Distributed Variational Inference and Privacy

Xiping Liu

Supervisor: Dr Richard E. Turner

Department of Engineering
University of Cambridge

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

Hughes Hall August 2019

Declaration

I, Xiping Liu of Hughes Hall, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Second Declaration: I declare that the code I wrote for this project is built upon the existing code repository written by Mrinank Sharma and Michael Hutchinson. The existing code contains implementation of Partitioned Variational Inference (PVI), differentially private stochastic gradient descent (DP-SGD), and the moments accountant. I made a new branch in the existing code repository, and added my implementation of Bayesian multi-dimensional linear regression models and adaptive clipping bounds. This code repository may be found on GitHub at this address: https://github.com/MrinankSharma/DP-PVI/tree/multidim-linreg. Sections 4.2 and 4.3 of this report relied on this code repository.

Total word count: 10432

Xiping Liu
August 2019

Acknowledgements

I would like to acknowledge and thank my supervisor, Dr Rich Turner, for his patient guidance and constant support throughout the project. I would also like to thank Mrinank Sharma, Michael Hutchinson, and Siddharth Swaroop, who have helped me a lot with both theoretical concepts and programming. It has been a pleasure to work with all of them, and I have learned a lot from the experience.

Last but not least, I would like to thank my beloved parents and girlfriend Ke who have always been there for me.

Abstract

This dissertation extends Partitioned Variational Inference (PVI) to support private federated learning using the concept of differential privacy (DP). Specifically, we describe the construction of two types of Differentially Private Partitioned Variational Inference (DP-PVI) algorithms: data-point level DP-PVI and data-set level DP-PVI, each of which has its own advantages and disadvantages. We point out the technical problems in the data-set level DP-PVI algorithm and provide possible solutions. Next, we apply the data-point level DP-PVI algorithm to Bayesian multi-dimensional linear regression models. We mainly conduct experiments on mini-batching and adaptive clipping bounds, investigating how they affect the performance of data-point level DP-PVI.

Table of contents

Chapter 1

Introduction

Many critical future applications of machine learning will be in distributed environments. For example, a cancer research center is organizing several health care providers to jointly train a machine learning model that can be used to make clinically important predictions. Each health care provider has data from many patients, but privacy issues and network bandwidth constraints prevent the cancer research center from centralizing all these data. This is an example of *federated learning*, where a number of clients (in this example, health care providers) interact with a central server (in this example, the cancer research center) to collaboratively train a shared machine learning model without centralizing all data together (McMahan et al., 2016; Konečnỳ et al., 2016).

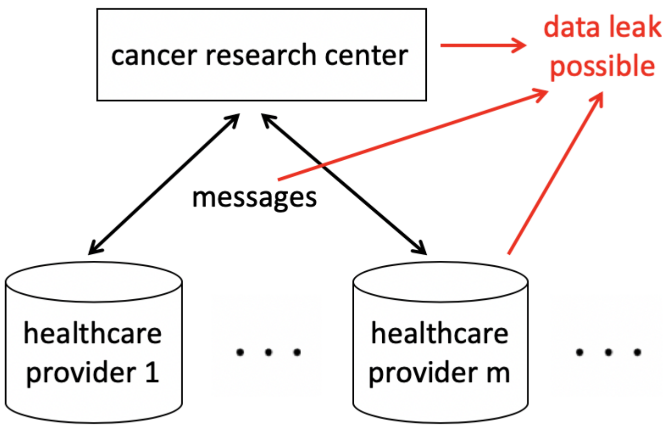


Fig. 1.1 Illustration of the cancer research center example and its potential privacy issues.

Moreover, in the above scenario, it is critical for the prediction system to know where it is uncertain, and thus when it might be returning inaccurate predictions. Therefore, *Bayesian inference* is needed, since it is able to quantify the uncertainty in parameter estimates and predictions of a probabilistic model. Dr Turner's group has developed an approach targeted to such applications called *Partitioned Variational Inference* (Bui et al., 2018).

Privacy is still an issue in the above example. Data of patients are highly private and sensitive, but machine learning models can leak information about patients' data used for training. Fredrikson et al. (2014) introduced *model inversion attacks*: an adversary uses predictions of a machine learning model to learn information about the training data. They also showed that when a machine learning model is used to guide medical treatments based on a patient's genotype and background, a model inversion attack is able to reveal patient's genetic markers. Fredrikson et al. (2015) further showed that by investigating confidence values associated with predictions of a neural network for face recognition, an adversary can recover images of people's faces used for training.

Moreover, even though data about individuals used in training are usually *anonymized* by removing identifiable information (e.g. names, ages, genders, etc.) and being labeled by an anonymous identifier, it is possible to reveal specific individuals using *auxiliary information*, i.e. additional, publicly available information (Narayanan and Shmatikov, 2008). They applied de-anonymization techniques to the Netflix Prize data set, which contains anonymous movie ratings of 500,000 subscribers of Netflix, and identified the Netflix records of two known users using the Internet Movie Database (IMDb) as auxiliary information. Gymrek et al. (2013) also showed that it is possible to recover particular surnames from anonymous DNA sequence data with the help of free, publicly accessible Internet resources.

