially smaller model capacity; iv) FedBNR-KD only outperforms FedBNR in 40% of cases in terms of RMSE, which is probably due to the small size of data (8% of all) used for knowledge distillation. However, FedBNR-KD is clearly more stable when client heterogeneity gets worse as the number of clients increases. It is also better calibrated than FedBNR and other Bayesian models in most cases.

3.2.3 Other Metrics

We include the maximum calibration error (MCE) and the Brier score (BRI) of the UCI experiments in Table 3.3 and 3.4. MCE measures the (estimated) worst difference between $p\%$ confidence intervals and $p'\%$ test points falling into these intervals. BRI measures the mean squared difference between the confidence $p\%$ and observations, where a test point falling in the CI is observed as 1, otherwise 0, and the regression task is binarized into a classification task. We also include the standard error of the mean (SEM) in Table 3.5 and 3.6. Our methods still perform better in most of the cases.

Table 3.3: Maximum calibration error (MCE) reported. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.03		0.16		0.42		0.16		0.40	
pFedGP	$0.77^{\Uparrow}$	$0.65^{\Uparrow}$	$0.82^{\Uparrow}$	$0.83^{\Uparrow}$	$0.93^{\Uparrow}$	$0.91^{\Uparrow}$	$0.78^{\Uparrow}$	$0.73^{\Uparrow}$	$0.94^{\Uparrow}$	$0.93^{\Uparrow}$
FedLoc	$0.51^{\Uparrow}$	$0.81^{\Uparrow}$	$\mathbf{0.20}^{\Downarrow}$	$0.44^{\Uparrow}$	$0.54^{\Uparrow}$	$0.76^{\Uparrow}$	$0.42^{\Uparrow}$	$0.28^{\Uparrow}$	$0.39^{\Downarrow}$	0.95
ours										
FedBNR	**0.09**	$0.32^{\Uparrow}$	$0.67^{\Uparrow}$	$0.62^{\Uparrow}$	$0.62^{\Uparrow}$	$0.72^{\Uparrow}$	**0.15**	$0.64^{\Uparrow}$	$\mathbf{0.39}^{\Downarrow}$	$\mathbf{0.31}^{\Downarrow}$
FedBNR-KD	0.10	**0.11**	0.33	**0.35**	**0.47**	**0.48**	**0.15**	**0.17**	0.49	0.50

Table 3.4: Brier score (BRI) reported. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.16		0.20		0.24		0.18		0.24	
pFedGP	$0.30^{\Uparrow}$	$0.28^{\Uparrow}$	$0.31^{\Uparrow}$	$0.31^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.30^{\Uparrow}$	$0.30^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$
FedLoc	0.21	$0.31^{\Uparrow}$	$\mathbf{0.18}^{\Downarrow}$	$0.25^{\Uparrow}$	$0.26^{\Uparrow}$	$0.30^{\Uparrow}$	$\mathbf{0.19}^{\downarrow}$	$\mathbf{0.16}^{\Downarrow}$	$0.24^{\Downarrow}$	$0.33^{\Uparrow}$
ours										
FedBNR	**0.18**	$0.22^{\Uparrow}$	$0.29^{\Uparrow}$	$0.28^{\Uparrow}$	$0.28^{\Uparrow}$	$0.30^{\Uparrow}$	$0.20^{\Uparrow}$	$0.29^{\Uparrow}$	$\mathbf{0.24}^{\Downarrow}$	$\mathbf{0.22}^{\Downarrow}$
FedBNR-KD	**0.18**	**0.18**	0.23	**0.23**	**0.25**	**0.26**	0.19	0.20	0.25	0.26

Table 3.5: UCI regression datasets, RMSE ± standard error of the mean (SEM) reported. First half.

