Bayesian Federated Learning in Predictive Space

by

Mohsin Hasan

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, 2023

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

Some sections of Chapter 2 and Chapter 3 were adapted from a paper submitted to AISTATS 2023, for which I was the first author. For that work, I collaborated with another student, Zehao Zhang, who aided in running some experiments reported in Chapter 3. For the work in this thesis, I collaborated with my supervisor Pascal Poupart, and other researchers: Guojun Zhang, Xi Chen, and Kaiyang Guo, who all provided feedback and discussions. All equations, proofs, algorithms and text in this thesis (and the AISTATS 2023 paper mentioned earlier) were developed and written by myself.

Abstract

Federated Learning (FL) involves training a model over a dataset distributed among clients, with the constraint that each client's data is private. This paradigm is useful in settings where different entities own different training points, such as when training on data stored on multiple edge devices. Within this setting, small and noisy datasets are common, which highlights the need for well-calibrated models which are able to represent the uncertainty in their predictions. Alongside this, two other important goals for a practical FL algorithm are 1) that it has low communication costs, operating over only a few rounds of communication, and 2) that it achieves good performance when client datasets are distributed differently from each other (are heterogeneous). Among existing FL techniques, the closest to achieving such goals include Bayesian FL methods which collect parameter samples from local posteriors, and aggregate them to approximate the global posterior. These provide uncertainty estimates, more naturally handle data heterogeneity owing to their Bayesian nature, and can operate in a single round of communication. Of these techniques, many make inaccurate approximations to the high-dimensional posterior over parameters which in turn negatively effects their uncertainty estimates. A Bayesian technique known as the "Bayesian Committee Machine" (BCM), originally introduced outside the FL context, remedies some of these issues by aggregating the Bayesian posteriors in the lower dimensional predictive space instead.

The BCM, in its original form, is impractical for FL due to requiring a large ensemble for inference. We first argue that it is well-suited for heterogeneous FL, then propose a modification to the BCM algorithm, involving distillation, to make it practical for FL. We demonstrate that this modified method outperforms other techniques as heterogeneity increases. We then demonstrate theoretical issues with the calibration of the BCM, namely that it is systematically overconfident. We remedy this by proposing β-Predictive Bayes, a Bayesian FL algorithm which performs a modified aggregation of the local predictive posteriors, using a tunable parameter β. β is tuned to improve the global model's calibration, before it is distilled. We empirically evaluate this method on a number of regression and classification datasets to demonstrate that it generally better calibrated than other baselines, over a range of heterogeneous data partitions.

Acknowledgements

I'd like to thank my supervisor Pascal Poupart for his constant support, advice, understanding and patience throughout my entire Masters degree.

I'd also like to thank my collaborators: Zehao Zhang, Guojun Zhang, Xi Chen, Mahdi Karami, Kaiyang Guo, Ahmad Rashid and Haolin Yu for useful ideas, and providing regular feedback and discussions concerning my work.

I'd like to thank my committee members Yaoliang Yu, and Hongyang Zhang for useful advice and feedback regarding my work.

I'd like to thank all my cousins for visiting me from the States and cheering me on through my presentation.

I'd like to thank my parents and siblings for supporting me and putting up with me as I dealt with the challenges of my degree and with COVID, I wouldn't have been able to do anything without them.

Dedication

This is dedicated to my parents.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgements v

Dedication vi

List of Figures x

List of Tables xi

1 Introduction 1

1.1 Contributions . . . 3

1.2 Thesis Outline . . . 3

2 Background 5

2.1 Bayesian Learning . . . 5

2.1.1 Approximate Bayesian Learning . . . 7

2.2 Federated Learning . . . 10

2.2.1 Related Works - Bayesian Federated Learning . . . 12

2.2.2 Related Works - One-Shot Federated Learning 14
2.2.3 Knowledge Distillation 15
2.2.4 Evaluating Model Calibration 15

3 The Distilled BCM Algorithm for Federated Learning **17**
3.1 Bayesian Committee Machine 17
3.1.1 Aggregation for Classification 19
3.1.2 Aggregation for Regression 19
3.2 Applying the BCM to Federated Learning 22
3.3 Experiments 24
3.3.1 Classification Datasets 25
3.3.2 Regression Datasets 26
3.3.3 Models 26
3.3.4 Baselines 26
3.3.5 Training Details 27
3.3.6 Classification Results 27
3.3.7 Regression Results 28

4 Calibrating the BCM: β-Predictive Bayes **29**
4.1 Analyzing the Calibration of the BCM 29
4.1.1 Analyzing the Predictive Mixture Model 33
4.1.2 Heuristic Argument for Classification 36
4.2 Calibrating the Aggregated Model 37
4.3 Experiments 39
4.3.1 Classification Results 39
4.3.2 Regression Results 40

5 Conclusion **44**
5.1 Limitations and Future Work 44

References 46

APPENDICES 53

A Gaussian Regression Formula 54

B Additional Experiments and Hyperparameters for D BCM 57

B.1 Additional Experimental Details 57

B.1.1 Hardware, Software, and Randomization Details 57

B.2 Hyperparameter Tuning 57

B.2.1 Heterogeneous Classification Dataset Construction 59

B.3 Additional Experiments 59

B.3.1 Classification Experiments 59

List of Figures

3.1 Test accuracies on the classification datasets with increasing data heterogeneity (tested with parameter settings of h={0.0, 0.3, 0.6, 0.9}). SR = single round method, MR= multi-round method. Averages and standard error over 10 seeds are reported. 25

4.1 Negative log likelihoods on the classification datasets with increasing data heterogeneity (tested with parameter settings of h={0.0, 0.3, 0.6, 0.9}). Averages and standard error over 5 seeds are reported. The omitted values (eg. for FedPA on EMNIST) denote results where the negative log likelihood diverged. 41

4.2 Expected Calibration Errors on the classification datasets with increasing data heterogeneity (tested with parameter settings of h={0.0, 0.3, 0.6, 0.9}). Averages and standard error over 5 seeds are reported. 42

List of Tables

3.1 Average test accuracies (± standard error) on classification datasets for $h = 0.3$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test). 21

3.2 Average test mean squared error (± standard error) on regression datasets, based on 10 seeds. **Lower is better**. Multi-round methods are written above the line, while methods run for a single round of communication are written below. The best technique over 1 round is bolded. $\Uparrow$ / $\Downarrow$: higher/lower mean squared error with $p < 1\%$, $\uparrow$ / $\downarrow$: lower/higher mean squared error with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test). 22

4.1 Summary of Predictive Variance Analysis 37

4.2 Average negative log likelihood (± standard error) on regression datasets, based on 5 seeds. **Lower is better**. 43

B.1 The hyperparameters tuned, their possible values in the grid search, and the algorithms each hyperparameter applies to. 61

B.2 The tuned values of hyperparameters for the classification datasets in the homogeneous case $h = 0$ 62

B.3 The tuned values of hyperparameters for the classification datasets, in the heterogeneous case $h > 0$ 63

B.4 The tuned values of hyperparameters for the regression datasets 64

B.5 Average test accuracies (± standard error) on classification datasets for $h = 0.0$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test). 65

B.6 Average test accuracies (± standard error) on classification datasets for $h = 0.6$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test). 65

B.7 Average test accuracies (± standard error) on classification datasets for $h = 0.9$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test). 66

Chapter 1

Introduction

Federated learning (FL) is a machine learning paradigm which trains a statistical model using decentralized data stored on client devices, with the constraint that client data is kept private (ie. not shared directly) [40]. FL has found use in smartphone applications, as well as many fields, such as healthcare, and finance due to the abundance of use-cases where sensitive training data is owned by separate entities [66].

The workflow of a typical FL algorithm involves the local training of models on each client, followed by the communication and aggregation of these into a single global model on a central server. Many FL techniques alternate between these two steps until some notion of convergence is reached [40].

For the purpose of this work, we are concerned with the design of FL algorithms with three key metrics in mind:

1. **Communication Cost**: the transmission of models between clients and servers can be expensive, especially when each model has many parameters. This is the case, for instance, when training neural networks. Cost effective FL techniques therefore aim to maximize the amount of local training at each client, while reducing the rounds of communication.

2. **Performance with Heterogeneous Data**: clients may have different data distributions from each other. This causes the local client models to differ, which presents issues when aggregating them into a global model [65]. In particular, when client data is heterogeneous, global models tend to perform poorly on the data of any one client.

3. **Calibration**: Clients may have data with too few training points, or too much variance. Therefore, a valuable goal in FL is to produce models that are **well calibrated**, or in other words: make probabilistic predictions with accurate uncertainty estimates.

Many techniques frame FL as a distributed optimization problem. For such methods, there is typically a trade-off between achieving a low communication cost, and good performance on heterogeneous data. In particular, they deal with heterogeneous data by having more frequent communication between clients to "synchronize" the models, which has the downside of increasing communication costs [40, 60]. These methods also don't have any systematic way of calibrating the global model.

Another branch of FL techniques instead take a Bayesian perspective on learning the model. They approximate the Bayesian posterior distribution of each client model, and aggregate them into the global Bayesian posterior [1, 46]. This allows the training of more effective global models on heterogeneous data by leveraging client uncertainty during aggregation [1]. Certain Bayesian FL methods also demonstrate how to aggregate local posteriors in only a single round of communication [46]. Furthermore, these methods explicitly represent and adjust the uncertainty in model parameters, which, in principle, means they are well-calibrated.

The issue with these methods is that, since the Bayesian posterior is a distribution over model parameters, it is a complex object that is expensive to represent and manipulate. Techniques therefore require some sort of approximation to the posterior. Many methods work with crude approximations to the client posterior (as eg. a multivariate Gaussian) [1, 46], which can incur heavy approximation error, resulting in poor calibration and accuracy.

Another Bayesian method, the "Bayesian Committee Machine" (BCM), originating in Gaussian process literature, instead aggregates the distribution over model predictions (referred to as the **Bayesian predictive posterior**) rather than the posterior over parameters [56]. The former can be approximated more closely since it is a lower dimensional distribution. The trade-off is that these methods instead need to rely on an approximate aggregation technique, which biases the global model (particularly in its calibration). These methods are also expensive for FL due to needing to predict using an ensemble over Bayesian models (each of which is in and of itself an ensemble model over posterior samples) at inference.

This thesis therefore aims to investigate practical algorithms for Bayesian FL which operate in a single round of communication, perform well on heterogeneous data, and provide accurate uncertainty estimates.

1.1 Contributions

The primary contributions of this work are:

- We argue that the BCM algorithm is well suited for the heterogeneous data setting in FL. We adapt this algorithm by distilling the ensemble model at a central server. We empirically demonstrate that this algorithm outperforms other FL algorithms on datasets with high heterogeneity, with respect to accuracy, despite only operating in a single round of communication.
- We propose a novel algorithm for Bayesian FL, called β-Predictive Bayes. This method, operating in a single round of communication, combines the Bayesian posterior over model predictions to construct an ensemble model using a tunable parameter β. β is trained to adjust the ensemble's calibration, before it is distilled into a single model for use at clients. The fact that the method operates over model predictions rather than model parameters lets the method enjoy better approximations to client posterior predictions. On the other hand, the fact the method uses a tunable parameter in its aggregation allows it to adjust for the systematic bias in aggregation techniques such as in the BCM. This lets it attain better calibration performance. The distillation step allows the global model to be used cheaply at inference.
- An empirical evaluation of β-Predictive Bayes on multiple regression and classification datasets, using partitions simulating varying levels of data heterogeneity. The proposed technique is competitive with or outperforms other baselines with respect to both accuracy and calibration, particularly as data heterogeneity increases.

1.2 Thesis Outline

This thesis is split into 5 chapters. The main content of the thesis is summarized below.

- **Chapter 2** presents a primer on Bayesian inference, Federated Learning, as well as other tools used throughout the work such as Knowledge Distillation.
- **Chapter 3** discusses the BCM as a starting point. The chapter argues that the assumptions underlying the BCM aggregation formula are natural in heterogeneous FL settings, and proposes a method for practically using it in FL. It then evaluates the accuracy of the adapted BCM method on multiple datasets, with varying levels of data heterogeneity.

- **Chapter 4** analyzes calibration issues with the BCM method, and proposes a novel aggregation technique to correct them. The new algorithm (β-Predictive Bayes) is demonstrated to be better calibrated on multiple regression and classification datasets, even as data heterogeneity increases.

Chapter 2

Background

We begin with a brief summary of Bayesian learning, and Federated learning, since both topics are needed to understand the content of this work.

In what follows, we focus the discussion on a supervised learning problem, and denote the dataset as $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\} = (\mathbf{X}, \mathbf{y})$.

2.1 Bayesian Learning

Bayesian learning is a method for constructing a statistical model given a dataset. Its main feature, in contrast with traditional (optimization-based) learning, is that it takes into account uncertainty over parameters [4].

Suppose we have data $\mathcal{D}$, and model parameters $\theta \in \mathbf{\Theta}$. The model is such that, given the parameters θ, it maps an input x into a **distribution** over outputs y. That is, the model outputs $f_\theta(x) = p(y|x, \theta)$.

For example, in a regression problem where the goal is to estimate a function given noisy observations, $y = f(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_o^2)$, a possible model is $f_\theta(x) = p(y|x, \theta) = \mathcal{N}(\theta_m^\top x, \theta_s^2)$. Here the overall model parameters are $\theta = [\theta_m, \theta_s]^\top$, consisting of components which influence the mean of the prediction (θ_m) and a scalar which captures its standard deviation (θ_s). For classification, $f_\theta(x) = p(y|x, \theta)$ would correspond to the distribution over class labels.

A Bayesian approach operates by setting a **model space prior** $p(\theta)$, a distribution over the model parameters. The prior encapsulates beliefs about the parameters before observing data.

For instance, in the regression example, if we expect the coefficients θ_m and observation noise θ_s to be small, one choice of prior may be a Gaussian centered at 0, and with a small variance ie. $p(\theta) = \mathcal{N}(0, 1)$.

Upon processing the dataset, this prior is updated to the **model space posterior**. The posterior is expressed as the conditional distribution of the parameters, given the data, ie. $p(\theta|\mathcal{D})$. The update from the prior to the posterior is done using Bayes' rule:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{2.1}$$

The term $p(\mathcal{D}|\theta)$ is referred to as the **likelihood**, and is viewed as a function of θ. It captures how likely the observed data is to be generated by the model given that model parameters are set to θ. Assuming we have independent and identically distributed samples (i.i.d.) for the dataset $\mathcal{D}$, we can factor the likelihood into $p(\mathcal{D}|\theta) = \prod p((y_i, x_i)|\theta)$. If we further assume that we are not modelling the distribution over inputs x, and are only interested in the distribution over y given an input x (as is done in supervised learning), we may write:

$$p(\mathcal{D}|\theta) = \prod_{(x_i,y_i)\in\mathcal{D}} p(y_i|x_i, \theta) \tag{2.2}$$

The term in the denominator, referred to as the **marginal likelihood** is the normalization constant needed to make the sure the posterior integrates to 1. In other words:

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta \tag{2.3}$$

Since $p(\mathcal{D})$ is only a constant, we can still distinguish more likely settings of the parameters, or do interesting operations on the posterior, without it. For this reason, in Bayesian inference it is common to write the posterior update as $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$, and ignore the $p(\mathcal{D})$ term.

This posterior is used to make predictions. To do so, we are interested in the distribution $p(y|x, \mathcal{D})$, referred to as the **(Bayesian) predictive posterior**. We can obtain this distribution using the sum and product rule of probability:

$$p(y|x,\mathcal{D}) = \int p(y|x,\theta)p(\theta|\mathcal{D})d\theta \tag{2.4}$$

$$= \mathbb{E}_{\theta\sim p(\theta|\mathcal{D})}[p(y|x,\theta)] \tag{2.5}$$

The above equation is said to **marginalize** out the parameters θ. For a well-chosen prior and model, $p(y|x,\mathcal{D})$ may be a well-calibrated distribution. In other words, the uncertainty in the distribution $p(y|x,\mathcal{D})$ will accurately reflect the noise present in the data (aleatoric uncertainty), and the uncertainty the model may have due to seeing too little data (epistemic uncertainty).