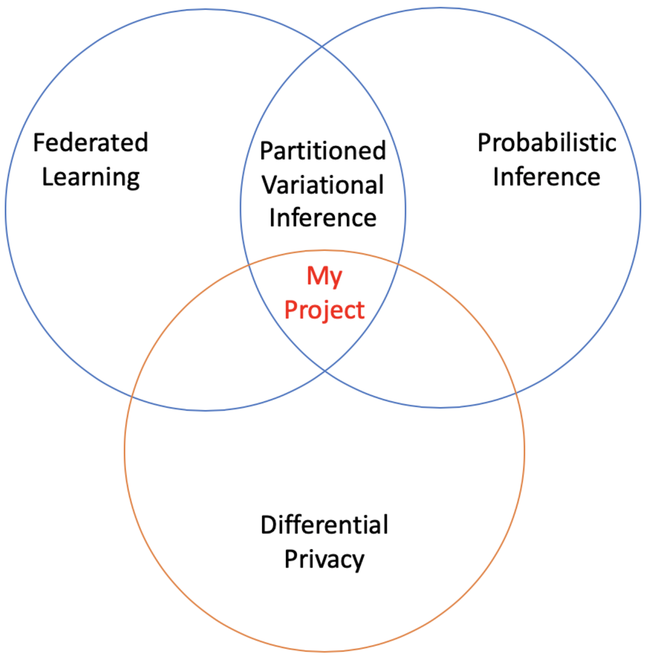


Fig. 1.2 Main concepts involved in this project.

In the light of the research findings above, machine learning models are able to leak information about individuals' data used for training under model inversion attacks, and those data are even under the risk of being de-anonymized, so improving the data privacy of machine learning models has become necessary and urgent. Therefore, the goal of my project is to extend Partitioned Variational Inference to support private federated learning using the concept of *differential privacy*. My project lies in the intersection of federated learning, probabilistic inference, and differential privacy, which is a rising field with limited previous work.

The rest of this dissertation is structured as follows. In chapter 2, we will cover necessary background on federated learning and differential privacy. In chapter 3, we will elaborate on our algorithms constructed by extending Partitioned Variational Inference with differential privacy techniques. In chapter 4, we will apply the algorithms which we construct in chapter 3 to Bayesian multi-dimensional linear regression models, and provide the results of our experiments. In chapter 5, we will summarize our conclusions and specify possible future work.

Chapter 2

Literature Review

In this chapter, we cover necessary background literature on federated learning and then on differential privacy.

2.1 Federated Learning

Traditional machine learning models require the whole training data set to be stored in a single machine or distributed among several machines in a homogeneous and balanced way. Compared to the traditional training pipelines, Bui et al. (2018) pointed out that federated learning is challenging because:

- data sets can be distributed inhomogeneously and unevenly across the clients (e.g. the data across client cannot be assumed to be i.i.d.);
- communication between the clients and the central server can be costly and unreliable;
- when making predictions, each client need to access the model instantly rather than sending data to the central server.

2.1.1 Partitioned Variational Inference

Partitioned Variational Inference (PVI) is an unifying framework which encompasses many approaches to *variational Bayesian methods*, and it tackles the aforementioned challenges in federated learning.

Suppose the training data set $\mathcal{D}$ is partitioned into M groups of data points $\{\mathcal{D}_1, ..., \mathcal{D}_M\}$. To model this data set, we consider a parametric probabilistic model defined by the prior $p(\theta)$ over parameters θ and the likelihood function $p(\mathcal{D}|\theta) = \prod_{m=1}^{M} p(\mathcal{D}_m|\theta)$.

For many models used in practice, exact Bayesian inference of the posterior distribution over the parameters $p(\theta|\mathcal{D})$ is in general intractable. Thus, we resort to *variational inference (VI)*, and use a variational distribution $q(\theta)$ to approximate the true posterior distribution. In detail, $q(\theta)$ can be expressed as follows:

$$q(\theta) = p(\theta)\prod_{m=1}^{M} t_m(\theta) \approx \frac{1}{\mathcal{Z}} p(\theta)\prod_{m=1}^{M} p(\mathcal{D}_m|\theta) = p(\theta|\mathcal{D}) \tag{2.1}$$

where $\mathcal{Z}$ is the normalizing constant of the true posterior. In this report, we refer to $q(\theta)$ as the approximate posterior, and $t_m(\theta)$ as the approximate likelihood, each of which will be updated by the PVI algorithm to approximate the effect of the likelihood term $p(\mathcal{D}_m|\theta)$ has on the posterior. Note that the approximate posterior $q(\theta)$ does not include a normalizing constant. Instead, the PVI algorithm will automatically ensure the approximate posterior in equation 2.1 is normalized (Bui et al., 2018).

```
Algorithm 1: Partitioned Variational Inference (PVI)
  Input: data partition $\{\mathcal{D}_1, ..., \mathcal{D}_M\}$, prior $p(\theta)$, a tractable family of distributions $\mathcal{Q}$
1 Initialize the approximate likelihoods:
      $t_m^{(0)}(\theta) = 1$ for all $m = 1, 2, ..., M$    (2.2)
2 Initialize the approximate posterior:
      $q^{(0)}(\theta) = p(\theta)$    (2.3)
3 for $i = 1, 2, ...$ until convergence do
4     $b_i$ = index of the next approximate likelihood to update
5     Compute the new approximate posterior:
          $q^{(i)}(\theta) = \underset{q(\theta)\in\mathcal{Q}}{\operatorname{argmax}} \int q(\theta) \ln \frac{q^{(i-1)}(\theta) p(\mathcal{D}_{b_i}|\theta)}{q(\theta) t_{b_i}^{(i-1)}(\theta)} d\theta$    (2.4)
6     Update the approximate likelihood:
          $t_{b_i}^{(i)}(\theta) = \frac{q^{(i)}(\theta)}{q^{(i-1)}(\theta)} t_{b_i}^{(i-1)}(\theta)$    (2.5)
7 end
```

Algorithm 1 describes the PVI algorithm in detail, which is a perfect fit for the federated learning setting. Suppose there are M clients, each of which is allocated a data group $\mathcal{D}_m$. Each client is responsible for storing and updating its associated approximate likelihood $t_m(\theta)$. On the other hand, a central server stores and updates the approximate posterior $q(\theta)$, and also communicates it to the clients.