	Skillcraft		SML		Parkinsons	
#clients	10	100	10	100	10	100
local+local	1.26±0.03	1.48±0.01	1.49±0.06	2.32±0.03	10.9±0.26	10.6±0.05
local+global	1.08±0.02	1.22±0.01	1.00±0.02	1.65±0.01	6.42±0.10	7.42±0.01
avg+local	1.05±0.02	1.06±0.02	0.61±0.06	0.81±0.05	9.84±0.31	8.80±0.11
kd+local	1.04±0.01	1.28±0.02	0.86±0.06	1.34±0.12	6.15±0.30	6.97±0.27
FedAvg	1.00±0.01	1.03±0.01	0.36±0.02	0.71±0.02	6.83±0.16	7.38±0.21
FedProx	0.98±0.01	1.05±0.01	0.34±0.01	0.62±0.02	6.32±0.27	7.38±0.25
pFedGP	0.99±0.01	1.15±0.01	0.75±0.05	1.34±0.04	9.36±0.16	8.91±0.09
FedLoc	1.08±0.01	4.15±0.26	2.83±0.07	5.43±0.04	8.40±0.02	11.5±0.03
ours						
FedBNR	0.98±0.01	**0.97±0.01**	**0.25±0.01**	**0.44±0.02**	**3.10±0.23**	5.42±0.20
FedBNR-KD	**0.96±0.01**	0.98±0.01	0.55±0.01	0.55±0.01	4.58±0.05	**4.67±0.04**

Table 3.6: UCI regression datasets, RMSE ± standard error of the mean (SEM) reported. Second half.

	Bike		CCPP	
#clients	10	100	10	100
local+local	0.79±0.02	0.91±0.01	14.6±0.31	19.3±2.08
local+global	0.59±0.01	0.73±0.01	5.62±0.19	7.03±0.16
avg+local	0.45±0.01	0.54±0.01	8.32±0.43	13.4±0.43
kd+local	0.51±0.01	0.61±0.01	10.0±0.61	17.1±0.63
FedAvg	**0.38±0.01**	0.42±0.01	4.45±0.06	4.44±0.04
FedProx	0.39±0.01	0.42±0.01	4.43±0.04	4.47±0.02
pFedGP	0.45±0.02	0.46±0.01	14.2±0.61	18.2±0.23
FedLoc	0.64±0.01	0.75±0.01	20.6±0.74	44.1±0.61
ours				
FedBNR	0.39±0.01	**0.42±0.01**	4.40±0.01	4.51±0.03
FedBNR-KD	0.43±0.01	0.48±0.01	**4.38±0.01**	**4.38±0.01**

Chapter 4

Conclusion

4.1 Summary

In this work, we proposed FedBNR, a novel Bayesian federated learning algorithm that learns a global federated GP without privacy leakage and introduced URK, a unifying definition for deep random features, to approximate kernels with randomized basis functions in the primal space. FedBNR consists of a non-Bayesian DNN and a Bayesian linear layer, thus both scalable and provides well-calibrated uncertainty estimates; FedBNR learns a randomized kernel represented by a DNN under the URK definition, thus has better model capacity than plain DNNs; FedBNR shares scatter matrices instead of direct features to achieve the exact global optimum of the last layer, thus is accurate and privacy-friendly. We derived two variants based on the FedAvg heuristic and the knowledge distillation. Both variants show empirically statistically significant improvements in terms of point estimation and calibration than other federated GP models when there is no label preference skew.

4.2 Limitations and Future work

- We presented the definition of URK and proved any URK is a valid kernel, but it is not clear whether any kernel has a corresponding URK that gives reasonable approximation. We were able to prove this property for any stationary kernel, and it is worth exploring the non-stationary kernel set.

- We gave a few examples on new architectures enabled by URK without rigorous demonstration of further contribution. It might be possible to extend these architectures to discrete domains such as NLP.
- We considered knowledge distillation for non-i.i.d. data but the empirical results are mixed, especially when the heterogeneity is mild. This is possibly related to the size of the knowledge distillation dataset, and artificially generating adversial samples might solve this problem.
- We focused on the uni-model case without any label preference skew. We could generalize to multi-models by considering mixtures of GPs or client clusters.

References

[1] Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. In *International Conference on Learning Representations*, 2020.