2.1.1 Approximate Bayesian Learning

The primary challenge in Bayesian learning is how the posterior is used to compute predictions according to equation 2.4. In some cases, such as linear regression, where we set the prior to be a Gaussian distribution and use a Gaussian model (similar to the example), the posterior can be analytically computed using 2.1. It will be a Gaussian with some data-dependent mean $\mu(\mathcal{D})$ and covariance $\Sigma(\mathcal{D})$ [4]. In this case, the posterior can be stored using this mean and covariance. The integral in equation 2.4 can then be carried out analytically on some given test-point x, to return a Gaussian (over y).

In more complicated settings, such as if the model $f_\theta(x) = p(y|x,\theta)$ is a neural network, the integral 2.4 can no longer be computed in closed-form (even if the prior is a simple distribution such as a Gaussian). In this case, the posterior is said to be **intractable**.

Since we can no longer exactly work with the posterior, we need to make some approximations, which moves us into the area known as "Approximate Bayesian learning". Here, there are two broad classes of methods of dealing with intractable posteriors. Both make use of the fact that, although the posterior may not be integrated, we can still evaluate the un-normalized posterior at some θ using $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$.

Variational Methods

Variational methods [55], construct simple distribution over the model parameters θ, denoted $q_w(\theta)$ with distribution parameters w. For instance, $q_w(\theta)$ may be a Gaussian, where w are the mean and covariance of the Gaussian. q_w is chosen to be easy to sample exactly from, and evaluate the density of. The parameters w are tuned to minimize some notion of

distance between the distribution $q_w(\theta)$ and the posterior $p(\theta|\mathcal{D})$. One such example is the Kullback-Leibler (KL) divergence between $q_w(\theta)$ and $p(\theta|\mathcal{D})$. This objective is commonly used because it can be manipulated into a form that may be evaluated even if the posterior is intractable:

$$\begin{aligned}
w^* &= \underset{w}{\operatorname{argmin}}\, \mathrm{KL}(q_w||p) \\
&= \underset{w}{\operatorname{argmin}} \int q_w(\theta) \log \frac{q_w(\theta)}{p(\theta|\mathcal{D})} d\theta \\
&= \underset{w}{\operatorname{argmin}} \int q_w(\theta) \log \frac{q_w(\theta)}{p(\theta|\mathcal{D})} d\theta \\
&= \underset{w}{\operatorname{argmin}} - \int q_w(\theta) \log \frac{p(\theta|\mathcal{D})}{q_w(\theta)} d\theta \\
&= \underset{w}{\operatorname{argmin}} - \int q_w(\theta) \log \frac{p(\theta)p(\mathcal{D}|\theta)}{q_w(\theta)} d\theta + \int q_w(\theta) \log p(\mathcal{D}) d\theta \\
&= \underset{w}{\operatorname{argmin}} - \int q_w(\theta) \log \frac{p(\theta)p(\mathcal{D}|\theta)}{q_w(\theta)} d\theta + \log p(\mathcal{D}) \\
&= \underset{w}{\operatorname{argmin}} - \int q_w(\theta) \log \frac{p(\theta)p(\mathcal{D}|\theta)}{q_w(\theta)} d\theta
\end{aligned} \tag{2.6}$$

We can constrain the distribution $q_w(\theta)$ so that we can evaluate the integral in the last line (by sampling $q_w(\theta)$). The $p(\theta)p(\mathcal{D}|\theta)$ term can also be evaluated analytically by iterating over the dataset. Thus, the last line gives an objective (also known as the negative Evidence Lower Bound (ELBo)), which may be minimized to obtain an approximation for the posterior.

Once $q_{w^*}(\theta)$ is obtained, we may sample from it to compute the expectation required for predictions 2.4.

It is worth noting that variational methods give an often biased approximation of the posterior, since the true posterior may not lie in the family of distributions of the form q_w

Markov Chain Monte Carlo Sampling

Another set of techniques for working with intractable posteriors is to obtain samples from the posterior, $S = \{\theta_1, ..., \theta_M\} \sim p(\theta|\mathcal{D})$ through a method called Markov Chain Monte Carlo (MCMC), and use them to approximate 2.4 as:

$$p(y|x,\mathcal{D}) = \int p(y|x,\theta)p(\theta|\mathcal{D})d\theta$$
$$\approx \frac{1}{M}\sum_j p(y|x,\theta_j) \tag{2.7}$$

MCMC works by starting from a setting of the parameters θ_0, and iteratively updating parameters according to a transition kernel $T(\theta'|\theta)$ [45]:

$$\theta_{i+1} \sim T(\theta'|\theta_i) \tag{2.8}$$

The iterates $\theta_1, ..., \theta_m$ move through parameter space, and are thought of as samples from the posterior. The transition kernel is designed so that it can easily be sampled from, and so that the chain of iterates spends time in each region proportional to its posterior probability. The transition kernel needs to obey certain conditions which allows the chain to asymptotically converge to the true samples (in the sense that expectations estimated using them will converge to their true values) [54].

Common and efficient MCMC methods make use of the gradient of the log-posterior, $\nabla_\theta \log p(\theta|\mathcal{D})$ in their transition kernel. For instance, the un-adjusted Langevin algorithm (ULA) is a common MCMC technique with the transition kernel [**?**]:

$$\theta_{t+1} \sim \mathcal{N}(h\nabla \log p(\theta|\mathcal{D}), 2h) \tag{2.9}$$

Where h is the step-size, a hyperparameter. The parameter updates can equivalently be written:

$$\theta_{i+1} = \theta_i + h\nabla \log p(\theta|\mathcal{D}) + \sqrt{2h}\eta_i \tag{2.10}$$
$$\eta_i \sim \mathcal{N}(0, I) \tag{2.11}$$

The gradient of the log-posterior can be computed since:

$$\begin{aligned}
\nabla_\theta \log p(\theta|\mathcal{D}) &= \nabla \log p(\theta) + \nabla \log p(\mathcal{D}|\theta) - \nabla \log p(\mathcal{D}) \\
&= \nabla \log p(\theta) + \nabla \log p(\mathcal{D}|\theta) \\
&= \nabla \log p(\theta) + \nabla \log \prod_{i=1}^{|\mathcal{D}|} p(y_i|x_i, \theta) \\
&= \nabla \log p(\theta) + \sum_{i=1}^{|\mathcal{D}|} \nabla \log p(y_i|x_i, \theta)
\end{aligned}$$

Combining these facts, the update equation in 2.10 resembles the gradient ascent used in traditional optimization based training, but with Gaussian noise added at each step. Many MCMC methods can be thought of as noisy modifications of corresponding optimization methods. In fact, Stochastic Gradient Langevin Dynamics [59] is a modification to ULA which approximates the likelihood gradient with a minibatch gradient, meaning it is the MCMC "equivalent" of Stochastic Gradient Descent (SGD).

Unlike the samples from the variational posterior $q_w(\theta)$, the MCMC samples will be approximate, since it is difficult to directly sample from the true posterior $p(\theta|\mathcal{D})$.

In this work, when mentioning Bayesian learning, we will typically assume MCMC is being used, due to its similarity with traditional gradient based training. However, most Bayesian FL aggregation techniques are agnostic to the underlying method being used, and the MCMC samples may be replaced with samples from the approximate posterior $q_{w^*}(\theta)$.

2.2 Federated Learning

In federated learning (FL), data is distributed across several clients. Let $\mathcal{D} = \mathcal{D}_1 \cup ... \cup \mathcal{D}_n$ where $\mathcal{D}_i = \{(x_1, y_1), ..., (x_{k_i}, y_{k_i})\}$ is the dataset of size k_i at client i. The goal of FL is to learn a predictive model over the entire dataset $\mathcal{D}$, without any data leaving each client, in order to preserve privacy.

A typical FL algorithm consists of the two basic steps:

1. Local training: each client learns a local model $f_{\theta_i}(x)$, using its dataset $\mathcal{D}_i$.
2. Global aggregation: the clients send their local models to a central server, which combines them to form a global model f_{θ_g} (which need not be of the same form as the local models).

For many algorithms, these two steps are alternated repeatedly until the global model reaches some form of convergence. In this case, after the global model is formed, it is sent back to the clients, which either copy the global weights to their model ($\theta_i \leftarrow \theta_g$) if the global model is of the same form as the local models. Otherwise the clients use it to update the local model in some manner.

Federated Averaging (FedAvg) [40] is the prototypical example of a Federated Learning algorithm. It implements the above two steps as:

1. FedAvg local training: the local models θ_i are trained for l epochs on their local datasets using SGD. Ie. an update on $theta_i$ is done as $\theta_i' = \theta_i - \tilde{\nabla}(\frac{1}{k_i}\sum_{j=1}^{k_i} l(x_j, y_j; \theta_i))$, where $\tilde{\nabla}$ refers to a minibatch approximation of the gradient, and $l(x_j, y_j; \theta_i)$ is the per-training point loss.

2. FedAvg global aggregation: the global model is formed by taking the average of the local parameters $\theta_g = \sum_i k_i \theta_i / (\sum_i k_i)$.

The two steps are repeated over multiple rounds of communication.

Suppose we denote the previous global model with θ_g, which is copied over to the local models $\theta_i \leftarrow \theta_g$, and we perform the local and global updates in one round to obtain θ_i' and θ_g'. The intuition behind FedAvg is that, assuming all local models are initialized identically, and local training is done only for a single iteration, the combination of the two steps corresponds to:

$$
\begin{aligned}
\theta_g' &= \sum_i \frac{k_i}{|\mathcal{D}|}\theta_i' \\
&= \sum_i \frac{k_i}{|\mathcal{D}|}(\theta_i - \tilde{\nabla}(\frac{1}{k_i}\sum_{j=1}^{k_i} l(x_j, y_j; \theta_i)) \\
&= \sum_i \frac{k_i}{|\mathcal{D}|}(\theta_g - \tilde{\nabla}(\frac{1}{k_i}\sum_{j=1}^{k_i} l(x_j, y_j; \theta_g)) \\
&= \theta_g - \sum_i \frac{k_i}{|\mathcal{D}|}\tilde{\nabla}(\frac{1}{k_i}\sum_{j=1}^{k_i} l(x_j, y_j; \theta_g)) \\
&= \theta_g - \tilde{\nabla}(\frac{1}{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{D}|} l(x_j, y_j; \theta_g))
\end{aligned}
$$

Where the last step is equivalent to performing an SGD update on the global parameters. Thus, in the limit of taking only 1 step of local training per round, FedAvg is equivalent to SGD on the global parameters.

As we increase the number of local iterations for training, the derivation above may not hold. In particular, if the data is **heterogeneous**, meaning that client datasets follow different distributions (in either the targets, $p(y)$, the inputs $p(x)$ or the relation between the two $p(y|x)$), then the above doesn't hold and FedAvg is no longer equivalent to performing global SGD. In fact, in this case, local client models may begin to diverge from each other, causing the global model to achieve poor performance [60]. To avoid this, methods such as FedAvg require more frequent rounds of communication. This can be quite costly in terms of communication costs.

Many FL algorithms use FedAvg as a template, and perform some modification to it with the goal of achieving faster convergence, or better performance in the heterogeneous data setting [36, 58, 57, 38, 42].

For instance:

- FedProx [36]: changes the local training step by adding a penalty between the global model and each client's local model to the training loss. Ie. the local update becomes:

$$\theta_i' = \theta_i - \tilde{\nabla}(\lambda||\theta_i - \theta_g|| + \frac{1}{k_i}\sum_{j=1}^{k_i} l(x_j, y_j; \theta_i)) \tag{2.12}$$

  The idea is that this penalty term keeps local models more closely synchronized with the global model, preventing (or slowing) divergence in the case of heterogeneous data.

- "AdaptiveFL" [51] uses an adaptive optimizer (such as Adam [32]) at the server for aggregating local models, instead of simply averaging them.

2.2.1 Related Works - Bayesian Federated Learning

Bayesian learning offers FL techniques the advantage of better performance in heterogeneous settings. As argued by [1], FedAvg can be thought of as a technique which obtains the mean of the global posterior, if each local posterior is approximated as a Gaussian

with the identity as the covariance matrix. Thus FedAvg implicitly assumes a form of homogeneity which may not be practical. Bayesian techniques offer the ability to remedy this by using more realistic approximations to the local posteriors.

The basic template for Bayesian FL is analogous to the one for traditional FL:

1. Local sampling: samples are drawn from the local Bayesian posteriors $\{\theta_i^1, \theta_i^2 ..., \theta_i^m\} \sim p(\theta|\mathcal{D}_i)$ using MCMC at each client.

2. Global aggregation: The local samples, or information gathered from the samples, is communicated to a server, which uses it to obtain an approximation for the global posterior $p(\theta|\mathcal{D})$

Many methods differ primarily in how they perform the aggregation step.

"**Embarrassingly Parallel MCMC**" [46] aggregates information from the local posteriors using the equation

$$\begin{aligned} p(\theta|\mathcal{D}) &= \frac{1}{p(\mathcal{D})} p(\mathcal{D}_1, ..., \mathcal{D}_n|\theta) p(\theta) \\ &= \frac{p(\theta)}{p(\mathcal{D})} \prod_{i=1}^{n} p(\mathcal{D}_i|\theta) \\ &= \frac{1}{p(\mathcal{D})} \prod_{i=1}^{n} p(\theta|\mathcal{D}_i) \frac{p(\mathcal{D}_i)}{p(\theta)} \\ &= \frac{1}{p^{n-1}(\theta)} \prod_{i} p(\theta|\mathcal{D}_i) \end{aligned} \tag{2.13}$$

In other words, the local posteriors may be multiplied (with a corrective factor from the prior) to obtain the density of the global posterior. The method uses the local samples to estimate the local densities either as Gaussians (ie. using the sample mean and covariance) or with a kernel density estimator. The parameters of these distributions are sent to the server, which can analytically compute the product 2.13 to obtain a formula for the global posterior density. This global density is then sampled to obtain the desired posterior samples to which Eq. (2.7) may be applied for inference.

It is worth noting that the original work was not designed for use with neural networks, and the memory costs associated with the method make it intractable for this setting. For instance when approximating the local posteriors as Gaussians and multiplying them, a

computational cost of $O(d^3)$ is required for inverting the covariance matrices, where d is the number of neural network parameters. This method is notable however for operating with only a single communication round.

"**Federated Posterior Averaging**" [1] is similar to the above technique, except that it only approximates the local posteriors as Gaussians, and devises a more efficient, iterative algorithm for aggregating the local posteriors (with cost linear in the number of network parameters), which it runs over multiple communication rounds. However, the iterative method doesn't compute the global posterior's covariance, only its mean. This is because the work is more interested in a point estimate for the posterior, rather than capturing the uncertainty in predictions.

The main issue with both these techniques is that they require some approximation of the global parameter posterior (e.g., in the form of a Gaussian), which can often be inaccurate when the number of model parameters is large. Such approximations are especially poor for neural network models, where the model space posterior is known to be multimodal [49].

Although originally meant to speed up the training of Gaussian processes, the "**Bayesian Committee Machine**" (BCM) [56] can be thought of as an aggregation method which combines low dimensional predictive posteriors $p(y|x, \mathcal{D}_i)$, rather than the parameter posteriors $p(\theta|\mathcal{D}_i)$. It essentially uses an analogue of Equation 2.13. This results in an ensemble model over the local Bayesian samples. The advantage of this method is that predictive posteriors are much simpler, and lend themselves to, for instance, Gaussian approximations, without sacrificing accuracy. On the other hand, the aggregation formula is no longer exact, but only approximately true, which causes other inaccuracies. Furthermore, the fact that it creates a large ensemble model makes it inefficient for use in FL. The BCM will be our starting point in developing our proposed method, and its drawbacks and advantages will be discussed in more detail in later chapters.

2.2.2 Related Works - One-Shot Federated Learning

One of the goals of this work is to present a method operating in a single communication round. Bayesian methods aren't the only ones which can achieve this.

"**One-Shot FL**" [23], as well as "**Federated Learning via Knowledge Transfer** (FedKT)" [35] perform one round training by constructing an ensemble from the client models, and compressing it into a single model using knowledge distillation on a public dataset. The methods differ in how they construct the teacher ensemble: whereas "One-Shot FL" averages the client predictions (and was tailored for SVM models), "FedKT" aggregates

based on majority voting by local models (and only applies to classification tasks). The latter technique uses a scheme called "consistent voting", where it uses discrepancies between client votes to determine which clients are uncertain about their predictions, and thus can be ignored in the majority vote. Our method is closely related to these approaches, but derives aggregation rules for the local client models using a Bayesian perspective, and is applicable to any type of task or predictive model. These existing techniques use non-Bayesian methods to obtain the ensemble. This means they do not account for uncertainty within client models, or must use a heuristic such as "consistent voting" to do so [35]. Due to this, the calibration offered by these models may not compare to Bayesian approaches. Furthermore, a Bayesian approach offers other benefits in the case of heterogeneous data, as argued in [1].