At each iteration, an idle client receives the current posterior from the server, optimizes the local free energy (equation 2.4), computes the change in the local approximate likelihood $\Delta_m(\theta) = t_m^{(\text{new})}(\theta)\,/\,t_m^{(\text{old})}(\theta)$, and sends this quantity to the server. Note that even though the clients optimize the local free energies, they do not update $q(\theta)$ directly. Instead, the server maintains a queue of approximate likelihood updates, and applies them to the approximate posterior one by one: $q^{(\text{new})}(\theta) = q^{(\text{old})}(\theta)\,\Delta_m(\theta)$ (Bui et al., 2018).

The central step in algorithm 1 is equation 2.4, which optimizes the local free energy. Let's look into this step in more detail. To begin with, we provide the definition of local free energy.

Definition 2.1.1 (Local Free Energy). *The local free energy at iteration i with index b_i is defined as:*

$$\mathcal{F}_{b_i}^{(i)}\big(q(\theta)\big) = \int q(\theta) \ln \frac{q^{(i-1)}(\theta) p(\mathcal{D}_{b_i}|\theta)}{q(\theta) t_{b_i}^{(i-1)}(\theta)}\, d\theta \tag{2.6}$$

Bui et al. (2018) showed that maximizing the local free energy (equation 2.4) is equivalent to the following KL optimization:

$$q^{(i)}(\theta) = \underset{q(\theta)\in\mathcal{Q}}{\text{argmin}}\ \text{KL}\big(q(\theta) \parallel \widehat{p}_{b_i}^{(i)}(\theta)\big) \tag{2.7}$$

where $\widehat{p}_{b_i}^{(i)}(\theta)$ is referred to as the tilted distribution and is defined as:

$$\widehat{p}_{b_i}^{(i)}(\theta) = \frac{1}{\mathcal{Z}_{b_i}^{(i)}} \frac{q^{(i-1)}(\theta)}{t_{b_i}^{(i-1)}(\theta)}\, p(\mathcal{D}_{b_i}|\theta) \tag{2.8}$$

$$= \frac{1}{\mathcal{Z}_{b_i}^{(i)}} \big(p(\theta) \prod_{m\neq b_i} t_m^{(i-1)}(\theta)\big)\, p(\mathcal{D}_{b_i}|\theta) \tag{2.9}$$

The tilted distribution can be viewed as a local estimate of the true posterior distribution for client b_i, since it comprises the prior distribution, the true local likelihood for client b_i, and the approximate likelihood terms for all other clients. The local free energy can be maximized in several ways, such as analytical updates or gradient based methods (Bui et al., 2018). We will come back to this point in later chapters when we start to look at specific models.

Lastly, we briefly compare the PVI algorithm with standard global variational inference, and point out several key properties of PVI that reveal the relationship to global VI. In standard variational inference, a global free energy with respect to the entire training data set is maximized. Here is the definition of global free energy.

Definition 2.1.2 (Global Free Energy)**.** *The global free energy with respect to the whole data set is defined as:*

$$\mathcal{F}\big(q(\theta)\big) = \int q(\theta) \ln \frac{p(\theta)\prod_{m=1}^{M} p(\mathcal{D}_m|\theta)}{q(\theta)}\, d\theta \tag{2.10}$$

Maximizing the global free energy above is equivalent to minimizing the KL divergence between the approximate posterior and the true posterior, $\mathrm{KL}\big(q(\theta) \parallel p(\theta|\mathcal{D})\big)$ (Bishop, 2006).

The following properties (Bui et al., 2018) reveal the connection between PVI and global variational inference.

Theorem 2.1.1. *Let $q^*(\theta) = p(\theta)\prod_{m=1}^{M} t_m^*(\theta)$ be the approximate posterior at convergence of the PVI algorithm. We have:*

1. *The sum of the local free energies is equal to the global free energy:*

$$\sum_{m=1}^{M} \mathcal{F}_m\big(q^*(\theta)\big) = \mathcal{F}\big(q^*(\theta)\big) \tag{2.11}$$

2. *Optimizing the local free energies optimizes the global free energy*

$$q^*(\theta) = \operatorname*{argmax}_{q(\theta)\in\mathcal{Q}} \mathcal{F}_m\big(q(\theta)\big) \textit{ for all } m \;\Rightarrow\; q^*(\theta) = \operatorname*{argmax}_{q(\theta)\in\mathcal{Q}} \mathcal{F}\big(q(\theta)\big) \tag{2.12}$$

The above properties state that the approximate posterior at convergence of the PVI algorithm recovers a global variation inference solution (both the optimal $q(\theta)$ and the global free energy at this optimum), so the approximation achieved by PVI is as good as that achieved by standard global VI (Bui et al., 2018).

2.2 Differential Privacy

Differential privacy (DP) (Dwork et al., 2006; Dwork and Roth, 2014) constitutes a powerful framework that prevents breaching of data subject privacy from the output of a computation. It provides a mathematical foundation for privacy, and is able to quantify the level of privacy guarantees that some methods provide. This quantification is particularly helpful not only for algorithm designers, who are interested in investigating the trade-off between statistical performance and level of privacy guarantees of different algorithms, but also for clients seeking protection of their data, who are able to compare the level of protection offered by rival algorithm providers (Dwork and Roth, 2014).