[3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[4] Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345. IEEE, 2016.

[5] Yanshuai Cao and David J Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*, 2014.

[6] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated bayesian optimization via thompson sampling. *Advances in Neural Information Processing Systems*, 33:9687–9699, 2020.

[7] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

[8] Marc Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490. PMLR, 2015.

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[10] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018.

[11] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

[12] Bobby He and Mete Ozay. Feature kernel distillation. In *International Conference on Learning Representations*, 2021.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Harold V Henderson and Shayle R Searle. On deriving the inverse of a sum of matrices. *Siam Review*, 23(1):53–60, 1981.

[15] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[16] Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. *Advances in neural information processing systems*, 20, 2007.

[17] Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. Learning private neural language modeling with attentive aggregation. In *2019 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

[18] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[19] Kavya Kopparapu and Eric Lin. Fedfmc: Sequential efficient federated learning on non-iid data. *arXiv preprint arXiv:2006.10937*, 2020.

[20] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[21] Daliang Li and Junpu Wang. FedMD: Heterogenous federated learning via model distillation. In *NeurIPS*, 2019.

[22] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[23] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

[24] Xiaoxiao Li, Meirui JIANG, Xiaofei Zhang, Michael Kamp, and Qi Dou. FedBN: Federated learning on non-IID features via local batch normalization. In *International Conference on Learning Representations*, 2020.

[25] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[26] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

[27] Liangxi Liu, Feng Zheng, Hong Chen, Guo-Jun Qi, Heng Huang, and Ling Shao. A bayesian federated learning framework with online laplace approximation. *arXiv preprint arXiv:2102.01936*, 2021.

[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[29] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR, 2019.

[30] Thomas Nickson, Tom Gunter, Chris Lloyd, Michael A Osborne, and Stephen Roberts. Blitzkriging: Kronecker-structured stochastic gaussian processes. *arXiv preprint arXiv:1510.07965*, 2015.

[31] Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. In *Uncertainty in Artificial Intelligence*, pages 1206–1216. PMLR, 2021.

[32] Sebastian W Ober and Carl Edward Rasmussen. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019.

[33] Junier B Oliva, Avinava Dubey, Andrew G Wilson, Barnabás Póczos, Jeff Schneider, and Eric P Xing. Bayesian nonparametric kernel-learning. In *Artificial intelligence and statistics*, pages 1078–1086. PMLR, 2016.

[34] Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[35] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[36] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

[37] Walter Rudin. *Fourier analysis on groups*. Courier Dover Publications, 2017.

[38] Yunus Saatçi. *Scalable inference for structured Gaussian process models*. PhD thesis, Citeseer, 2012.

[39] Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate gaussian process regression. *Advances in neural information processing systems*, 15, 2002.

[40] Matthias W Seeger, Christopher KI Williams, and Neil D Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics*, pages 254–261. PMLR, 2003.

[41] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[42] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

[43] Aman Sinha and John C Duchi. Learning kernels with random features. *Advances in Neural Information Processing Systems*, 29, 2016.

[44] Alex Smola and Peter Bartlett. Sparse greedy gaussian process regression. *Advances in neural information processing systems*, 13, 2000.

[45] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

[46] Minxue Tang, Xuefei Ning, Yitu Wang, Yu Wang, and Yiran Chen. Fedgp: Correlation-based active client selection strategy for heterogeneous federated learning. *arXiv preprint arXiv:2103.13822*, 2021.

[47] Joseph J Thompson, Mark R Blair, Lihan Chen, and Andrew J Henrey. Video game telemetry as a critical tool in the study of complex skill learning. *PloS one*, 8(9):e75129, 2013.

[48] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

[49] Gia-Lac Tran, Edwin V Bonilla, John Cunningham, Pietro Michiardi, and Maurizio Filippone. Calibrating deep convolutional gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1554–1563. PMLR, 2019.

[50] Athanasios Tsanas, Max Little, Patrick McSharry, and Lorraine Ramig. Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests. *Nature Precedings*, pages 1–1, 2009.