2.2.3 Knowledge Distillation

A tool used in this work, and a few mentioned earlier, is knowledge distillation. The goal of knowledge distillation is to compress a given larger "teacher" model into a smaller "student" model which matches its predictions on a shared data distribution [28]. A common technique for distillation is to train the student model on some dataset and add a term to the loss which measures the difference between the teacher and student predictions.

A number of FL techniques assume the server has access to a public (typically unlabeled) dataset $\mathcal{U}$ which serves as the distillation dataset [23, 35, 7]. In this case, the student is trained to minimize a loss of the form:

$$\mathcal{L}(\theta_S) = \sum_{x \in \mathcal{U}} l(f_{\theta_S}(x), f_{\theta_T}(x)) \tag{2.14}$$

Where $f_{\theta_S}(x)$ and $f_{\theta_T}(x)$ are the predictive models of the student and teacher respectively (with parameters θ_S and θ_T). $l(\cdot, \cdot)$ measures the discrepancy between predictions. For classification, this discrepancy can be measured by the Kullback-Leibler divergence between the distributions over classes, while for regression tasks, it can be measured by the mean-squared error.

2.2.4 Evaluating Model Calibration

A model is referred to as well-calibrated if its probabilistic predictions align with their true probabilities. For instance, if a well calibrated model predicts the class "cat" for 100

pictures of cats with probability 0.9, then 90 of these pictures will actually correspond to cats [24]. For regression, a well-calibrated model that predicts a variance should have that variance correspond to the actual observation noise σ_o^2.

One metric for measuring a classification model's calibration is "Expected Calibration Error" (ECE). It works by computing probabilistic predictions on a training set. Then the predictions are binned to intervals based on the highest probability assigned to each class (we denote the probability as $\hat{p}_i$, and the predicted class as $\hat{y}_i$). The mth bin B_m will have $|B_m|$ points, and will have a "confidence" score: $\text{conf}(B_m) = \frac{1}{|B_m|}\sum_i \hat{p}_i$ (ie. the average probability) and an "accuracy" of $\text{acc}(B_m) = \frac{1}{|B_m|}\sum_i 1(\hat{y}_i = y_i)$. The ECE is simply the weighted average of the discrepancy between the confidence and accuracy [24]:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{|\mathcal{D}|}\Big|\text{conf}(B_m) - \text{acc}(B_m)\Big| \tag{2.15}$$

An issue with ECE is that it is sensitive to the exact setting of the bins (ie. how many bins we decide to use). Additionally, some intervals over $\hat{p}_i$ may contain very few points, which can skew the metric. A smaller ECE typically suggests a better calibrated model.

Another metric used to evaluate calibration is the negative log-likelihood:

$$L = -\sum_{i=1}^{|\mathcal{D}|} \log p(y_i|x_i, \theta) \tag{2.16}$$

The reason this is also a calibration score is because (in expectation), it is minimized if and only if the model is well-calibrated [25].

Chapter 3

The Distilled BCM Algorithm for Federated Learning

We take the perspective of learning a global Bayesian model on a dataset composed of n shards: $\mathcal{D} = \mathcal{D}_1 \cup ... \cup \mathcal{D}_n$, each stored on an individual client. We assume we are in the supervised setting so that each datapoint is an input and output pair (x, y). Ultimately we'd like to construct a model which can make predictions using the (global) predictive posterior $p(y|x, \mathcal{D})$.

In this chapter, we first present the "Bayesian Committee Machine" [56], a method for aggregating local Bayesian models into a global Bayesian model. The method is typically seen as a way to reduce the computational cost of training Gaussian processes over large datasets. We argue that the model is well suited for heterogeneous data in the Federated setting, and adapt the method to make it practical for FL. We demonstrate the advantages of the method over other FL aggregation techniques empirically.

3.1 Bayesian Committee Machine

The "Bayesian Committee Machine (BCM)" introduced in [56], approximates the global predictive posterior by multiplying the local client predictive posteriors. This is justified, under certain independence assumptions on the data, by the following observation:

Theorem 1. *If the data shards are independent, i.e.,* $p(\mathcal{D}) = \prod_i p(\mathcal{D}_i)$ *as well as condi-*

tionally independent given a data-point (x, y)*,* $p(\mathcal{D}|y, x) = \prod_i p(\mathcal{D}_i|y, x)$*, then:*

$$p(y|x, \mathcal{D}) = \frac{1}{p(y|x)^{n-1}} \prod_i p(y|x, \mathcal{D}_i) \tag{3.1}$$

Proof.

$$\begin{aligned} p(y|x, \mathcal{D}) &= p(y|x, \mathcal{D}_1, ..., \mathcal{D}_n) \\ &= p(\mathcal{D}_1, ..., \mathcal{D}_n|y, x)\frac{p(y|x)}{p(\mathcal{D})} \\ &= \frac{p(y|x)}{p(\mathcal{D})} \prod_i p(\mathcal{D}_i|y, x) \\ &= \frac{p(y|x)}{p(\mathcal{D})} \prod_i p(y|x, \mathcal{D}_i)\frac{p(\mathcal{D}_i)}{p(y|x)} \\ &= \frac{1}{p(y|x)^{n-1}} \prod_i p(y|x, \mathcal{D}_i) \end{aligned} \tag{3.2}$$

Above $p(y|x)$ is the "prior predictive distribution", and is determined by the chosen model space prior $p(w)$. Note that each distribution need not be identical. □

The BCM model is typically used in the context Gaussian processes (GP) [56, 6, 15], as a way to make inference feasible on large datasets. The reason is that, for a dataset of size n, GP inference has a computational cost that scales as $O(n^3)$ (due to requiring a matrix inversion). By using the BCM, the dataset is instead partitioned into smaller subsets of size k, and inference may be performed separately on each of the n/k subsets. This reduces the cost to $O((n/k) \cdot k^3)$.

The main hurdle in using the BCM is the conditional independence assumption in the theorem above. It can be written $p(\mathcal{D}_i|\mathcal{D}_1, ..., \mathcal{D}_{i-1}, y, x) = p(\mathcal{D}_i|y, x)$. This approximately holds if the other shards $\{\mathcal{D}_1, ..., \mathcal{D}_{i-1}\}$ don't provide any information on the shard $\mathcal{D}_i$. For instance, this may be true if each shard consists of separated clusters of the input space, or if, in classification, each shard contains different classes. In the case of training a GP, this requirement may be satisfied by clustered or sorting the data before it is partitioned.

We argue that, in the FL setting, these assumptions can also be approximately true for heterogeneous data. For instance, one way for clients to have heterogeneous datasets is if they contain data located in different portions of input space (for instance, different digits at each client for MNIST classification). This means the decomposition of (3.2) is a useful

approximation for our setting. This approximation will not always be precise, and we will discuss its drawbacks and methods for remedying them in the following chapter.

Assuming that each client is able to provide some approximation to its local predictive posterior $p(y|x, \mathcal{D}_i)$, Equation (3.2) can be interpreted as an FL aggregation technique. Below, we don't restrict ourselves to the GP setting, and treat $p(y|x, \mathcal{D}_i)$ as a Bayesian predictive posterior obtained by any method. To proceed further we must make some assumptions on the form of the predictions.

3.1.1 Aggregation for Classification

For classification: y is discrete. The product in (3.2) can be computed directly, i.e., for each value of $y \in \{c_1, ..., c_K\}$ (where c_j is the jth class label):

$$p(y = c_j|x, \mathcal{D}) = \frac{1}{p(y = c_j|x)^{n-1}} \prod_i p(y = c_j|x, \mathcal{D}_i) \tag{3.3}$$

The prior and posterior distributions may be approximated from samples in (3.3).

We can interpret this in terms of uncertainties. First, rewriting the formula as:

$$p(y|x, \mathcal{D}) = p(y|x) \prod_i \frac{p(y|x, \mathcal{D}_i)}{p(y|x)}$$

Each client contributes a factor of $\frac{p(y|x,\mathcal{D}_i)}{p(y|x)}$ (the quotient of the posterior and prior in predictive space). If client i doesn't learn much, and has little data (has high uncertainty), its local posterior will be closer to the prior. Thus the factor $\frac{p(y|x,\mathcal{D}_i)}{p(y|x)} \approx 1$ for each y. This means the factor does not contribute much to the overall prediction.

3.1.2 Aggregation for Regression

For a regression task, suppose $y \in \mathbb{R}^d$. In this case, we can approximate the local predictive posteriors as (multivariate) Gaussians: $p(y|x, \mathcal{D}_i) = \mathcal{N}(\mu_i, \Sigma_i)$ (with μ_i, Σ_i depending on x). We similarly approximate the prior predictive distribution $p(y|x) = \mathcal{N}(\mu_p, \Sigma_p)$.

Since the aggregation formula (3.2) multiplies or divides these densities, the global

predictive posterior will also be a Gaussian with some mean μ_g and covariance Σ_g:

$$\Sigma_g = \left(\sum_i \Sigma_i^{-1} - (n-1)\Sigma_p^{-1} \right)^{-1} \tag{3.4}$$

$$\mu_g = \Sigma_g \left(\sum_i \Sigma_i^{-1}\mu_i - (n-1)\Sigma_p^{-1}\mu_p \right) \tag{3.5}$$

The required means and covariances in these formulas may be estimated given samples from each predictive distribution, which in turn may be obtained from samples of the model space posterior $p(\theta|\mathcal{D}_i)$ using any MCMC method.

Note that this aggregation formula has an intuitive interpretation. Suppose we are in the one-dimensional setting, where $\Sigma_i = \sigma_i^2$ is the variance of the output from the client i. Further suppose that we have selected a prior with mean $\mu_p = 0$ and high uncertainty $\Sigma_p = \sigma_p^2 \gg \sigma_i^2$ (which are reasonable settings for an uninformative prior) so that $(\sigma^2)_p^{-1} \approx 0$. Then the aggregation formulas in (3.4) and (3.5) approximately yield the weighted sum:

$$\mu_g \approx \sum_i r_i \mu_i, \quad r_i = \frac{(\sigma^2)_i^{-1}}{\sum_i (\sigma^2)_i^{-1}} \tag{3.6}$$

The weight r_i characterizes the uncertainty client i has in its prediction at input x. A client with high uncertainty would have a correspondingly low weight, and therefore less influence on the overall (mean) prediction. This is a helpful feature in settings with heterogeneous data. In these settings, a client dataset $\mathcal{D}_i$ may not contain any data resembling, or close to some query input x, and we wouldn't like the global prediction at x to be influenced by such clients. Instead, the aggregated mean will be most influenced by clients with high certainty.

Justification for Gaussian Approximation

Since we aren't restricting ourselves to the GP setting, the idea that the predictive posterior $p(y|x, \mathcal{D}_i)$ may be approximated as a Gaussian needs some justification.

Generally speaking, approximating a distribution by a Gaussian is reasonable when the distribution is unimodal and we assume a loss based on the squared distance to a unique target value. Recall that the predictive posterior $p(y|x, \mathcal{D}_i)$ is a distribution over output values y. In supervised regression, we typically assume that there is a single target value y^* and we often seek to minimize the squared error $(y - y^*)^2$. Similarly, under

Table 3.1: Average test accuracies (± standard error) on classification datasets for $h = 0.3$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test).

		$h = 0.3$ heterogeneity				
Method	Rounds	MNIST	Fashion MNIST	EMNIST	CIFAR10	CIFAR100
FedAvg	≥ 5	95.27 ± 0.05$^{\Downarrow}$	79.14 ± 1.12$^{\Downarrow}$	84.02 ± 0.05$^{\Downarrow}$	40.49 ± 0.33$^{\Downarrow}$	9.74 ± 0.12$^{\Downarrow}$
FedPA	≥ 5	91.21 ± 0.08$^{\Downarrow}$	78.32 ± 0.45$^{\Downarrow}$	71.11 ± 0.39$^{\Downarrow}$	43.09 ± 0.23$^{\Downarrow}$	8.04 ± 0.11$^{\Downarrow}$
FedAvg	1	73.74 ± 1.59$^{\Downarrow}$	37.52 ± 2.28$^{\Downarrow}$	67.68 ± 0.80$^{\Downarrow}$	38.18 ± 0.54$^{\Downarrow}$	7.90 ± 0.18$^{\Downarrow}$
FedPA	1	87.36 ± 0.71$^{\Downarrow}$	70.16 ± 1.68$^{\Downarrow}$	35.44 ± 1.17$^{\Downarrow}$	23.19 ± 1.20$^{\Downarrow}$	5.05 ± 0.36$^{\Downarrow}$
EP MCMC	1	90.99 ± 0.37$^{\Downarrow}$	71.75 ± 1.40$^{\Downarrow}$	48.95 ± 1.05$^{\Downarrow}$	21.65 ± 1.39$^{\Downarrow}$	12.14 ± 0.25$^{\Downarrow}$
FedProx	1	84.11 ± 1.36$^{\Downarrow}$	69.63 ± 1.34$^{\Downarrow}$	42.50 ± 1.02$^{\Downarrow}$	10.08 ± 0.05$^{\Downarrow}$	1.62 ± 0.11$^{\Downarrow}$
AdaptFL	1	91.60 ± 0.34$^{\Downarrow}$	74.48 ± 0.87$^{\Downarrow}$	41.23 ± 1.33$^{\Downarrow}$	13.64 ± 0.96$^{\Downarrow}$	5.34 ± 0.18$^{\Downarrow}$
FedBE	1	91.22 ± 0.42$^{\Downarrow}$	80.39 ± 0.92$^{\Downarrow}$	76.84 ± 0.28$^{\Downarrow}$	42.01 ± 0.39$^{\Downarrow}$	10.22 ± 0.14$^{\Downarrow}$
Oneshot FL	1	93.70 ± 0.09	83.72 ± 0.13$^{\downarrow}$	82.06 ± 0.06$^{\downarrow}$	43.63 ± 0.25$^{\Downarrow}$	10.68 ± 0.11$^{\Downarrow}$
FedKT	1	94.01 ± 0.04	84.03 ± 0.10$^{\Downarrow}$	83.52 ± 0.10$^{\Downarrow}$	42.24 ± 0.24$^{\Downarrow}$	9.91 ± 0.15$^{\Downarrow}$
BCM (**ours**)	1	**95.68** ± 0.04$^{\Uparrow}$	**85.60** ± 0.06$^{\Uparrow}$	**86.18** ± 0.14$^{\Uparrow}$	**62.42** ± 0.14$^{\Uparrow}$	**25.04** ± 0.13$^{\Uparrow}$
D BCM (**ours**)	1	95.55 ± 0.04	85.25 ± 0.08	85.44 ± 0.14	61.79 ± 0.19	23.96 ± 0.17

suitable conditions, Bayesian consistency [47] ensures that the expectation of the predictive posterior will converge to the target value y^* in probability (i.e., $E_{p(y|x,\mathcal{D}_i)}[y] \to y^*$). In the case of a Gaussian predictive posterior, the probability that a prediction y is correct is proportional to the exponential of the squared distance to the expectation (i.e., $p(y|x, \mathcal{D}_i) \propto \exp((y - E_{p(y|x,\mathcal{D})}[y])^2)$. Hence, the assumption of a Gaussian predictive posterior is in line with the assumption of a unique target y^* and the minimum squared error in supervised regression.

It's important to note that this is different from assuming that the model space posterior $p(\theta|\mathcal{D}_i)$ can be approximated as a Gaussian. The latter is not as reasonable since there are typically many equivalent models (due to symmetries) that can generate the same data $\mathcal{D}_i$. For instance, if we consider the space of neural networks with fully connected layers, it is well known that hidden nodes can be interchanged to obtain symmetrically equivalent models [49]. Hence the model posterior $p(\theta|\mathcal{D}_i)$ is typically multimodal and far from Gaussian.