Let's look into the mathematical details of differential privacy. To begin with, we provide the definition of differential privacy.

Definition 2.2.1 (ε-Differential Privacy). *A randomized algorithm $\mathcal{A}$ is said to be ε-differentially private if for all pairs of data sets $(\mathcal{D}, \mathcal{D}')$, which differ in one data point only, and for any possible subset of outputs $\mathcal{S}$, the following inequality holds:*

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{S}) \leq e^{\varepsilon} \Pr(\mathcal{A}(\mathcal{D}') \in \mathcal{S}) \tag{2.13}$$

Since this definition is symmetric across data sets, for any pair of data sets $(\mathcal{D}, \mathcal{D}')$ differing only in one data point, the inequality in the definition can be rewritten as:

$$e^{-\varepsilon} \leq \frac{\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{S})}{\Pr(\mathcal{A}(\mathcal{D}') \in \mathcal{S})} \leq e^{\varepsilon} \tag{2.14}$$

This inequality means that when two data sets $(\mathcal{D}, \mathcal{D}')$ are similar (i.e. differ in one data point), the two output probability densities should also be similar (the ratio of two densities is close to one). In this way, it's difficult to distinguish between the above two data sets given outputs $\big(\mathcal{A}(\mathcal{D}), \mathcal{A}(\mathcal{D}')\big)$, and thus algorithm $\mathcal{A}$ provides privacy guarantees to data sets $(\mathcal{D}, \mathcal{D}')$.

The positive privacy parameter ε measures the strength of privacy guarantee with smaller values corresponding to stronger privacy (Dwork and Roth, 2014). However, there is still no rigorous method for choosing and interpreting ε. Hsu et al. (2014) proposed a simple economic model based on estimating the monetary cost of leaking private data, which enables users of differential privacy to choose the privacy parameter ε in a principled way. According to their findings, the value of ε usually does not exceed 10, but this value might not be appropriate for machine learning models.

In practice, differential privacy operates by adding noise to the data-dependent terms in algorithms and limiting the amount of information that can be taken from a single data point. In this way, an adversary will be not able to tell whether a particular output of the algorithm is because of noise or because of the contribution from a specific data point. On the other hand, a client can also claim that a particular output of the algorithm is because of noise instead of the contribution from a specific data point, which relates to the concept of plausible deniability (Dwork and Roth, 2014).

Differential privacy is immune to *post-processing*: An adversary, without additional knowledge (e.g. information about a specific data point, or aggregate information of a collection of data points) about the private database, cannot compute a function of the output of a differentially private algorithm and make it less differentially private (Dwork and Roth, 2014). That is, data leak is not possible for a differentially private algorithm if an adversary only has access to the output of the algorithm, no matter what auxiliary information is available.

ε-differential privacy defined above, also known as *pure DP*, is sometimes too inflexible and a relaxed version called (ε, δ)-differential privacy is often used instead. It is defined as follows:

Definition 2.2.2 ((ε, δ)-Differential Privacy). *A randomized algorithm $\mathcal{A}$ is said to be (ε, δ)-differentially private if for all pairs of data sets $(\mathcal{D}, \mathcal{D}')$, which differ in one data point only, and for any possible subset of outputs $\mathcal{S}$, the following inequality holds:*

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{S}) \leq e^{\varepsilon} \Pr(\mathcal{A}(\mathcal{D}') \in \mathcal{S}) + \delta \tag{2.15}$$

Similar to ε, δ is another privacy parameter which measures the strength of privacy guarantee, with smaller values corresponding to stronger privacy. When $\delta = 0$, an (ε, δ)-differentially private algorithm is equivalent to a pure ε-differentially private algorithm. Dwork and Roth (2014) showed that (ε, δ)-DP provides a probabilistic ε-DP guarantee with probability $1 - \delta$.

Next, we introduce how to make an algorithm differentially private. Before that, we provide the definition of ℓ_2-sensitivity, which is an important quantity for functions operating on data sets.

Definition 2.2.3 (ℓ_2-Sensitivity). *The ℓ_2-sensitivity of a function, $\mathcal{D} \mapsto f(\mathcal{D}) \in \mathbb{R}^n$, is denoted as $\Delta_2(f)$ and is defined as:*

$$\Delta_2(f) = \max_{\mathcal{D}, \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2 \tag{2.16}$$

where $\mathcal{D}$ and $\mathcal{D}'$ differ in one data point only.

For a function with a given ℓ_2-sensitivity, the Gaussian mechanism can be used to make this function (ε, δ)-differentially private (Dwork and Roth, 2014).

Theorem 2.2.1 (Gaussian Mechanism). *Let $\mathcal{D} \mapsto f(\mathcal{D}) \in \mathbb{R}^n$ be a function with ℓ_2-sensitivity $\Delta_2(f)$ and $\varepsilon \in (0, 1)$. Adding noise $\eta_i \sim \mathcal{N}(0, \sigma^2)$ to each of the n components of the output* *guarantees $f(\mathcal{D}) + \boldsymbol{\eta}$ to be (ε, δ)-differentially private when the following inequality holds:*

$$\sigma^2 > 2\ln(1.25/\delta)\,\Delta_2^2(f)/\varepsilon^2 \tag{2.17}$$

In many machine learning models, a particular algorithm is often repeatedly applied to a data set. For example, gradient descent optimization algorithms repeatedly calculate the gradient of a loss function over collections of data points, and use this gradient to adjust model parameters so that the loss is minimized. One of the very useful features of differential privacy compared to many other privacy formulations is that it is able to *compose* individual privacy guarantees provided by repeatedly applying an algorithm to the same data set into an overall privacy guarantee (note that a privacy guarantee refers to a pair of (ε, δ) values). Intuitively, using an algorithm on a data set multiple times will weaken our privacy guarantee because of the potential of each application to leak more information (Jälkö et al., 2016).