[51] Pınar Tüfekci. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140, 2014.

[52] Sathishkumar VE and Yongyun Cho. A rule-based model for seoul bike sharing demand prediction using weather data. *European Journal of Remote Sensing*, 53(sup1):166–183, 2020.

[53] Maxime Vono, Vincent Plassier, Alain Durmus, Aymeric Dieuleveut, and Eric Moulines. Qlsd: Quantised langevin stochastic dynamics for bayesian federated learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*,

volume 151 of *Proceedings of Machine Learning Research*, pages 6459–6500. PMLR, 28–30 Mar 2022.

[54] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2019.

[55] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[56] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[57] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[58] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International conference on machine learning*, pages 1775–1784. PMLR, 2015.

[59] Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, 29:2586–2594, 2016.

[60] Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015.

[61] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[62] Max A Woodbury. *Inverting modified matrices*. Statistical Research Group, 1950.

[63] Ang Xie, Feng Yin, Yue Xu, Bo Ai, Tianshi Chen, and Shuguang Cui. Distributed gaussian processes hyperparameter optimization for big data using proximal admm. *IEEE Signal Processing Letters*, 26(8):1197–1201, 2019.

[64] Feng Yin, Zhidi Lin, Qinglei Kong, Yue Xu, Deshi Li, Sergios Theodoridis, and Shuguang Robert Cui. Fedloc: Federated learning framework for data-driven cooperative localization and location data processing. *IEEE Open Journal of Signal Processing*, 1:187–215, 2020.

[65] Fransisco Zamora-Martinez, Pablo Romeu, Pablo Botella-Rocamora, and Juan Pardo. On-line learning of indoor temperature forecasting models towards energy efficiency. *Energy and Buildings*, 83:162–172, 2014.

[66] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

APPENDICES

Appendix A

Further Details

A.1 Prior Covariance

In section 2.1.2, we stated that we only work with $\mathbf{\Sigma} = \lambda^2\mathbf{I}$ for the prior covariance of the BLR. This is due to the fact that we use random features to recover the true underlying kernel. In general, a BLR with basis functions ϕ and prior

$$\Pr(\mathbf{w}) = N(\mathbf{0}, \mathbf{\Sigma}) \tag{A.1}$$

is equivalent to a GP with the following prior kernel function:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] && \text{(A.2)} \\ &= \phi(\mathbf{x})^\top \mathbb{E}[\mathbf{w}\mathbf{w}^\top]\phi(\mathbf{x}') && \text{(A.3)} \\ &= \phi(\mathbf{x})^\top (\mathrm{Cov}(\mathbf{w}) + \mathbb{E}[\mathbf{w}]\mathbb{E}[\mathbf{w}]^\top)\phi(\mathbf{x}') && \text{(A.4)} \\ &= \phi(\mathbf{x})^\top \mathbf{\Sigma}\phi(\mathbf{x}') && \text{(A.5)} \\ &= \sum_{i,j} \phi(\mathbf{x})_i \mathbf{\Sigma}_{i,j} \phi(\mathbf{x}')_j && \text{(A.6)} \end{aligned}$$

, where the subscripts means the i^{th} or j^{th} entry of the vector/matrix. In our approach, each ϕ consists of a set of samples from $\mathbf{z}$, where $\phi(\mathbf{x})_i$ and $\phi(\mathbf{x}')_i$ are sampled at the same time for any i. We construct the kernel as $URK(\mathbf{x}, \mathbf{x}') = \mathbb{E}[\mathbf{z}(\mathbf{x})\mathbf{z}(\mathbf{x})'] \approx \phi(\mathbf{x})^\top \phi(\mathbf{x}')/\sqrt{m-1}$. When $\mathbf{\Sigma} \neq \lambda^2\mathbf{I}$, there will be interaction terms between different random samples $\phi(\mathbf{x})_i \mathbf{\Sigma}_{i,j} \phi(\mathbf{x}')_j, i \neq j$, which does not have any reasonable meaning or interpretation, and we will not be recovering any form of the desired URK.