Table 3.2: Average test mean squared error (± standard error) on regression datasets, based on 10 seeds. **Lower is better**. Multi-round methods are written above the line, while methods run for a single round of communication are written below. The best technique over 1 round is bolded. $\Uparrow$ / $\Downarrow$: higher/lower mean squared error with $p < 1\%$, $\uparrow$ / $\downarrow$: lower/higher mean squared error with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test).

Method	Rounds	Air Quality	Bike	Wine Quality	Real Estate	Forest Fire
FedAvg	5	6.128 ± 0.074	0.293 ± 0.018$^{\downarrow}$	2.815 ± 0.018$^{\Uparrow}$	0.360 ± 0.008$^{\Uparrow}$	2.791 ± 0.051
FedPA	5	9.163 ± 0.155$^{\Uparrow}$	1.561 ± 0.136$^{\Uparrow}$	3.077 ± 0.144$^{\Uparrow}$	0.592 ± 0.009$^{\Uparrow}$	2.837 ± 0.043
FedAvg	1	6.344 ± 0.087	0.337 ± 0.026	2.821 ± 0.019$^{\Uparrow}$	0.354 ± 0.006$^{\Uparrow}$	2.789 ± 0.054
FedPA	1	6.556 ± 0.129$^{\uparrow}$	1.663 ± 0.035$^{\Uparrow}$	2.870 ± 0.028$^{\Uparrow}$	0.454 ± 0.009$^{\Uparrow}$	2.792 ± 0.051
EP MCMC	1	6.297 ± 0.036$^{\uparrow}$	0.350 ± 0.016$^{\uparrow}$	2.488 ± 0.044$^{\Uparrow}$	0.327 ± 0.007	2.803 ± 0.045
FedProx	1	6.210 ± 0.064	**0.200** ± 0.014$^{\Downarrow}$	2.479 ± 0.036	1.292 ± 0.178$^{\Uparrow}$	3.129 ± 0.068$^{\Uparrow}$
AdaptFL	1	6.497 ± 0.118	2.158 ± 0.053$^{\Uparrow}$	3.263 ± 0.029$^{\Uparrow}$	0.405 ± 0.009$^{\Uparrow}$	3.528 ± 0.240$^{\Uparrow}$
FedBE	1	9.263 ± 0.239$^{\Uparrow}$	2.479 ± 0.069$^{\Uparrow}$	3.455 ± 0.016$^{\Uparrow}$	0.845 ± 0.008$^{\Uparrow}$	**2.786** ± 0.038
OneshotFL	1	9.477 ± 0.253$^{\Uparrow}$	2.476 ± 0.070$^{\Uparrow}$	3.452 ± 0.012$^{\Uparrow}$	0.847 ± 0.009$^{\Uparrow}$	2.790 ± 0.064
BCM (**ours**)	1	**6.172** ± 0.037	0.337 ± 0.015$^{\uparrow}$	**2.374** ± 0.038	**0.319** ± 0.006	2.803 ± 0.044
D BCM (**ours**)	1	6.187 ± 0.044	0.330 ± 0.013	2.378 ± 0.039	0.322 ± 0.006	2.815 ± 0.034

3.2 Applying the BCM to Federated Learning

We can apply the above aggregation formulae from the BCM model in a federated algorithm. Namely, if we assume that we are learning a parametric model at each client $p(w|\mathcal{D}_i)$, then samples from this model can be shared among the clients, and used to make predictions. The steps of the algorithm would be:

1. Sampling: At each client, use MCMC sampling to generate samples according to the local posteriors $\{\theta\}_i \sim p(\theta|\mathcal{D}_i)$.

2. Communication: each client communicates the samples to each other.

3. Approximate local posteriors: At prediction time, use each set of samples to produce predictions according to $p(y|x, \mathcal{D}_i)$ (using Equation (2.7)). This involves running inference for each sample, then averaging the probabilistic predictions.

4. Aggregation: Using Eq. (3.5) (for regression tasks) or (3.3) (for classification tasks), aggregate individual predictions into the global predictive posterior $p(y|x, \mathcal{D})$.

Algorithm 1 D BCM

Input: Client datasets $\mathcal{D}_i$, sampler MCMC_sample
Output: Model θ^*

for each client i **do**
  $\{\theta\}_i = \text{MCMC\_sample}(\mathcal{D}_i)$ {//step 1}
  Communicate $\{\theta\}_i$ to server {//step 2}
end for
At Server:
$\hat{p}(y|x, \mathcal{D}_i) = \frac{1}{|\{\theta\}_i|} \sum_{\theta \in \{\theta\}_i} p(y|x, \theta)$ {//step 3}
$\hat{p}(y|x, \mathcal{D}) = \text{Aggregate}(\hat{p}(y|x, \mathcal{D}_i))$ {//step 4, Eq 3.3, 3.5}
$\theta^* = \text{Distill}(\hat{p}(y|x, \mathcal{D}))$ {//step 5}
return θ^*

Since the aggregation is done in the predictive space, the algorithm essentially builds an ensemble of models to predict according to the global posterior. If there are n clients, each of which draws M samples from the local posterior, the ensemble has $O(Mn)$ models.

The primary issues with this straight forward algorithm are:

- Communication cost: Each of the n clients needs to broadcast its M model samples to each other client. In other words, a single client will need to send $M \cdot (n-1)$ models. A total of $O(Mn^2)$ models will need to be communicated overall, which can be expensive as the number of clients grows.
- Computational cost of prediction: Inference for a single test point requires predicting with $M \cdot n$ models. If each model is large or computationally expensive (such as a neural network), this may be prohibitively expensive for a single client, both in terms of memory and compute. Typical practice in FL is to assume that clients are relatively lightweight in terms of computational power.
- Privacy concerns: each client will be able to access the trained models from other clients. This means that trust needs to be extended between each client. In contrast, typical practice in FL is that each client's model should only be shared with a central server. In the latter case, trust only needs to be extended to the server, rather than each individual client.

Therefore, although M can be small (for instance $M \leq 6$ in our experiments), this algorithm is only suited for applications where there are a limited number of clients, and

where each has the computational capabilities to store the ensemble (for instance: the cross-silo setting).

To make the algorithm more practical for Federated learning, we propose two modifications:

1. In step 2 (communication): instead of sharing client models with each other, they are shared with a central server
2. We append an additional step - step 5 Distillation: The server forms the ensemble to approximate $p(y|x, \mathcal{D})$ according to equations 3.3 and 3.5, and uses **knowledge distillation** to compress it into a single model using an unlabeled distillation dataset. This model is sent back to clients

We call this variant "distilled BCM" or D BCM. The modifications aid in the issues raised before:

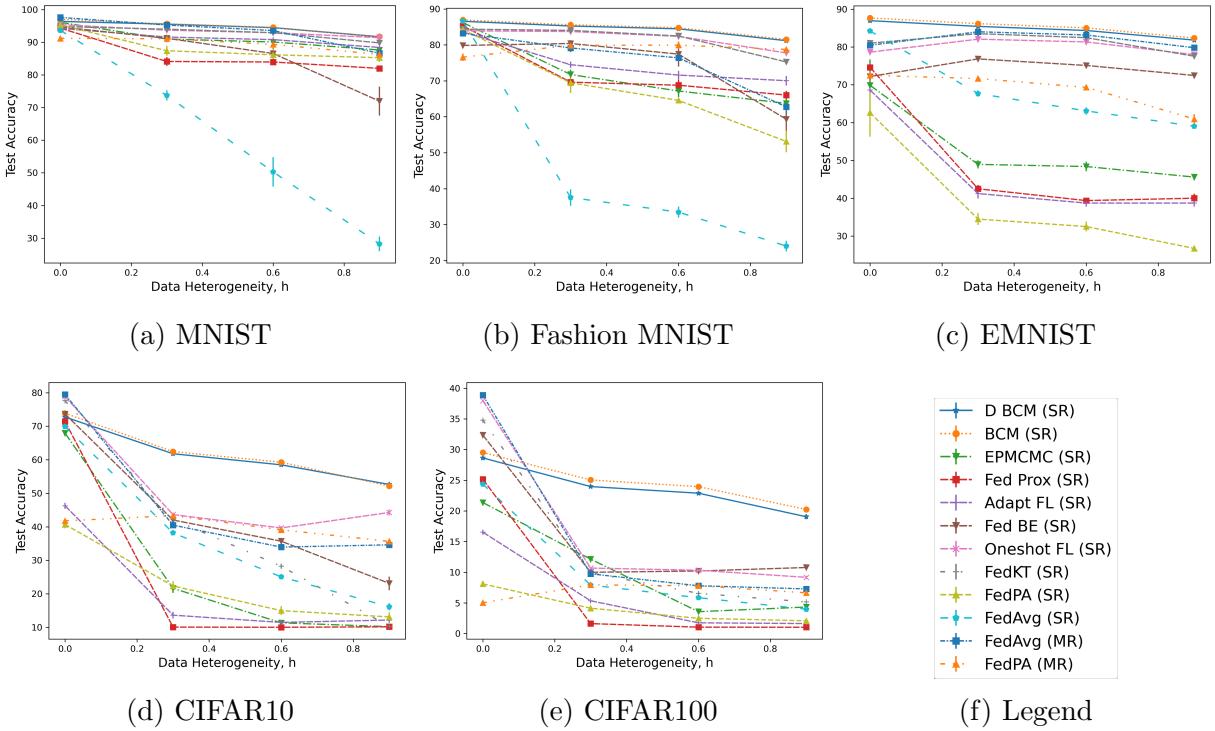
- Each client only needs to communicate M models with the server. This means the overall communication cost is $O(Mn)$
- At inference each client only needs to predict with the single, distilled, model. The more expensive ensemble prediction is instead shifted to the server (during knowledge distillation).
- Each client only needs to directly trust and share models with the central server.

The procedure is summarized in Algorithm 1.

Note that the algorithm works with a single round of communication. This is because the sampling from the local posteriors can be done individually by each client, and the aggregation step computes $p(y|x, \mathcal{D})$ in one step. This feature of the method alleviates the heavy cost of communication that multiple rounds bring, particularly, since a few samples suffice.

3.3 Experiments

We verify the effectiveness of our method, both with and without distillation, by training it on multiple regression and classification datasets, and comparing the performance to a



(a) MNIST (b) Fashion MNIST (c) EMNIST

(d) CIFAR10 (e) CIFAR100 (f) Legend

Figure 3.1: Test accuracies on the classification datasets with increasing data heterogeneity (tested with parameter settings of h=$\{0.0, 0.3, 0.6, 0.9\}$). SR = single round method, MR= multi-round method. Averages and standard error over 10 seeds are reported.

selection of baseline algorithms. All tests were run across 5 clients. The code for experiments, as well as details for the randomized seeds, and compute infrastructure are given in the supplementary material.

Datasets were split into a global train and test set. The train set was further divided among clients as well as a distillation set (20% of the training set). All reported results correspond to the test set performance.

3.3.1 Classification Datasets

The method was evaluated for classification on the following datasets: MNIST [34], Fashion MNIST [61], EMNIST [9] (using a split with 62 classes), CIFAR10 [33], and CIFAR100 [33]. The way in which each of these datasets are distributed among clients is controlled by a "heterogeneity parameter", h. For $h = 0$, the data is split uniformly among the clients,

and is thus homogeneously distributed. On the other hand, for $h = 1$ the data is sorted by class before being split among clients. This means that each client observes data from different classes with little overlap. In this sense, $h = 1$ represents the "fully heterogeneous" setting. For $0 < h < 1$, data from the above extremes is mixed: a fraction of size h of the homogeneous data shard is replaced with the fully heterogeneous data for each client. In the heterogeneous setting ($h > 0$), the aggregation technique plays a key role in constructing a good global model, since each local model performs poorly given the imbalanced training data.

3.3.2 Regression Datasets

The regression datasets used for evaluation include: the "wine quality" [10], "air quality" [14], "forest fire" [11], "real estate" [62], and "bike rental" [20] datasets from the UCI repository [18]. These datasets were sorted according to certain features (such as the date, for "airquality"), then split among clients, to simulate heterogeneous data.

3.3.3 Models

A two-layer fully connected network with 100 hidden units was used for MNIST, Fashion MNIST, EMNIST, as well as the regression datasets. For the CIFAR10 and CIFAR100 datasets a Convolutional Neural network was used with 3 convolution layers, each followed by "Max Pooling" layers, with a single fully connected layer at the end. For all networks, the ReLU activation function was used between layers. For classification, the output was the predictive distribution over classes, while for regression the output was the mean of the predictive distribution. A Gaussian prior is assumed over the network parameters, $p(\theta) = \mathcal{N}(0, \sigma^2 I)$, with $\sigma = \texttt{5e4}$.

3.3.4 Baselines

The Federated techniques compared include: Federated Averaging (FedAvg) [40], Federated Posterior Averaging (FedPA) [1], Embarrassingly Parallel MCMC (EP MCMC) [46], FedProx [36], Adaptive FL [51], Federated Bayesian Ensemble (FedBE) [7], One Shot Federated Learning (OneshotFL) [23], and Federated Learning via Knowledge Transfer (FedKT) [35] (which is for classification only).

Hyperparameters for all methods and datasets were tuned with a grid search. The search was performed primarily over optimization hyperparameters (such as learning rate,

and the optimization algorithm) and sampler hyperparameters (such as temperature, and sample frequency). Further details are included in the supplementary material.

In the case of FedAvg, FedProx, FedBE, OneshotFL and FedKT, either SGD with momentum or the Adam optimizer [32] was used for local optimization (as selected by a grid search). For the rest of the methods, including our own, since they require MCMC sampling, cyclic stochastic gradient Hamiltonian Monte-Carlo (cSGHMC) [64] was used. For EP MCMC, the algorithm was computationally intractable for neural network models due to the calculation of the inverse of a covariance matrix over parameters. Thus a diagonal covariance matrix was assumed (which corresponds to the assumption that the local posteriors are approximated by an axis-aligned Gaussian). All methods were run for a single round. For comparison to more typical multi-round methods, FedAvg and FedPA are also run for multiple rounds.

3.3.5 Training Details

For MNIST, Fashion MNIST and EMNIST, the training was run for 25 epochs per client overall (split into 5 rounds for FedAvg and FedPA, while only run in a single round for the rest). For CIFAR10 and CIFAR100, training was run for 50 epochs per client (split into 10 rounds for multi-round methods). For the regression datasets, a total of 20 epochs are used for all datasets except "air quality", which used 100 epochs (in both cases, divided into 5 rounds for multi-round methods). The methods involving sampling used a maximum of 6 samples for all experiments.

3.3.6 Classification Results

The results on the classification tasks with $h = 0.3$ (a heterogeneous setting), averaged over 10 seeds, are recorded in Table 3.1, along with an indicator of statistical significance. For all datasets, we can identify that among both single and multi round methods, BCM performs best. This result is statistically significant for all datasets other than MNIST. These results suggest that despite training for only a single round, our method is competitive with more widely used multi-round techniques.

We also compare the performance of the different algorithms as the level of data heterogeneity (h) is increased. The results are plotted in Figure 3.1. For these experiments, both single round (SR) and multi-round (MR) methods were evaluated. From these results, we can see that in all datasets, our method outperforms the other techniques as the heterogeneity increases. In particular, we observe that as $h > 0$ increases, our method suffers the

least performance loss, and obtains the highest accuracy out of the baselines. Additional details (such as tabulated results) for these experiments are included in the supplementary material.

For the experiments above, D BCM performs slightly below BCM, but outperforms all other single round methods. From this we can observe that the distillation step (and the reduced model complexity of the student) only induces a small performance penalty on the method, and the method still performs competitively.

3.3.7 Regression Results

The results for the regression setting, averaged over 10 seeds, are recorded in Table 3.2, along with an indicator for statistical significance. We can observe that our method performs best outside of the "bike" and "forest fire" dataset. For the former, our method performs second best behind FedProx, while for the latter, the differences are not statistically significant (and most baselines perform similarly).

In this setting, for the "bike" dataset, D BCM outperforms BCM. A possible reason for this is that the reduced model complexity of the student has a regularizing effect, which helps the method to avoid overfitting.

Chapter 4

Calibrating the BCM: β-Predictive Bayes

In the previous chapter we assumed the Bayesian Committee Machine's (BCM) decomposition 3.2 was correct, and used it to approximate the global model. In this chapter, we analyze when and how this approximation fails: namely in that it can produce a poorly calibrated global model (even if, as we observed in the previous chapter, the accuracy may still be high). We propose a method to rectify this issue.