2.2.1 Moments Accountant

When individual privacy guarantees are composed, there are many mechanisms which bound the values of ε and δ. In particular, Abadi et al. (2016) proposed the *moments accountant*, an advanced technique which is able to accumulate the privacy cost by tracking the moments of the privacy loss, and provide a tighter upper bounds for ε and δ. Here, we walk through the details of the moments accountant according to the original paper.

Definition 2.2.4 (Privacy Loss)**.** *For a stochastic algorithm $\mathcal{A}$, a pair of adjacent data sets $(\mathcal{D}, \mathcal{D}')$ (i.e. differ in one data point only), and some outcome $\mathcal{S}$, the privacy loss at $\mathcal{S}$ is defined as:*

$$c(\mathcal{S}; \mathcal{A}, \mathcal{D}, \mathcal{D}') = \ln \frac{\Pr[\mathcal{A}(\mathcal{D}) = \mathcal{S}]}{\Pr[\mathcal{A}(\mathcal{D}') = \mathcal{S}]} \tag{2.18}$$

This definition means that if the probabilities of an algorithm outcome are very different for adjacent data sets, observing this outcome reveals information about the data sets, and thus the privacy loss c should have large (absolute) value. Since the algorithm in the definition is stochastic, the privacy loss c is itself a random variable. Note that according to equation 2.14, an $(\varepsilon, 0)$ privacy guarantee means that the absolute value of the privacy loss random variable $|c|$ is always less than or equal to ε.

Definition 2.2.5 (λth Moment of $\mathcal{A}$)**.** *For a stochastic algorithm $\mathcal{A}$, a useful quantity is the maximum value of the log of the moment generating function of the privacy loss:*

$$\alpha_{\mathcal{A}}(\lambda) = \max_{\mathcal{D}, \mathcal{D}'} \ln \mathbb{E}_{\mathcal{S} \sim \mathcal{A}(\mathcal{D})} \left[\exp\left(\lambda c(\mathcal{S}; \mathcal{A}, \mathcal{D}, \mathcal{D}') \right) \right] \tag{2.19}$$

Theorem 2.2.2 (Properties of $\alpha_{\mathcal{A}}(\lambda)$)**.** $\alpha_{\mathcal{A}}(\lambda)$ *has two important properties that are useful for the moments accountant.*

1. ***Composability:*** *If the stochastic algorithm $\mathcal{A}$ consists of a sequence of adaptive steps $\mathcal{M}_1, ..., \mathcal{M}_k$, then for any λ:*

$$\alpha_{\mathcal{A}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{M}_i}(\lambda) \tag{2.20}$$

2. ***Tail bound:*** *For any $\varepsilon > 0$ and λ, the stochastic algorithm $\mathcal{A}$ is (ε, δ)-differentially private for*

$$\delta \leq \exp(\alpha_{\mathcal{A}}(\lambda) - \lambda\varepsilon) \tag{2.21}$$

For each steps $\mathcal{M}_1, ..., \mathcal{M}_k$ of the algorithm $\mathcal{A}$, the moments accountant method computes $\alpha_{\mathcal{M}_i}(\lambda)$ for several values of λ, and then applies the composability property to bound the moments of the algorithm overall $\alpha_{\mathcal{A}}(\lambda)$. In typical applications of the moments accountant method, a target value for either ε or δ is first fixed, and then the tail bound property is used to compute the best possible value of the other privacy parameter (Abadi et al., 2016). Rewritten from equation 2.21, the following equations are used compute the other privacy parameter:

$$\delta = \min_{\lambda} \exp(\alpha_{\mathcal{A}}(\lambda) - \lambda\varepsilon) \tag{2.22}$$

$$\varepsilon = \min_{\lambda} \frac{1}{\lambda}(\alpha_{\mathcal{A}}(\lambda) - \ln\delta) \tag{2.23}$$

Each λ value provides an upper bound on the privacy parameter, and we take the minimum of ε or δ across all λ values because smaller values of ε and δ are equivalent to stronger privacy guarantees.

The main challenge that remains is to bound the value $\alpha_{\mathcal{M}_i}(\lambda)$ for each step. In the case of a Gaussian mechanism with random sampling, it is possible to estimate those moments. Let $\mathcal{D} = \{x_i\}$ and $\mathcal{D}' = \mathcal{D} \cup x'$ where $x \in \mathcal{X}$. Let $f : \mathcal{X} \to \mathbb{R}^n$ with $\|f(\cdot)\|_2 \leq \Delta$. Consider the following Gaussian mechanism:

$$\mathcal{A}(\mathcal{D}) = \sum_{i \in \mathcal{I}} f(x_i) + \Delta\sigma\boldsymbol{\eta} \tag{2.24}$$

where $\eta_i \sim \mathcal{N}(0,1)$ for $i = 1, ..., n$ and $\mathcal{I}$ is a subset of indices in which each index is chosen independently with probability q (Abadi et al., 2016).