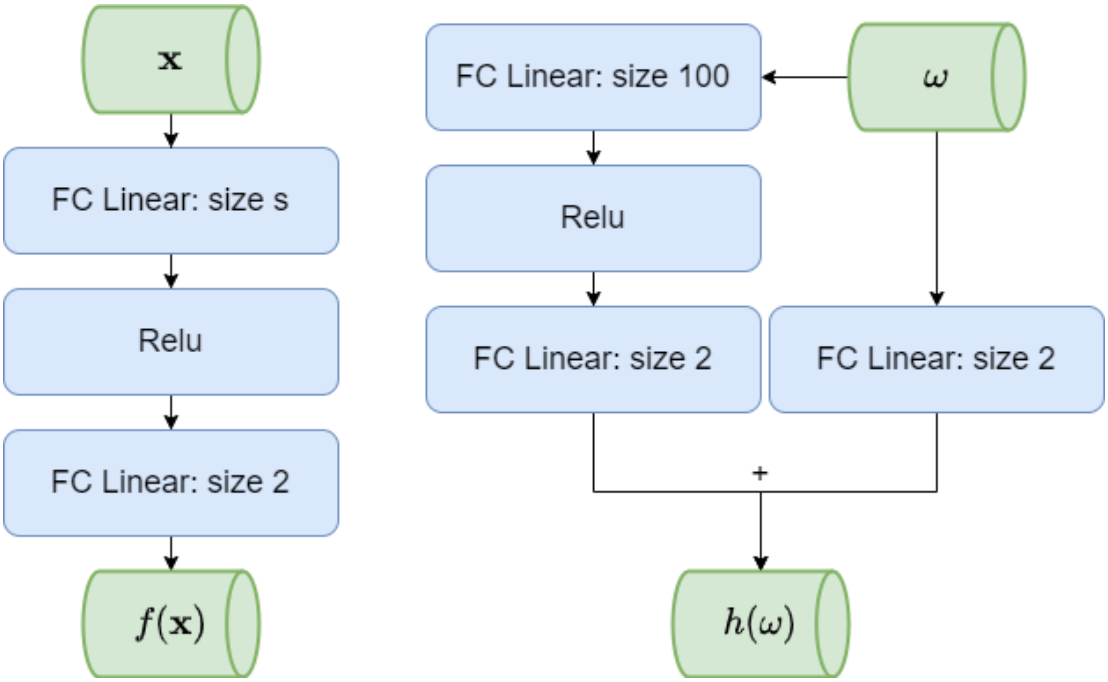

Figure A.1: Architecture of DNNs used in UCI experiments. Left: the feature extractor for $\mathbf{x}$; right: the distribution shifter for ω. FC is short for "fully connected". The layer size s varies for different datasets.

A.2 Experiment Hyperparameters

Figure A.1 shows the architecture of DNNs used in the UCI experiment. For FedAvg, FedProx, and pFedGP, we used the left DNN as their feature extractor. For our methods, we used the rightmost architecture in Figure 3.1 with the same feature extractor f, a very small distribution shifter h (the right DNN in Figure A.1), and $\omega \sim N(\mathbf{0}, \mathbf{I}_5)$.

We run each random seed of each dataset on 1 CPU and 1 NVIDIA T4 GPU with 16GB RAM. Some important hyperparameters are listed in Table A.1. These hyperparameters are selected through grid searching, as suggested by FedProx.

Table A.1: Hyperparameters used in the UCI experiment. For FedLoc $L = 10\rho$ to run the Proximal ADMM algorithm.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
size s	200		2000		2000		5000		5000	
FedProx μ	0.5	1.0	0.1	1.0	1.0	1.0	0.1	0.01	0.001	1.0
FedLoc ρ	5e3	5e4	5e3	5e4	1e3	5e3	5e4	5e4	1e5	1e5
FedBNR-KD α	10	2	1	0.5	5	2	5	0.5	5	5