A well-calibrated model is useful in FL since individual client datasets may be small or noisy, and we'd like to quantify our uncertainty in these cases. Furthermore, calibration is a particularly important metric for Bayesian methods since a "well-specified" Bayesian model should be well-calibrated [13]. By well-specified, we require (among other conditions) that the model is flexible enough to capture the "true" generative process of the data, and that the prior has large enough support to include this process. Therefore, a poorly calibrated Bayesian model lets us diagnose issues with the model specification or the approximate inference. If our aggregation technique itself produces calibration error, then we'd be unable to make use of this feature of Bayesian inference.

4.1 Analyzing the Calibration of the BCM

We can analyze the BCM equation in the context of Gaussian process regression, since it allows us to explicitly calculate the predictive mean and variance.

We assume a smooth, isotropic kernel function $k(x, x') = k(||x - x'||)$. We assume a model with Gaussian noise:

$$y = f(x) + \epsilon \tag{4.1}$$
$$\epsilon \sim \mathcal{N}(0, \sigma_o^2) \tag{4.2}$$

Where σ_o^2 is the "observation"/noise variance in the predictions (in other words, the aleatoric uncertainty in our predictions).

On some dataset $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, and with test point x^*, the Gaussian process inference equations predict a mean and variance for $p(y^*|x^*, \mathcal{D}_i) = \mathcal{N}(\mu(x^*), \sigma^2(x^*))$. If we denote $\mathbf{k}_* = [k(x^*, x_1), ..., k(x^*, x_{|\mathcal{D}|})]^\top$ the vector of kernel evaluations between the test point x^* and the points in the dataset $\mathcal{D}$, and $\mathbf{K}$ as the kernel matrix (where $K_{i,j} = k(x_i, x_j)$ for $x_i, x_j \in \mathcal{D}$) then the inference equations are [50]:

$$\mu(x^*) = \mathbf{k}_*^\top (\mathbf{K} + \sigma_o^2 I)^{-1} \mathbf{y} \tag{4.3}$$
$$\sigma^2(x^*) = \sigma_o^2 + k(x^*, x^*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma_o^2 I)^{-1} \mathbf{k}_* \tag{4.4}$$

Suppose the input data-points lie in some bounded region $x_i \in R$. For these inference equations we outline two observations about the predictive variance $\sigma^2(x_*)$:

- **Lemma 1.** *Assume $x^* \in R$. Under some mild conditions on the kernel function, and under the assumption of Gaussian or Laplacian observation noise, as the number of data-points increases $\sigma^2(x^*) \to \sigma_o^2$ (and in addition, the predictive mean converges to the true function value: $\mu(x^*) \to f(x^*)$) [8].*

  This intuitively means that for points close to the training data, our GP model attains the correct mean and variance after seeing enough data.

- **Lemma 2.** *Assume $x^* \notin R$, and is sufficiently far away from all training points such that $k(x^*, x_i) \approx 0$ for all $x_i \in \mathbf{X}$. Then the predictive variance becomes $\sigma^2(x^*) = \sigma_o^2 + k(x^*, x^*)$, which we refer to as the "prior variance" σ_p^2. Also, the predictive mean becomes $\mu(x^*) = 0$.*

Proof. The inference equation 4.3 for the predictive variance reads:

$$\begin{aligned}\sigma^2(x^*) &\approx \sigma_o^2 + k(x^*, x^*) - \mathbf{0}^\top (\mathbf{K} + \sigma_o^2 I)^{-1} \mathbf{0} \\ &\approx \sigma_o^2 + k(x^*, x^*)\end{aligned}$$

Similarly, the predictive mean is $\mu(x^*) \approx \mathbf{0}^\top (\mathbf{K} + \sigma_o^2 I)^{-1} \mathbf{y} = 0$. □

In other words, for a sufficiently distant test point, the predictive variance reverts to a higher "prior" variance $\sigma_p^2 = \sigma_o^2 + k(x^*, x^*)$. This is also expected behaviour for a well-calibrated model, since it reflects the model's higher uncertainty on unobserved data.

Equipped with these observations, we can analyze the case with partitioned data. We assume two idealized partitions for the overall dataset $\mathcal{D}$ into the m client datasets $\mathcal{D}_1, ..., \mathcal{D}_m$:

- **Idealized Homogeneous Partition**: each dataset $\mathcal{D}_i$ maintains the same distribution as the global dataset $\mathcal{D}$. One way of doing this is to randomly subsample the global dataset (without replacement) to form each local dataset.
- **Idealized Heterogeneous Partition**: all points $x_i \in \mathcal{D}_i$ and $x_j \in \mathcal{D}_j$ are separated by a kernel distance $k(x_i, x_j) \approx 0$. In other words, the local datasets form clusters in the input space, when viewed from the lens of the kernel function.

We can now show that the BCM equations underestimate the predictive variance in the case of the idealized homogeneous partition:

Theorem 2. *If data is split among m clients in an idealized homogeneous partition, and $x^* \in R$ is a single test point, then the BCM equations 3.4 underestimate the predictive variance $\sigma^2_{\text{BCM}}(x^*) < \sigma_o^2$ as the size of the data subsets grows ($|\mathcal{D}_i| \to \infty$).*

Proof. According to the BCM aggregation equation for regression:

$$\sigma^{-2}_{\text{BCM}}(x^*) = \sum_i \sigma_i^{-2}(x^*) - (m-1)\sigma_p^{-2}(x^*)$$

Where $\sigma_i^{-2}(x^*)$ is the inverse of the predictive variance output by the GP trained on $\mathcal{D}_i$. As the size of the dataset $\mathcal{D}_i$ increases, from lemma 1 we know that this quantity will converge to σ_o^2. On the other hand $\sigma_p^2 = k(x^*, x^*) + \sigma_o^2$. Combining these facts (in the limit of increasing data points):

$$\begin{aligned}
\sigma_{\text{BCM}}^{-2}(x^*) &= m\sigma_o^{-2} - (m-1)(\sigma_o^2 + k(x^*, x^*))^{-1} \\
&> m\sigma_o^{-2} - (m-1)\sigma_o^{-2} \\
&= \sigma_o^{-2}
\end{aligned}$$

Where the second inequality follows from the fact that $k(x^*, x^*) > 0$ (for a positive kernel function).

This implies $\sigma_{\text{BCM}}^2 < \sigma_o^2$ as claimed.

Furthermore, for a kernel function with a large prior variance $\sigma_p^{-2} \approx 0$ (which is typical in practice when using an uninformative prior), we see that:

$$\begin{aligned}
\sigma_{\text{BCM}}^{-2} &= m\sigma_o^{-2} - (m-1)\sigma_p^{-2} \\
&\approx m\sigma_o^{-2}
\end{aligned}$$

And we end up underestimating the predictive variance by a factor of m, for m clients. □

This means that the predictions provided by BCM are overconfident. On the other hand in the same setting, for the mean prediction, if we assume a large prior variance $\sigma_p^{-2} \approx 0$, a quick calculation shows that we still obtain accurate results. Starting with the simplified equation for the BCM predictive mean 3.6 and applying lemma 1:

$$\begin{aligned}
\mu_{\text{BCM}}(x^*) &= \sum_i \sigma_i^{-2}\mu_i / (\sum_i \sigma_i^{-2}) \\
&\approx \sum_i \sigma_o^{-2} f(x^*) / (\sum_i \sigma_o^{-2}) \\
&= f(x^*)
\end{aligned}$$

In other words, the predictive mean still recovers the true mean function at x^*.

This sheds light on the results of the previous chapter: the BCM model (or its distilled form) performed well for homogeneous data because only the mean predictions were evaluated (through the Mean Squared Error for regression, and accuracy for classification) not the calibration.

We can also check to see that the BCM model produces a calibrated predictive variance in the case of the idealized heterogeneous data setting, in line with the assumptions in the previous chapter:

Theorem 3. *If data is split among m clients in an idealized heterogeneous partition, and x^* is a single test point in the close vicinity of some data subset $\mathcal{D}_k$, then the BCM equations 3.4 correctly estimate the predictive variance $\sigma^2_{\text{BCM}}(x^*) = \sigma_o^2$ as the size of the data subsets grows ($|\mathcal{D}_i| \to \infty$).*

Proof. The BCM aggregation equation for regression is:

$$\sigma^{-2}_{\text{BCM}}(x^*) = \sum_i \sigma_i^{-2}(x^*) - (m-1)\sigma_p^{-2}(x^*)$$

Since x^* is in the vicinity of some data subset $\mathcal{D}_k$, we can apply lemma 1 to get $\sigma_k^{-2}(x^*) = \sigma_o^{-2}$. On the other hand, since the data subsets are split up heterogeneously, the test point x^* is far from the other data sub-sets such that we can apply lemma 2 to obtain $\sigma_i^{-2}(x^*) = \sigma_p^2(x^*)$ for all $i \neq k$. Plugging these into the above expression yields:

$$\begin{aligned}
\sigma^{-2}_{\text{BCM}}(x^*) &= \sigma_k^{-2}(x^*) + \sum_{i \neq k} \sigma_i^{-2}(x^*) - (m-1)\sigma_p^{-2}(x^*) \\
&= \sigma_o^{-2} + (m-1)\sigma_p^{-2}(x^*) - (m-1)\sigma_p^{-2}(x^*) \\
&= \sigma_o^{-2}
\end{aligned}$$

□

4.1.1 Analyzing the Predictive Mixture Model

From the above we see that the BCM model produces overconfident estimates for the (idealized) homogeneous setting. An alternate model which produces well-calibrated estimates in this setting is the **predictive mixture model**. This model aggregates the local posteriors as:

$$\sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|} p(y|x, \mathcal{D}_i) \tag{4.5}$$

ie. as a mixture, as opposed to the product of the BCM.

For GP regression, when each $p(y|x, \mathcal{D}_i) = \mathcal{N}(\mu_i(x), \sigma_i^2(x))$ is Gaussian, this model produces a Gaussian mixture as its predictive distribution. In general, this won't be Gaussian, but we can approximate it with an overall Gaussian prediction $\mathcal{N}(\mu_{\text{mix}}(x), \sigma_{\text{mix}}^2(x))$ matching the mean and variance of the Gaussian mixture. This results in:

$$\mu_{\text{mix}}(x) = \sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \mu_i(x) \tag{4.6}$$

$$\sigma_{\text{mix}}^2(x) = \sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|} (\sigma_i^2(x) + \mu_i^2(x)) - \mu_{\text{mix}}^2(x) \tag{4.7}$$

Which follows from the fact the moments of a mixture are equal to the mixtures of the moments.

Unlike the BCM model, for homogeneous data, the mixture model is well-calibrated:

Theorem 4. *If data is split among m clients in an idealized homogeneous partition, and $x^* \in R$ is a single test point in the bounded region of the data, then the predictive mixture equations 4.6 correctly estimate the predictive variance $\sigma_{\text{mix}}^2(x^*) = \sigma_o^2$ as the size of the data subsets grows ($|\mathcal{D}_i| \to \infty$).*

Proof. In this setting, appealing to 1, each local predictor outputs $\mathcal{N}(\mu_i(x^*) = f(x^*), \sigma_i^2(x^*) = \sigma_o^2)$

$$\begin{aligned}
\sigma_{\text{mix}}^2 &= \sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|} (\sigma_i^2(x^*) + \mu_i^2(x^*)) - \mu_{\text{mix}}^2(x^*) \\
&= \sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|} (\sigma_o^2 + f^2(x^*)) - (\sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|} f(x^*))^2 \\
&= \sigma_o^2 + f^2(x^*) - f^2(x^*) \\
&= \sigma_o^2
\end{aligned}$$

□

However, the drawback of the mixture model is that for the idealized heterogeneous partition, it overestimates the predictive uncertainty:

Theorem 5. *If data is split among m clients in an idealized heterogeneous partition, and x^* is a single test point in the close vicinity of some data subset $\mathcal{D}_k$, then the predictive mixture equations 4.6 overestimate the predictive variance $\sigma^2_{\text{mix}}(x^*) > \sigma^2_o$ as the size of the data subsets grows ($|\mathcal{D}_i| \to \infty$).*

Proof. The mixture models predictive variance is:

$$\sigma^2_{\text{mix}}(x^*) = \sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|}(\sigma_i^2(x^*) + \mu_i^2(x^*)) - \mu^2_{\text{mix}}(x^*)$$

Since x^* lies in the vicinity of $\mathcal{D}_k$, we have (applying lemma 1) that $\sigma_k^2(x^*) = \sigma_o^2$ (and $\mu_k(x^*) = f(x^*)$), while from lemma 2 we know $\sigma_i^2(x^*) = \sigma_p^2(x^*) > \sigma_o^2$ (and $\mu_i(x^*) = 0$) for all $i \neq k$.

$$\begin{aligned}
\sigma^2_{\text{mix}}(x^*) &= \frac{|\mathcal{D}_k|}{|\mathcal{D}|}(\sigma_o^2 + f^2(x^*)) + \sum_{i \neq k} \frac{|\mathcal{D}_i|}{|\mathcal{D}|}(\sigma_p^2(x^*)) - \mu^2_{\text{mix}}(x^*) \\
&= \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\sigma_o^2 + \sigma_p^2(x^*)\Big(1 - \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\Big) + \frac{|\mathcal{D}_k|}{|\mathcal{D}|}f^2(x^*) - \frac{|\mathcal{D}_k|}{|\mathcal{D}|}f^2(x^*) \\
&= \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\sigma_o^2 + \sigma_p^2(x^*)\Big(1 - \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\Big) \\
&= \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\sigma_o^2 + (\sigma_o^2 + k(x^*, x^*))\Big(1 - \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\Big) \\
&= \sigma_o^2 + k(x^*, x^*)\Big(1 - \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\Big) \\
&> \sigma_o^2
\end{aligned}$$

In other words we overestimate σ_o^2 by a constant of $k(x^*, x^*)\big(1 - \frac{|\mathcal{D}_k|}{|\mathcal{D}|}\big)$. □

We note that in this case, the predictive mean of the model in this case is also not accurate, since $\mu_{\text{mix}}(x^*) = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}f(x^*) \neq f(x^*)$. However, this result is less relevant than the variance for realistic data partitions, since, revisiting the proof of lemma 2, if $k(x^*, x^*) = \epsilon < 1$, the variance goes to prior variance $\sigma_p^2(x^*)$ with error $O(\epsilon^2)$, while the mean goes to 0 with approximate error $O(\epsilon)$.

4.1.2 Heuristic Argument for Classification

The preceding analysis may be extended to classification (or more general) models under the following assumptions. Below, we denote the true underlying predictive model (with correct aleatoric uncertainty) as $p_T(y|x)$, and the prior predictive model as $p_P(y|x)$.

1. For x^* in the vicinity of data subset $\mathcal{D}_i$, and assuming all data subsets are large enough, the local predictive model converges to the true model, i.e. $p(y|x, \mathcal{D}_i) = p_T(y|x)$
2. For x^* far away from the data subset $\mathcal{D}_i$, the local predictor outputs the prior model: $p(y|x, \mathcal{D}_i) = p_P(y|x)$
3. The prior model $p_P(y|x)$ has higher variance, uncertainty, and/or entropy than the true model $p_T(y|x)$.

All these are general desiderata for a well calibrated model. The first bullet ensures that for in-domain points, the model is accurate, and the second bullet ensures that for out-of-domain points, the model reverts to the predictions it would have made in the absence of data (the prior predictions). The final bullet-point ensures that the prior predictions are sufficiently uncertain.

Although some of these points may not be satisfied by some approximate models, they are the ideal goal for a well-calibrated model. For instance, in classification with neural networks using a softmax layer, item 2 may not be true, and it is the goal of multiple methods to correct this defect [44]. This is why, for analytically simple models such as GP regression, these assumptions hold.

Therefore, we can sketch an argument for why we expect the previous analysis to extend to more general models, in the ideal well-calibrated case. For simplicity we assume $p_P(y|x) \propto 1$, a uniform distribution (in the case of a bounded domain on y).

- **BCM model**: For homogeneous data, each local predictive distribution is correct on the test point, so that $p(y|x, \mathcal{D}_i) = p_T(y|x)$. By multiplying together these m distributions, we effectively obtain an aggregated predictive distribution: $p_{\text{BCM}}(y|x, \mathcal{D}) \propto p_T^m(y|x)$, which will in general be sharper than the desired distribution $p_T(y|x)$. We therefore see how the BCM models tends to produce overconfident predictions in the homogeneous case.