Without loss of generality, suppose $f(x_i) = \mathbf{0}$ and $f(x') = \Delta \cdot \boldsymbol{e}_1$ where $\boldsymbol{e}_1$ is a unit vector representing the first axis of the coordinate system. Then $\mathcal{A}(\mathcal{D})$ and $\mathcal{A}(\mathcal{D}')$ are distributed identically except on the first axis. Focusing on this axis, since the probability of choosing x' is q, the output probability densities of the mechanism are (Abadi et al., 2016):

$$p(\mathcal{A}(\mathcal{D})_1 = z) \sim \mu_0(z) = \mathcal{N}(z|0, \Delta^2\sigma^2) \tag{2.25}$$

$$p(\mathcal{A}(\mathcal{D}')_1 = z) \sim \mu_1(z) = q \cdot \mathcal{N}(z|\Delta, \Delta^2\sigma^2) + (1-q) \cdot \mathcal{N}(z|0, \Delta^2\sigma^2) \tag{2.26}$$

Then, by definition of the λth moment, $\alpha_{\mathcal{A}}(\lambda)$ can be computed using the following equations:

$$\alpha_{\mathcal{A}}(\lambda) = \ln \max(E_1, E_2) \tag{2.27}$$

$$E_1 = \mathbb{E}_{z \sim \mu_0}\left[(\frac{\mu_0(z)}{\mu_1(z)})^{\lambda} \right] \tag{2.28}$$

$$E_2 = \mathbb{E}_{z \sim \mu_1}\left[(\frac{\mu_1(z)}{\mu_0(z)})^{\lambda} \right] \tag{2.29}$$

where E_1 and E_2 can be computed by numerical integration. Note that computing the values of ε and δ does not depend on the data used for the mechanism, so they can be computed in advance given fixed values of Δ, q, and target ε or δ (Abadi et al., 2016).

2.2.2 Differentially Private Stochastic Gradient Descent

Abadi et al. (2016) proposed the *differentially private stochastic gradient descent (DP-SGD)* algorithm, which is outlined in algorithm 2. The Gaussian mechanism with random sampling is applied to *stochastic gradient descent (SGD)* to minimize some loss function. Specifically, at each step of the SGD, we compute the gradient for a random subset of data points, clip the ℓ_2 norm of each gradient, add noise to the average of all gradients in order to protect privacy, and perform the descent step.

The Gaussian mechanism can be successfully applied, because the gradient clipping step bounds the ℓ_2-sensitivity of the gradient function. The clipping bound C is an important hyperparameter: if the clipping bound is too small, the average clipped gradient may be very different from the true gradient; if the clipping bound is too large, we will add too much noise to the average gradient, since the noise scales with σC. Abadi et al. (2016) suggested that C can be chosen as the median of the norms of the unclipped gradients over the course of training. Moreover, Abadi et al. (2016) suggested learning rate should be adaptive, starting at a large value and reducing over time. In addition, regarding the lot size L, larger L will reduce the relative effect of the added noise, while smaller L will result in less privacy cost.

Abadi et al. (2016) commented that the best lot size is roughly $\sqrt{N}$ where N is the number of training data points.

Abadi et al. (2016) originally used DP-SGD in deep neural networks, but it can be used in many other models to create differentially private algorithms. One example is that Jälkö et al. (2016) used it to perform variational inference for non-conjugate models.

Algorithm 2: Differentially Private Stochastic Gradient Descent (DP-SGD)

```
   Input: data set $\mathcal{D} = \{x_1, ..., x_N\}$, loss function $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_i \mathcal{L}(\boldsymbol{\theta}, x_i)$
   Parameters: learning rate $\alpha_t$, clipping bound $C$, DP noise scale $\sigma$, lot size $L$, number
               of iterations $T$
1  Initialize $\boldsymbol{\theta}_0$ randomly
2  for $t = 1, ..., T$ do
3      Take a random sample $L_t$ with sampling probability $q = L/N$
4      foreach $i \in L_t$ do
5          Compute gradient: $\boldsymbol{g}_t(x_i) = \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\boldsymbol{\theta}_{t-1}, x_i)$
6          Clip gradient: $\bar{\boldsymbol{g}}_t(x_i) = \boldsymbol{g}_t(x_i) / \max(1, \frac{\|\boldsymbol{g}_t(x_i)\|_2}{C})$
7      end
8      Add noise: $\tilde{\boldsymbol{g}}_t = \frac{1}{L_t}\left[\sum_i \bar{\boldsymbol{g}}_t(x_i) + \sigma C \boldsymbol{z}\right]$ where $z_i \sim \mathcal{N}(0,1)$
9      Descent: $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \tilde{\boldsymbol{g}}_t$
10 end
   Output: $\boldsymbol{\theta}_T$ and compute the overall privacy cost $(\varepsilon, \delta)$ using the moments accountant
```

We have now covered all the necessary background on federated learning and differential privacy. In the next chapter, we will extend the Partitioned Variational Inference algorithm using differential privacy techniques.

Chapter 3

Differentially Private Partitioned Variational Inference

In this chapter, we first specify the context of the Partitioned Variational Inference setting that we seek to protect, clarifying all assumptions. Then we elaborate on two types of *Differentially Private Partitioned Variational Inference (DP-PVI)* algorithms, namely data-point level DP-PVI and data-set level DP-PVI.

3.1 Context

Recalling the PVI setting, a number of clients, each of which has a local data shard we seek to protect, interact with a central parameter server to collaboratively train a shared machine learning model. An adversary tries to use obtainable information of this machine learning model to learn private information about the training data. The full context makes the following assumptions (Sharma, 2019):

- The adversary is able to access the approximate posterior $q(\theta)$ stored at the central server all the time (both during the course of training and after training).
- The adversary is able to intercept messages between the central server and each client.
- The adversary is able to pretend as a client, or force an existing client to leak their local data and / or add additional data used for training to this client.
- The adversary is able to pretend as the central server, sending messages to each client and reading incoming messages from each client.

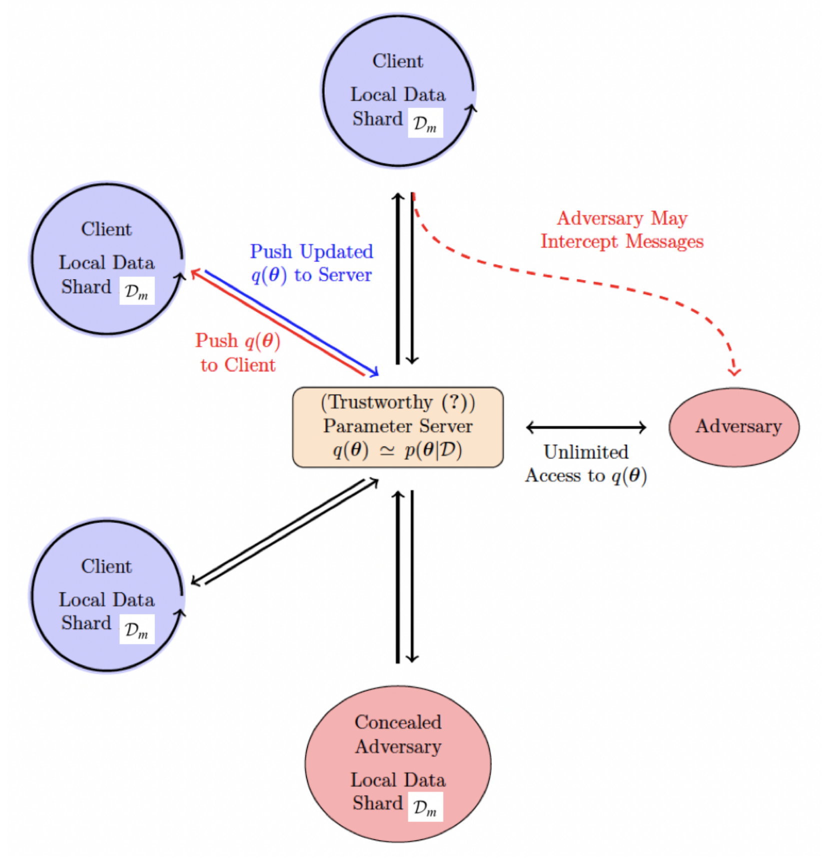


Fig. 3.1 Full context of the PVI setting that we seek to protect (Sharma, 2019).

3.2 Data-point Level DP-PVI

Data-point level DP-PVI can be constructed simply by applying a differentially private mechanism to perform the local free energy optimization step (equation 2.4) at each client. For example, DP-SGD (algorithm 2) can be used by setting the loss function to be the negative local free energy (Sharma, 2019). Note that since DP-SGD minimizes the loss function, we use negative local free energy here.

At each client $m = 1, \ldots, M$, DP-SGD limits the contribution of each data point in the local data shard $\mathcal{D}_m$ to the update of the approximate posterior $q(\theta)$, and adds noise which scales with the maximum magnitude of this contribution. Therefore, since any external communications (i.e. the approximate posterior updates) between the client and the parameter server is privacy preserving, each client is protected with an independent privacy barrier. Figure 3.2 illustrates the privacy barriers achieved by the data-point level DP-PVI algorithm.

Data-point level DP-PVI has two main advantages. One is that the clients do not need to encrypt their outgoing messages or verify the parameter server is trustworthy. The other is that each client is able to tune its individual privacy settings according to its own local data set and the level of privacy guarantees it aims (Sharma, 2019).