On the other hand, for heterogeneous data, only one of the factors in the product distribution converges to $p_T(y|x)$, while the others still match the uniform prior $p_P(y|x)$. In this case, the product ends up $p_{\text{BCM}}(y|x, \mathcal{D}) \propto p_T(y|x)$, producing an accurate model.

- **Mixture model**: For heterogeneous data, one of the local predictive distributions is correct on the test point, with $p(y|x, \mathcal{D}_k) = p_T(y|x)$, while the others match the flat prior $p_P(y|x)$. In the mixture model, the overall prediction then mixes between the correct distribution, and the wider/more uncertain prior distribution: $p_{\text{mix}}(y|x, \mathcal{D}) = (|\mathcal{D}_k|/|\mathcal{D}|)p_T(y|x) + (1 - |\mathcal{D}_k|/|\mathcal{D}|)p_P(y|x)$. This softens the overall distribution, producing a more uncertain predictive distribution than the correct one.

  On the other hand, for homogeneous data, each predictive distribution is identical, and equal to the true distribution, so that the mixture returns the correct distribution $p_{\text{mix}}(y|x, \mathcal{D}) = \sum_i (|\mathcal{D}_i|/|\mathcal{D}|)p_T(y|x) = p_T(y|x)$.

We summarize our observations for the analysis in table 4.1.

Table 4.1: Summary of Predictive Variance Analysis

	Heterogeneous Data	**Homogeneous Data**
BCM/Product Model	Accurate variance	Underestimates variance
Mixture Model	Overestimates variance	Accurate variance

4.2 Calibrating the Aggregated Model

The preceding analysis suggests a way to combine the predictive mixture model 4.5, and the BCM (henceforth referred to as the "product model", to contrast with the mixture model) 3.2.

Namely, to obtain the correct calibration in the predictive model, we should interpolate between the mixture, which is accurate for homogeneous data, and the product, which is accurate for heterogeneous data.

For interpolation parameter β, the model, which we refer to as β-Predictive Bayes or β-PredBayes is:

$$p_\beta(y|x,\mathcal{D}) = \Big(\frac{1}{p(y|x)^{n-1}}\prod_i p(y|x,\mathcal{D}_i)\Big)^\beta \Big(\sum_i \frac{|\mathcal{D}_i|}{|\mathcal{D}|} p(y|x,\mathcal{D}_i)\Big)^{1-\beta} \tag{4.8}$$

So that a case of $\beta = 0.0$ corresponds to the mixture model, and $\beta = 1.0$ corresponds to the product model.

In the case of regression, assuming Gaussian outputs for each local predictive distribution $p(y|x,\mathcal{D}_i)$, and using the Gaussian approximation for the mixture distribution 4.6, it can be shown that the aggregated predictive distribution is approximately a Gaussian $p_\beta(y|x,\mathcal{D}) = \mathcal{N}(\mu_\beta(x), \sigma^2_\beta(x))$ with:

$$\sigma^{-2}_\beta(x) = \beta \cdot \sigma^{-2}_{\text{prod}} + (1-\beta)\cdot \sigma^{-2}_{\text{mix}}(x) \tag{4.9}$$

$$\mu_\beta(x) = \sigma^2_\beta(x)\big(\beta \cdot \sigma^{-2}_{\text{prod}}(x)\mu_{\text{prod}}(x) + (1-\beta)\cdot \sigma^{-2}_{\text{mix}}(x)\mu_{\text{mix}}(x)\big) \tag{4.10}$$

In other words the inverse-variance (or precision) interpolates between that of the product and mixture distributions.

We learn the setting of the β by choosing it to minimize the negative log-likelihood of a dataset on the server, $\mathcal{U}$ using a gradient-based optimizer. In other words:

$$\beta^* = \operatorname*{argmin}_\beta \sum_{(x,y)\in\mathcal{U}} -\log p_\beta(y|x,\mathcal{D}) \tag{4.11}$$

In the case of regression, the Gaussian negative log-likelihood can be used for training (by applying 4.9 as the approximation for $p_\beta(y|x,\mathcal{D})$).

This is feasible in our setting because we already make use of a distillation dataset at the server. We can simply use this distillation dataset to automatically tune the value of β, without needing to explicitly know where we are along the homogeneous-heterogeneous partition spectrum. Since we are only tuning a single scalar parameter, a small dataset may suffice.

Besides the updated aggregation step and learning β, the algorithm remains unchanged from 1. The overall steps for β-predBayes are presented in algorithm 2.

Algorithm 2 Distilled β-PredBayes

Input: Client datasets $\mathcal{D}_i$, sampler MCMC_sample
Output: Model θ^*

for each client i **do**
  $\{\theta\}_i = \text{MCMC\_sample}(\mathcal{D}_i)$ {//step 1}
  Communicate $\{\theta\}_i$ to server {//step 2}
end for
At Server:
$\hat{p}(y|x, \mathcal{D}_i) = \frac{1}{|\{\theta\}_i|} \sum_{\theta \in \{\theta\}_i} p(y|x, \theta)$ {//step 3}
$\hat{p}_\beta(y|x, \mathcal{D}) = \text{Aggregate}(\hat{p}(y|x, \mathcal{D}_i))$ {//step 4, Eq 4.8}
$\beta^* = \text{argmin}_\beta \sum - \log \hat{p}_\beta(y|x, \mathcal{D})$ {//step 5, tune β}
$\theta^* = \text{Distill}(\hat{p}_{\beta^*}(y|x, \mathcal{D}))$ {//step 6}
return θ^*

4.3 Experiments

We evaluate β-predBayes on the same regression and classification datasets as chapter 3. All tests once again used 5 clients, and identical splits for the train, test and distillation sets. We use the same method to partition the dataset for the heterogeneous and homogeneous cases (sorting the data by class, for classification, or by an input feature for regression, before splitting it among clients).

Since the objective of the experiments is to evaluate the calibration of the predictions (not just their accuracy), we use the metric of negative-log-likelihood (NLL) on the test set [24]. For classification, we may also use the expected calibration error (ECE), which measures the difference between the confidence (probabilities) assigned to predicted classes, and the probability they are predicted correctly.

4.3.1 Classification Results

We evaluate β-predBayes, in both its distilled (listed as D β-predBayes) and non-distilled forms, along with the product (BCM) and mixture models. We also evaluate most of the baselines in chapter 3, all in a single round of communication. One exception is Federated Learning via Knowledge Transfer (FedKT) [35] because this method's aggregation uses majority voting for the global prediction, meaning that it doesn't output a probability distribution over classes, which would be needed to evaluate calibration.

The results for the negative log likelihood over different settings of the heterogeneity are plotted in figure 4.1, while expected calibration error for these same classes are shown in 4.2.

For the negative log-likelihood, the β-PredBayes model (and its distilled variant) perform best (least NLL), followed by the Mixture model, on the tested datasets. As heterogeneity increases, the NLL loss generally increases for other methods, while it largely remains stable for β PredBayes

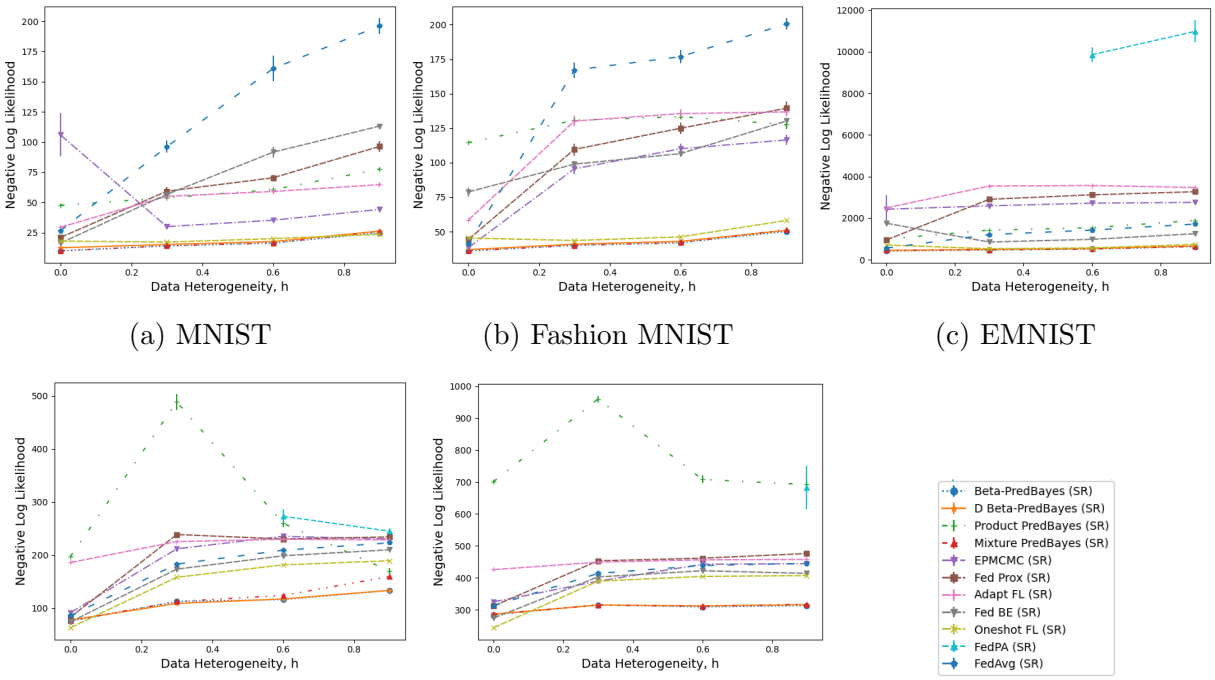
For the expected calibration error, as heterogeneity increases, the metric jumps more erratically for some methods (which may be due to the sensitivity of the metric to certain settings like the number of bins used). But the overall trend is still that β-PredBayes perfoms best (with lowest ECE), followed by its distilled variant, followed by the mixture model. Its worth noting that, for ECE, β-PredBayes outperforms the mixture model for high heterogeneity setting ($h = 0.9$), which is expected from the analysis which predicted that mixture models wouldn't be well-calibrated in this setting.

4.3.2 Regression Results

For regression, β-predBayes was evaluated, along with the product and mixture models as well as the baselines from chapter 3 which use an ensemble for predictive inference, since these output a distribution over predictions (rather than just the mean).

A note on implementation for regression: the distilled student model is the same architecture as the client models, except in the last layer where it is made to output both a mean and an input-dependent variance. This network is trained to minimize the KL divergence between its output Gaussian distribution, and that of the teacher network.

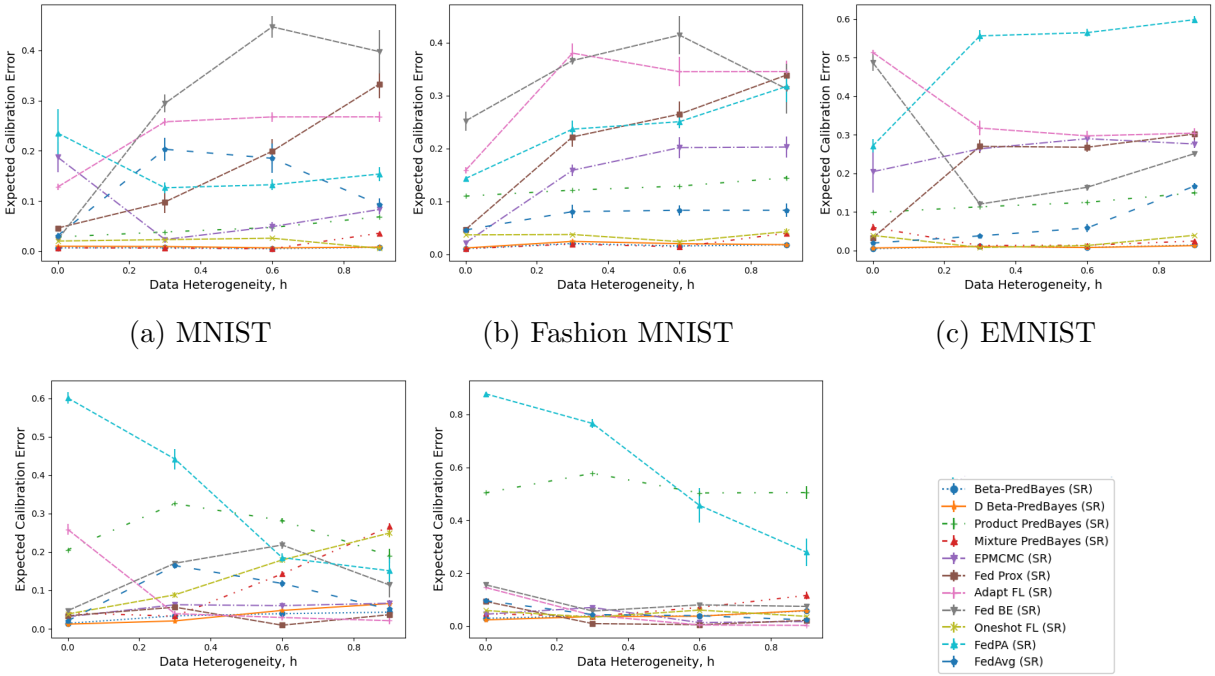
The resulting (Gaussian) negative log-likelihoods for a heterogeneity parameter of $h = 1.0$ are shown in table 4.2. We can see from this table that for all datasets except "Forest Fire", the distilled form of β-PredBayes performs best. It is possible that the distilled version outperforms the non-distilled one due to some regularization effect of having a smaller model.



(a) MNIST (b) Fashion MNIST (c) EMNIST

(d) CIFAR10 (e) CIFAR100 (f) Legend

Figure 4.1: Negative log likelihoods on the classification datasets with increasing data heterogeneity (tested with parameter settings of h={0.0, 0.3, 0.6, 0.9}). Averages and standard error over 5 seeds are reported. The omitted values (eg. for FedPA on EMNIST) denote results where the negative log likelihood diverged.



(a) MNIST (b) Fashion MNIST (c) EMNIST

(d) CIFAR10 (e) CIFAR100 (f) Legend

Figure 4.2: Expected Calibration Errors on the classification datasets with increasing data heterogeneity (tested with parameter settings of h={0.0, 0.3, 0.6, 0.9}). Averages and standard error over 5 seeds are reported.

Table 4.2: Average negative log likelihood (± standard error) on regression datasets, based on 5 seeds. **Lower is better**.

Method	Air Quality	Bike	Wine Quality	Real Estate	Forest Fire
EP MCMC	11.114 ± 1.241	1.710 ± 0.121	2.977 ± 0.185	1.883 ± 0.303	1.664 ± 0.073
FedBE	9.311 ± 0.156	0.826 ± 0.027	2.126 ± 0.013$^{\Uparrow}$	0.507 ± 0.012	1.397 ± 0.025
OneshotFL	11.315 ± 0.410	0.843 ± 0.026	2.164 ± 0.026$^{\Uparrow}$	0.562 ± 0.019	1.397 ± 0.025
Mixture	9.028 ± 0.049	1.176 ± 0.010	2.555 ± 0.051	0.634 ± 0.006	**1.390** ± 0.005
Product	8.981 ± 0.134	1.433 ± 0.038	2.894 ± 0.109	0.708 ± 0.057	2.564 ± 0.023
β-PredBayes (**ours**)	4.739 ± 0.130	0.943 ± 0.019	1.998 ± 0.059	0.473 ± 0.029	1.551 ± 0.011
D β-PredBayes (**ours**)	**4.550** ± 0.167	**0.159** ± 0.036	**1.262** ± 0.031	**0.210** ± 0.041	1.525 ± 0.041

Chapter 5

Conclusion

This work presented β-Predictive Bayes, an algorithm which aggregates local Bayesian posteriors in predictive space using a tunable parameter β, and then distills the resulting model. Owing to its Bayesian nature, the method is well suited for heterogeneous FL settings, and only requires a single round of communication. The tunable parameter allows the method to obtain accurate calibration performance. We performed experiments on various classification and regression datasets to show that the method performs competitively with other FL techniques, and that it outperforms them in more heterogeneous settings on both accuracy and calibration. The work reinforces the idea that Bayesian learning is an effective tool for obtaining well calibrated models in heterogeneous settings without incurring heavy communication costs.

5.1 Limitations and Future Work

Some directions for improving the proposed method include:

- **Personalization** The proposed approach aims to construct a global model for all clients (in this case, based on the global Bayesian posterior). As such, in cases where the clients have different class-conditional distributions $p(y|x)$, a personalized model would be desirable for each client. This is an important direction for future work.
- **Privacy** The proposed method communicates weight samples to the server, which are obtained via MCMC. Although no data is shared, information about the data could be leaked via the model parameters unless a differentially private mechanism

is used or encryption is applied. This trait is shared with other techniques (such as FedAvg and FedPA). It would be interesting to explore the use of differentially private sampling mechanisms [16].

- **Server dataset** β-Predictive Bayes uses a public dataset stored at the server for distillation and tuning β. In cases where such a public dataset isn't available, it may need to be synthetically generated. It would be better to avoid this generation process, and develop a data-free technique for compressing the ensemble and learning β in the future. One possibility for this is to have clients communicate the gradients of their dataset's negative log likelihood to the server, and then use these to approximate the gradient for learning β.

References

[1] 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*, 2021.

[2] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On markov chain monte carlo methods for tall data. *Journal of Machine Learning Research*, 18(47):1–43, 2017.

[3] Raef Bassily, Albert Cheu, Shay Moran, Aleksandar Nikolov, Jonathan Ullman, and Steven Wu. Private query release assisted by public data. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 695–703. PMLR, 13–18 Jul 2020.

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

[5] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.

[6] Yanshuai Cao and David J. Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *ArXiv*, abs/1410.7827, 2014.

[7] Hong-You Chen and Wei-Lun Chao. Fed{be}: Making bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, 2021.

[8] Taeryon Choi and Mark J. Schervish. On posterior consistency in nonparametric regression problems. *Journal of Multivariate Analysis*, 98(10):1969–1987, 2007.

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[10] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009.

[11] Paulo Cortez and Aníbal de Jesus Raimundo Morais. A data mining approach to predict forest fires using meteorological data. 2007.

[12] Ronald Cramer, Ivan Bjerre Damgård, et al. *Secure multiparty computation*. Cambridge University Press, 2015.

[13] A. P. Dawid. The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379):605–610, 1982.

[14] Saverio De Vito, Ettore Massera, Marco Piga, Luca Martinotto, and Girolamo Di Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129(2):750–757, 2008.

[15] Marc P. Deisenroth and Jun W. Ng. Robust bayesian committee machine for large-scale gaussian processes. In *Large-Scale Kernel Machines Workshop at ICML 2015*, 2015.

[16] Christos Dimitrakakis, Blaine Nelson, Zuhe Zhang, Aikateirni Mitrokotsa, and Benjamin IP Rubinstein. Differential privacy for bayesian inference through posterior sampling. *Journal of machine learning research*, 18(11):1–39, 2017.

[17] Joseph L Doob. Application of the theory of martingales. *Le calcul des probabilites et ses applications*, pages 23–27, 1949.

[18] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[19] Alain Durmus and Eric Moulines. High-dimensional bayesian inference via the unadjusted langevin algorithm, 2016.

[20] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15, 2013.

[21] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15, 2013.

[22] Arun Ganesh and Kunal Talwar. Faster differentially private samplers via rényi divergence analysis of discretized langevin mcmc. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7222–7233. Curran Associates, Inc., 2020.

[23] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning, 2019.

[24] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. 2017.

[25] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[26] Mikko Heikkilä, Joonas Jälkö, Onur Dikmen, and Antti Honkela. Differentially private markov chain monte carlo. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[27] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[28] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[29] IJCAI Proceedings. IJCAI camera ready submission. https://proceedings.ijcai.org/info.

[30] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In Ricardo Silva, Amir Globerson, and Amir Globerson, editors, *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[31] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020.

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[33] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[35] Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1484–1490. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[36] 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.

[37] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[38] 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.

[39] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, Peyman Adibi, Payam Barnaghi, and Amit P Sheth. Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, 4(3):161–175, 2018.

[40] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[41] Diego Mesquita, Paul Blomstedt, and Samuel Kaski. Embarrassingly parallel mcmc using deep invertible transformations. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 1244–1252. PMLR, 22–25 Jul 2020.

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

[43] Jacob M. Montgomery, Florian M. Hollenbach, and Michael D. Ward. Improving predictions using ensemble bayesian model averaging. *Political Analysis*, 20(3):271–291, 2012.

[44] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021.

[45] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[46] Willie Neiswanger, Chong Wang, and Eric P. Xing. Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 623–632, Arlington, Virginia, USA, 2014. AUAI Press.

[47] Agustín G Nogales. On consistency of the Bayes estimator of the density. *Mathematics*, 10(4):636, 2022.

[48] Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.

[49] Arya A Pourzanjani, Richard M Jiang, and Linda R Petzold. Improving the identifiability of neural networks for Bayesian inference. In *NeurIPS Workshop on Bayesian Deep Learning*, 2017.

[50] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[51] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[52] Sam Roweis. Gaussian identities. *Unpublished Notes*, 1999.

[53] Ossi Räisä, Antti Koskela, and Antti Honkela. Differentially private hamiltonian monte carlo, 2021.

[54] J. Sanz-Serna. Markov chain monte carlo and numerical differential equations. *Lecture Notes in Mathematics*, 2082, 01 2014.

[55] Minh-Ngoc Tran, Trong-Nghia Nguyen, and Viet-Hung Dao. A practical tutorial on variational bayes, 2021.

[56] Volker Tresp. A bayesian committee machine. *Neural computation*, 12(11):2719–2741, 2000.

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

[58] 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.

[59] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[60] Blake E. Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local SGD for heterogeneous distributed learning. *CoRR*, abs/2006.04735, 2020.

[61] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

[62] I-Cheng Yeh and Tzu-Kuang Hsu. Building real estate valuation models with comparative approach through case-based reasoning. *Applied Soft Computing*, 65:260–271, 2018.

[63] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 09–15 Jun 2019.

[64] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning. *International Conference on Learning Representations*, 2020.

[65] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[66] Zhaohua Zheng, Yize Zhou, Yilong Sun, Zhang Wang, Boyi Liu, and Keqiu Li. Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges. *Connection Science*, 34(1):1–28, 2022.

[67] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning, 2020.

APPENDICES

Appendix A

Gaussian Regression Formula

We derive the aggregation formulas for regression described in Equations (5) and (6) in the main paper. These are similar to aggregation formulas for products of Gaussian densities. The only difference is that we are also dividing by Gaussian densities in this case. We therefore confirm the formulas with a derivation.

We would like to obtain an expression for the global predictive posterior $p(y|x, \mathcal{D})$ in (A.1).

$$p(y|x, \mathcal{D}) = \frac{1}{p(y|x)^{n-1}} \prod_i p(y|x, \mathcal{D}_i) \tag{A.1}$$

We assume:

- $p(y|x, \mathcal{D}_i) = \mathcal{N}(\mu_i, \Sigma_i)$ (the local posterior for client i is a Gaussian distribution)
- $p(y|x) = \mathcal{N}(\mu_p, \Sigma_p)$ (the predictive prior is also a Gaussian)

For convenience, we work with the precision matrix (rather than the covariance matrix) $S = \Sigma^{-1}$.

First note that for the generic Gaussian (μ, S^{-1}), its log-density is expressed as:

$$\log p(x) = -\frac{1}{2}(x - \mu)^\top S(x - \mu) + C \tag{A.2}$$

$$= -\frac{1}{2}[x^\top Sx - \mu^\top Sx - x^\top S\mu + \mu^\top S\mu] + C \tag{A.3}$$

where C is a normalization constant. Taking the logarithm of (A.1) and expanding the right hand side we obtain:

$$\begin{aligned}
\log p(y|x,\mathcal{D}) &= \sum_i \log p(y|x,\mathcal{D}_i) - (n-1)\log p(y|x) \\
&= \sum_i -\frac{1}{2}(x-\mu_i)^\top S_i(x-\mu_i) + C_i \\
&\quad - (n-1)[-\frac{1}{2}(x-\mu_p)^\top S_p(x-\mu_p) + C_p] \\
&= -\frac{1}{2}[x^\top(\sum_i S_i)x - (\sum_i \mu_i^\top S_i)x \\
&\quad - x^\top(\sum_i S_i\mu_i) + \sum_i \mu_i^\top S_i\mu_i \\
&\quad - (n-1)x^\top S_p x \\
&\quad + (n-1)\mu_p^\top S_p x + (n-1)x^\top S_p\mu_p \\
&\quad - (n-1)\mu_p^\top S_p\mu_p] + \sum_i C_i - (n-1)C_p
\end{aligned}$$

Writing the overall normalization constant as $\bar{C}$, and collecting like terms gives:

$$
\begin{aligned}
\log p(y|x,\mathcal{D}) = -\frac{1}{2}[&x^\top(\sum_i S_i - (n-1)S_p)x \\
&- (\sum_i \mu_i^\top S_i - (n-1)\mu_p^\top S_p)x \\
&- x^\top(\sum_i S_i\mu_i - (n-1)S_p\mu_p) \\
&+ \sum_i \mu_i^\top S_i\mu_i - (n-1)\mu_p^\top S_p\mu_p] + \bar{C}
\end{aligned}
\tag{A.4}
$$

Since (A.4) is a quadratic form in x, we can deduce that $p(y|x,\mathcal{D})$ is Gaussian with some mean μ_g, and precision S_g. Matching the coefficients of (A.4) with (A.3), we can identify the precision from the first term as:

$$
S_g = \sum_i S_i - (n-1)S_p \tag{A.5}
$$

We can compare the third term (the linear term with $x^\top$) in both equations to see the mean:

$$
\begin{aligned}
S_g\mu_g &= \sum_i S_i\mu_i - (n-1)S_p\mu_p \\
\implies \mu_g &= S_g^{-1}[\sum_i S_i\mu_i - (n-1)S_p\mu_p]
\end{aligned}
\tag{A.6}
$$

Recasting Equations (A.5) and (A.6) in terms of the covariance we obtain the desired result:

$$
\Sigma_g = \Big(\sum_i \Sigma_i^{-1} - (n-1)\Sigma_p^{-1}\Big)^{-1} \tag{A.7}
$$

$$
\mu_g = \Sigma_g\Big(\sum_i \Sigma_i^{-1}\mu_i - (n-1)\Sigma_p^{-1}\mu_p\Big) \tag{A.8}
$$

Appendix B

Additional Experiments and Hyperparameters for D BCM

B.1 Additional Experimental Details

Additional details for the experiments are provided below.

B.1.1 Hardware, Software, and Randomization Details

The code for experiments was written in the Python language (version 3.8.10), primarily using the Pytorch (version 1.9.0), Numpy (version 1.19.5) and Scipy (version 1.6.2) libraries. Randomization was done by setting seeds for Pytorch and Numpy.

Experiments were carried out on a compute cluster using a single Nvidia GPU (either the T4, or P100).

B.2 Hyperparameter Tuning

Hyperparameters were selected based on searching a grid for the best performing settings according to the validation set performance (accuracy for classification, and mean squared error for regression).

The hyperparameters tuned, and their corresponding grids are outlined in Table B.1 for both classification and regression. Note that FedPA requires a sampler at each client,

and an optimizer at the server. To distinguish where each hyperparameter is used for this algorithm, we therefore label these cases FedPA(C) and FedPA(S) respectively. We further distinguish between the single round (SR), and multi-round (MR) versions of this algorithm.

The optimizers used for client training include SGD, SGD with momentum (SGDM), and Adam, while for distillation, we also used Stochastic Weight Averaging (SWA) as a possible optimizer.

The tuned hyperparameter settings for the homogeneous ($h = 0$) classification datasets are in Table B.2, while for the heterogeneous setting $h > 0$ they are in Table B.3. For the regression datasets, the tuned values are in Table B.4.

Note about reading tables: for these tables, if a hyperparameter is repeated more than once, with an algorithm named beside it in brackets, it means the hyperparameter for that algorithm is different. The rest of the algorithms associated with that hyperparameter use the value listed without brackets. For instance, in Table B.4, for the "Bike" dataset, the sampler learning rate is listed in the row "Sampler Learning Rate" as 2e-1, while it is listed separately with the additional specification "FedPA (C) MR" (multi-round), as 5e-1. This means that FedPA, in the multi-round case, uses a sampler learning rate of 5e-1, while the other sampling algorithms (including FedPA SR - in the single round case) use 2e-1.

Other hyperparameters not part of the grid search include:

- Batch size: fixed to 100 for all experiments
- Momentum in SGDM: fixed to 0.9 for all experiments
- Model architecture (as described in the main paper)

More algorithm-specific decisions/hyperparameters include:

- FedBE: 10 model samples were drawn from the approximate posterior to use in the ensemble for all experiments (following the experiments in the original paper. This gave a total ensemble size of 16 models = 10 (sampled) + 5 (client models) + 1 (averaged model). By contrast BCM contained an ensemble with 5 models.
- Adaptive FL: The FedYogi server update was used, based on the results from, which suggested that it performed best among their proposed variants. $\beta_1 = 0.9$ and $\beta_2 = 0.99$ were fixed, again, based on the paper.

- One-Shot FL: For the classification case, aggregation is done by averaging the logits of the client models. (This is opposed to averaging and normalizing the probabilities, after the softmax layer).

In tables B.1, B.2, B.3 and B.4, LR is used to denote learning rate.

B.2.1 Heterogeneous Classification Dataset Construction

The process for constructing a heterogeneous classification dataset is as follows:

- A parameter $h \in [0, 1]$ is fixed
- The dataset is sorted by class labels, and split evenly into shards for each client (the "fully heterogeneous shards")
- A copy of the dataset is made and split such that each shard contains a roughly uniform class distribution for each client (the "homogeneous shards")
- To form the final shard for a client, a fraction h of each homogeneous shard is replaced with the data from the corresponding fully-heterogeneous shard.

In this way, $h = 0$ corresponds to the homogeneous data setting, while $h = 1.0$ corresponds to a degenerate heterogeneous case (for class distributions in each client).

B.3 Additional Experiments

B.3.1 Classification Experiments

The results for the classification datasets, on $h = \{0.0, 0.6, 0.9\}$ (over 10 random seeds, and with indicators for statistical significance) are written in Tables B.5, B.6, and B.7 respectively. These are provided to give additional detail to the results provided in Figure 1 of the main paper. More precisely, the results in Figure 1 (of the main paper) show the qualitative trends of each algorithm as heterogeneity increases, while the results in Tables B.5, B.6, and B.7 show corresponding precise numerical values with estimates of statistical significance. The case for $h = 0.3$ is included as Table 1 in the main paper. In the case of $h = \{0.6, 0.9\}$, the best performing one round algorithm (in terms of average performance)

is BCM, followed by the distilled variant D BCM. This is also the case for $h = 0.0$ with the exception of the CIFAR datasets. For these, other algorithms perform slightly better than BCM (or D BCM). A possible reason for this is that the conditional independence assumption underlying the aggregation formula (A.1) fails to hold in the homogeneous data setting. Nevertheless, the results for $h > 0$ reinforce the idea that the aggregation formula is useful in heterogeneous settings.

Table B.1: The hyperparameters tuned, their possible values in the grid search, and the algorithms each hyperparameter applies to.

Hyperparameter	Grid Settings		Algorithms Used In
	Classification	Regression	
Optimizer FedPA(S), FedProx, AdaptFL, FedBE	{SGD, SGDM, Adam}		FedAvg, OneshotFL, FedKT,
Local LR FedProx, AdaptFL, FedBE	{1e-1, 1e-2, 1e-3}	{1e-1, 1e-2, 1e-3, 1e-4}	FedAvg, OneshotFL, FedKT,
Server LR	{1, 5e-1, 1e-1, 1e-2}		FedPA(S), AdaptFL
Cov. Param (ρ)	{0.4, 0.9, 1.0}		FedPA(C)
Proximal Param. (λ)	{1, 1e-1, 1e-2, 1e-3}		FedProx
Adaptivity (τ)	{1, 1e-1, 1e-2, 1e-3}		AdaptFL
Sampler LR FedPA(C)	{5e-1, 1e-1, 1e-2,1e-3}	{5e-1, 2e-1, 1e-1, 1e-2, 1e-3}	(D)BCM, EPMCMC,
Maximum Samples FedPA(C)	{4,6,12}		(D)BCM, EPMCMC,
Temperature FedPA(C)	$\{\frac{1}{\lvert\mathcal{D}\rvert}\}$	{1, 5e-1, 5e-2, $\frac{1}{\lvert\mathcal{D}\rvert}$ }	(D)BCM, EPMCMC,
Sampler Cycles FedPA(C)	{5}	{2, 4,5}	(D)BCM, EPMCMC,
Samples per cycle FedPA(C)	{2}	{1,2,3}	(D)BCM, EPMCMC,
Distill Optimizer FedKT, FedBE	{SGDM, Adam, SWA}		D BCM, OneshotFL,
Distill LR FedKT, FedBE	{1e-2, 5e-3, 1e-4}	{1e-2, 5e-3, 1e-3, 1e-4}	D BCM, OneshotFL,
Distill Epochs FedKT, FedBE	{100,50, 20}		D BCM, OneshotFL,

Table B.2: The tuned values of hyperparameters for the classification datasets in the homogeneous case $h = 0$

Hyperparameter	Tuned Value				
	MNIST	Fashion MNIST	EMNIST	CIFAR10	CIFAR100
Optimizer	SGDM				
Optimizer (FedProx)	SGDM	Adam	Adam	SGDM	SGDM
Optimizer (AdaptFL)	SGDM	SGDM	SGDM	SGD	SGD
Local LR	1e-1	1e-1	1e-1	1e-2	1e-2
Local LR (FedProx)	1e-1	1e-3	1e-3	1e-2	1e-2
Local LR (AdaptFL)	1e-2	1e-2	1e-2	1e-1	1e-1
Server LR	1	5e-1	1e-1	5e-1	5e-1
Server LR (AdaptFL)	1e-1				
Cov. Param (ρ)	0.4				
Proximal Param (λ)	1e-2	1e-3	1e-3	1e-3	1e-3
Adaptivity (τ)	1e-2				
Sampler LR	5e-1	1e-1	1e-1	1e-1	1e-1
Sampler LR (FedPA(C) SR, EP MCMC)	1e-1				
Sampler LR (FedPA(C) MR)	1e-2				
Maximum Samples	6				
Temperature	$\frac{1}{\lvert\mathcal{D}\rvert}$				
Sampler Cycles	5				
Samples per cycle	2				
Distill Optimizer	Adam				
Distill LR	1e-4				
Distill Epochs	100				

Table B.3: The tuned values of hyperparameters for the classification datasets, in the heterogeneous case $h > 0$

Hyperparameter	Tuned Value				
	MNIST	Fashion MNIST	EMNIST	CIFAR10	CIFAR100
Optimizer	SGDM				
Optimizer (FedProx)	Adam	Adam	Adam	SGDM	SGDM
Optimizer (AdaptFL)	SGD				
Local LR	1e-2	1e-2	1e-3	1e-3	1e-3
Local LR (FedProx)	1e-3	1e-3	1e-3	1e-2	1e-2
Local LR (AdaptFL)	1e-1				
Server LR	1	5e-1	1e-1	5e-1	5e-1
Server LR (AdaptFL)	1e-1				
Cov. Param (ρ)	0.4				
Proximal Param (λ)	1e-2	1e-2	1e-2	1e-3	1e-3
Adaptivity (τ)	1e-2				
Sampler LR	1e-1				
Sampler LR (FedPA(C) MR)	1e-2				
Maximum Samples	6				
Temperature	$\frac{1}{\lvert\mathcal{D}\rvert}$				
Sampler Cycles	5				
Samples per cycle	2				
Distill Optimizer	Adam				
Distill LR	1e-4				
Distill Epochs	100				

Table B.4: The tuned values of hyperparameters for the regression datasets

Hyperparameter	Tuned Value				
	Air Quality	Bike	Wine Quality	Real Estate	Forest Fire
Optimizer	Adam				
Optimizer (FedPA(S))	SGDM	Adam	SGDM	SGDM	SGDM
Optimizer (FedProx)	SGDM	Adam	Adam	Adam	SGDM
Optimizer (AdaptFL)	SGDM	SGDM	SGD	SGD	SGD
Local LR	1e-2	1e-2	1e-3	1e-2	1e-4
Local LR (FedProx)	1e-2	1e-2	1e-2	1e-1	1e-1
Local LR (AdaptFL)	1e-2	1e-1	1e-1	1e-1	1e-2
Server LR	1e-1	1e-2	5e-1	1	1
Server LR (AdaptFL)	1e-1	1e-1	1	1e-1	1
Cov. Param (ρ)	0.4	1.0	0.9	0.4	0.4
Proximal Param (λ)	1e-2	1e-3	1e-3	1e-2	1e-1
Adaptivity (τ)	1e-2	1e-1	1	1e-2	1e-2
Sampler LR	1e-1	2e-1	2e-1	2e-1	1e-2
Sampler LR (FedPA(C) MR)	1e-1	5e-1	5e-1	1e-2	1e-2
Maximum Samples	6	4	4	6	4
Temperature	1	$\frac{1}{\lvert\mathcal{D}\rvert}$	5e-2	5e-1	5e-1
Sampler Cycles	5	5	4	5	2
Samples per cycle	1	2	2	2	2
Distill Optimizer	Adam				
Distill LR	1e-3	1e-3	5e-3	5e-3	5e-3
Distill Epochs	100				

Table B.5: Average test accuracies (± standard error) on classification datasets for $h = 0.0$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test).

$h = 0.0$ heterogeneity						
Method	Rounds	MNIST	Fashion MNIST	EMNIST	CIFAR10	CIFAR100
FedAvg	≥ 5	97.61 ± 0.04$^{\Uparrow}$	83.24 ± 0.36$^{\Downarrow}$	80.46 ± 0.19$^{\Downarrow}$	79.46 ± 0.07$^{\Downarrow}$	38.86 ± 0.16$^{\Uparrow}$
FedPA	≥ 5	95.48 ± 0.28$^{\Downarrow}$	76.59 ± 1.09$^{\Downarrow}$	75.52 ± 0.45$^{\Downarrow}$	41.83 ± 0.47$^{\Downarrow}$	5.03 ± 0.31$^{\Downarrow}$
FedAvg	1	93.63 ± 0.32$^{\Downarrow}$	86.61 ± 0.09	84.23 ± 0.11$^{\Downarrow}$	69.89 ± 0.33$^{\Downarrow}$	24.36 ± 0.31$^{\Downarrow}$
FedPA	1	95.30 ± 0.07$^{\Downarrow}$	84.29 ± 0.26$^{\Downarrow}$	62.62 ± 6.37$^{\Downarrow}$	40.71 ± 1.19$^{\Downarrow}$	8.10 ± 0.46$^{\Downarrow}$
EP MCMC	1	96.09 ± 0.05$^{\Downarrow}$	86.22 ± 0.09$^{\Downarrow}$	69.86 ± 6.89$^{\Downarrow}$	67.94 ± 0.17$^{\Downarrow}$	21.36 ± 0.16$^{\Downarrow}$
FedProx	1	94.34 ± 0.21$^{\Downarrow}$	85.30 ± 0.17$^{\Downarrow}$	74.61 ± 0.85$^{\Downarrow}$	71.40 ± 0.35$^{\Downarrow}$	25.16 ± 0.26$^{\Downarrow}$
AdaptFL	1	94.27 ± 0.28$^{\Downarrow}$	83.61 ± 0.20$^{\Downarrow}$	68.62 ± 0.55$^{\Downarrow}$	46.27 ± 0.89$^{\Downarrow}$	16.52 ± 0.32$^{\Downarrow}$
FedBE	1	94.70 ± 0.16$^{\Downarrow}$	79.85± 0.72$^{\Downarrow}$	72.18 ± 1.24$^{\Downarrow}$	73.65 ± 0.32$^{\downarrow}$	**32.35** ± 0.09$^{\Uparrow}$
Oneshot FL	1	95.39 ± 0.07$^{\Downarrow}$	83.91 ± 0.06$^{\Downarrow}$	78.64 ± 0.15$^{\Downarrow}$	**76.45** ± 0.14 $^{\Uparrow}$	30.66 ± 0.09$^{\Uparrow}$
FedKT	1	94.81 ± 0.05$^{\Downarrow}$	84.38 ± 0.08	81.01 ± 0.15$^{\Downarrow}$	69.35 ± 0.25$^{\Downarrow}$	28.88 ± 0.18
BCM (**ours**)	1	**97.01** ± 0.05$^{\Uparrow}$	**86.96** ± 0.04$^{\Uparrow}$	**87.68** ± 0.10$^{\Uparrow}$	73.85 ± 0.11$^{\Uparrow}$	29.52 ± 0.11$^{\Uparrow}$
D BCM (**ours**)	1	96.39 ± 0.08	86.57 ± 0.03	86.96 ± 0.08	72.74 ± 0.10	28.63 ± 0.11

Table B.6: Average test accuracies (± standard error) on classification datasets for $h = 0.6$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test).

$h = 0.6$ heterogeneity						
Method	Rounds	MNIST	Fashion MNIST	EMNIST	CIFAR10	CIFAR100
FedAvg	≥ 5	93.59 ± 0.11$^{\Downarrow}$	76.39 ± 0.72$^{\Downarrow}$	83.18 ± 0.04$^{\Downarrow}$	33.94 ± 0.25$^{\Downarrow}$	7.79 ± 0.16$^{\Downarrow}$
FedPA	≥ 5	89.41 ± 0.08$^{\Downarrow}$	79.59 ± 0.38$^{\Downarrow}$	67.54 ± 0.59$^{\Downarrow}$	39.47 ± 0.31$^{\Downarrow}$	7.96 ± 0.09$^{\Downarrow}$
FedAvg	1	50.28 ± 4.51$^{\Downarrow}$	33.46 ± 1.54$^{\Downarrow}$	63.10 ± 0.95$^{\Downarrow}$	25.06 ± 0.66$^{\Downarrow}$	5.85 ± 0.18$^{\Downarrow}$
FedPA	1	86.81 ± 0.85$^{\Downarrow}$	63.72 ± 1.60$^{\Downarrow}$	33.51 ± 1.02$^{\Downarrow}$	14.52 ± 1.24$^{\Downarrow}$	2.96 ± 0.29$^{\Downarrow}$
EP MCMC	1	90.07 ± 0.05$^{\Downarrow}$	67.17 ± 1.69$^{\Downarrow}$	48.39 ± 1.26$^{\Downarrow}$	11.35± 0.53$^{\Downarrow}$	3.57 ± 0.39$^{\Downarrow}$
FedProx	1	83.95 ± 0.87$^{\Downarrow}$	68.78 ± 1.57$^{\Downarrow}$	39.38 ± 0.77$^{\Downarrow}$	10.03 ± 0.03$^{\Downarrow}$	1.04 ± 0.03$^{\Downarrow}$
AdaptFL	1	90.81 ± 0.39$^{\Downarrow}$	71.60 ± 1.20$^{\Downarrow}$	38.72 ± 0.93$^{\Downarrow}$	11.49 ± 0.58$^{\Downarrow}$	1.76 ± 0.13$^{\Downarrow}$
FedBE	1	86.57 ± 1.74$^{\Downarrow}$	77.43 ± 3.48$^{\Downarrow}$	75.11 ± 0.28$^{\Downarrow}$	35.71 ± 0.64$^{\Downarrow}$	10.20 ± 0.18$^{\Downarrow}$
Oneshot FL	1	92.92 ± 0.08$^{\Downarrow}$	82.37 ± 0.14$^{\Downarrow}$	81.33 ± 0.05$^{\Downarrow}$	39.00 ± 0.56$^{\Downarrow}$	10.14 ± 0.12$^{\Downarrow}$
FedKT	1	92.93 ± 0.05$^{\Downarrow}$	82.48 ± 0.17$^{\Downarrow}$	82.47 ± 0.05$^{\Downarrow}$	28.20 ± 0.57$^{\Downarrow}$	6.54 ± 0.22$^{\Downarrow}$
BCM (**ours**)	1	**94.54** ± 0.08	**84.76** ± 0.10$^{\Uparrow}$	**85.04** ± 0.23$^{\Uparrow}$	**59.26** ± 0.21$^{\Uparrow}$	**23.95** ± 0.31$^{\Uparrow}$
D BCM (**ours**)	1	94.47 ± 0.05	84.47 ± 0.11	84.36 ± 0.23	58.48 ± 0.20	22.88 ± 0.25

Table B.7: Average test accuracies (± standard error) on classification datasets for $h = 0.9$, based on 10 seeds. **Higher is better**. The best technique among single round methods is bolded. $\Uparrow$ / $\Downarrow$: higher/lower accuracy with $p < 1\%$, $\uparrow$ / $\downarrow$: higher/lower accuracy with $p < 5\%$ (relative to D BCM) (according to the Wilcoxon signed-rank test).

$h = 0.9$ heterogeneity						
Method	Rounds	MNIST	Fashion MNIST	EMNIST	CIFAR10	CIFAR100
FedAvg	≥ 5	86.64 ± 1.04$^{\Downarrow}$	62.75 ± 1.80$^{\Downarrow}$	79.81 ± 0.04$^{\Downarrow}$	34.62 ± 0.22$^{\Downarrow}$	7.28 ± 0.19$^{\Downarrow}$
FedPA	≥ 5	86.39 ± 0.17$^{\Downarrow}$	78.33 ± 0.17$^{\Downarrow}$	59.70 ± 0.58$^{\Downarrow}$	34.86 ± 0.40$^{\Downarrow}$	6.88 ± 0.20$^{\Downarrow}$
FedAvg	1	28.23 ± 2.26$^{\Downarrow}$	24.01 ± 1.48$^{\Downarrow}$	59.03, ± 0.56$^{\Downarrow}$	16.16 ± 1.05$^{\Downarrow}$	3.97 ± 0.15$^{\Downarrow}$
FedPA	1	83.69 ± 0.94$^{\Downarrow}$	57.24 ± 1.80$^{\Downarrow}$	29.77 ± 0.81$^{\Downarrow}$	13.52 ± 0.70$^{\Downarrow}$	2.76 ± 0.24$^{\Downarrow}$
EP MCMC	1	87.63, ± 0.39$^{\Downarrow}$	63.73 ± 1.50$^{\Downarrow}$	45.59 ± 0.83$^{\Downarrow}$	10.21 ± 0.16$^{\Downarrow}$	4.33 ± 0.54$^{\Downarrow}$
FedProx	1	81.99 ± 0.83$^{\Downarrow}$	66.05 ± 1.13$^{\Downarrow}$	40.00 ± 1.23$^{\Downarrow}$	10.17 ± 0.16$^{\Downarrow}$	1.03 ± 0.03$^{\Downarrow}$
AdaptFL	1	88.41 ± 0.26$^{\Downarrow}$	70.03 ± 1.35$^{\Downarrow}$	38.71 ± 0.93$^{\Downarrow}$	12.18 ± 0.70$^{\Downarrow}$	1.63 ± 0.15$^{\Downarrow}$
FedBE	1	71.95 ± 4.43$^{\Downarrow}$	59.31 ± 4.38$^{\Downarrow}$	72.47 ± 0.18$^{\Downarrow}$	23.13 ± 2.04$^{\Downarrow}$	10.77 ± 0.25$^{\Downarrow}$
Oneshot FL	1	91.40 ± 0.09$^{\Downarrow}$	77.77 ± 0.42$^{\Downarrow}$	77.98 ± 0.07$^{\Downarrow}$	43.18 ± 0.81$^{\Downarrow}$	9.36 ± 0.19$^{\Downarrow}$
FedKT	1	89.81 ± 0.14$^{\Downarrow}$	75.26 ± 0.56$^{\Downarrow}$	77.55 ± 0.07$^{\Downarrow}$	10.21 ± 0.28$^{\Downarrow}$	5.10 ± 0.10$^{\Downarrow}$
BCM (**ours**)	1	**91.75** ± 0.06	**81.52** ± 0.18$^{\Uparrow}$	**82.35** ± 0.24$^{\Uparrow}$	52.15 ± 0.44$^{\downarrow}$	**20.24** ± 0.40$^{\Uparrow}$
D BCM (**ours**)	1	91.68 ± 0.07	81.17 ± 0.16	81.82 ± 0.26	**52.62** ± 0.48	19.05 ± 0.39