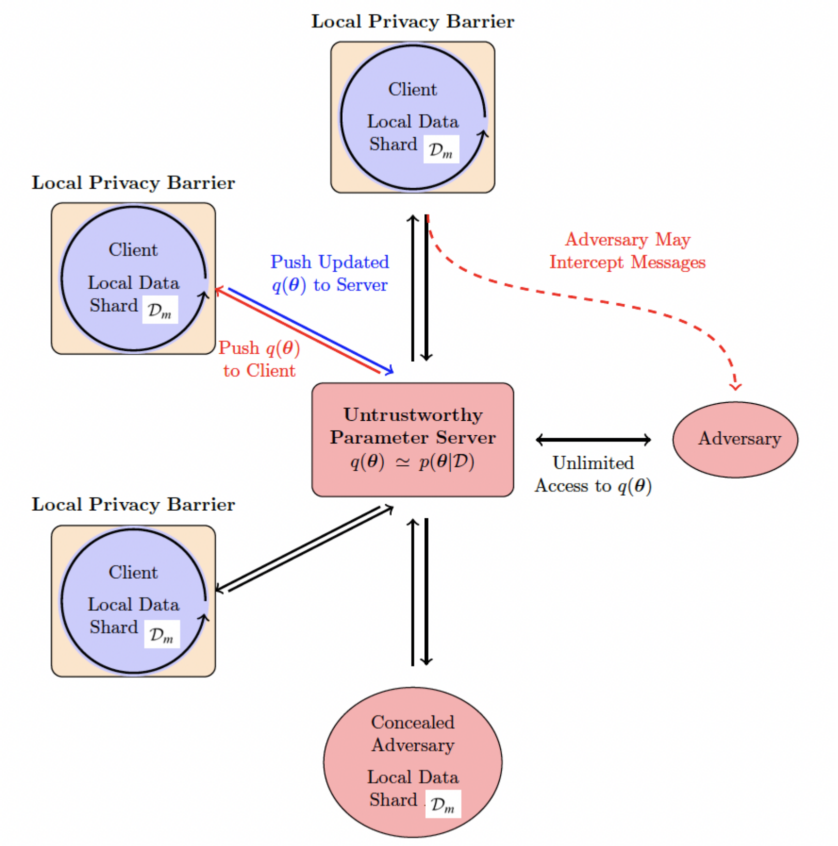

Local Privacy Barrier
Client
Local Data Shard $\mathcal{D}_m$
Local Privacy Barrier
Client
Local Data Shard $\mathcal{D}_m$
Push Updated $q(\boldsymbol{\theta})$ to Server
Push $q(\boldsymbol{\theta})$ to Client
Adversary May Intercept Messages
Untrustworthy Parameter Server $q(\boldsymbol{\theta}) \simeq p(\boldsymbol{\theta}|\mathcal{D})$
Unlimited Access to $q(\boldsymbol{\theta})$
Adversary
Local Privacy Barrier
Client
Local Data Shard $\mathcal{D}_m$
Concealed Adversary
Local Data Shard $\mathcal{D}_m$

Fig. 3.2 Privacy barriers achieved by the data-point level DP-PVI algorithm (Sharma, 2019).

3.3 Data-set Level DP-PVI

Recalling the fundamental definition of differential privacy (definition 2.2.1), we can view each client's local data shard $\mathcal{D}_m$ as one data point in this definition. In this way, data-set level DP-PVI can be constructed by limiting the contribution of each local data shard to the update of the approximate posterior $q(\theta)$ and then adding corrupting noise.

Suppose that the approximate posterior is parameterized with parameters $\boldsymbol{\lambda}$ and is denoted as $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$. Note that for now on, we will use vector notations for $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$, since both of them usually consist of more than one components for many probabilistic models used in practice.

Algorithm 3: Data-set Level DP-PVI (Problematic) (Sharma, 2019)

```
Input: data partition $\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_M\}$, prior $\boldsymbol{\lambda}^{(p)}$, learning rate $\alpha \in [0,1]$, clipping
       bound $C$, DP noise scale $\sigma$
1  Initialize the approximate likelihoods:
       $$t_m^{(0)}(\boldsymbol{\theta}) = 1 \text{ for all } m = 1, 2, ..., M \qquad (3.1)$$
2  Initialize parameters of the approximate posterior:
       $$\boldsymbol{\lambda}^{(0)} = \boldsymbol{\lambda}^{(p)} \qquad (3.2)$$
3  for $i = 1, 2, ...$ until convergence do
4      for $m = 1, 2, ..., M$ do
5          Compute new parameters for this client:
               $$\boldsymbol{\lambda}_m = \underset{\boldsymbol{\lambda}}{\text{argmax}} \int q(\boldsymbol{\theta}|\boldsymbol{\lambda}) \ln \frac{q(\boldsymbol{\theta}|\boldsymbol{\lambda}^{(i-1)}) p(\mathcal{D}_m|\boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\lambda}) t_m^{(i-1)}(\boldsymbol{\theta})} d\boldsymbol{\theta} \qquad (3.3)$$
6          $\Delta\boldsymbol{\lambda}_m = \boldsymbol{\lambda}_m - \boldsymbol{\lambda}^{(i-1)}$
7          Clip and corrupt update:
               $$\tilde{\Delta}\boldsymbol{\lambda}_m = \alpha \cdot \left[ \frac{\Delta\boldsymbol{\lambda}_m}{\max(1, \|\Delta\boldsymbol{\lambda}_m\|_2/C)} + \frac{\sigma C}{\sqrt{M}} \boldsymbol{z} \right] \text{ where } z_k \overset{iid}{\sim} \mathcal{N}(0,1) \qquad (3.4)$$
8          $\boldsymbol{\lambda}_m = \boldsymbol{\lambda}^{(i-1)} + \tilde{\Delta}\boldsymbol{\lambda}_m$
9          Update the approximate likelihood:
               $$t_m^{(i)}(\boldsymbol{\theta}) = \frac{q(\boldsymbol{\theta}|\boldsymbol{\lambda}_m)}{q(\boldsymbol{\theta}|\boldsymbol{\lambda}^{(i-1)})} t_m^{(i-1)}(\boldsymbol{\theta}) \qquad (3.5)$$
10     end
11     Compute new global parameters:
           $$\boldsymbol{\lambda}^{(i)} = \boldsymbol{\lambda}^{(i-1)} + \sum_{m=1}^{M} \tilde{\Delta}\boldsymbol{\lambda}_m \qquad (3.6)$$
12     Update privacy cost using the moments accountant
13 end
```

Sharma (2019) introduced a data-set level DP-PVI algorithm, which is outlined in algorithm 3. In this algorithm, at each iteration, each client computes a partial update of the approximate posterior, clips and corrupts this update with noise, and uses this partial update to refine the local approximate likelihood term. On the other hand, at each iteration, the central parameter server uses the sum of all partial updates from each client to update the approximate posterior. Combining equations 3.4 and 3.6, the effective update rule of the parameter server is the following:

$$\boldsymbol{\lambda}^{(i)} = \boldsymbol{\lambda}^{(i-1)} + \alpha \cdot \left[\sum_{m=1}^{M} \frac{\Delta \boldsymbol{\lambda}_m}{\max(1, \|\Delta \boldsymbol{\lambda}_m\|_2 / C)} \right] + \sigma C \boldsymbol{z} \quad \text{where} \quad z_k \overset{iid}{\sim} \mathcal{N}(0,1) \tag{3.7}$$

However, the above data-set level DP-PVI algorithm has technical problems. At each iteration, each client has direct access to a proportion of the overall noise $\sigma C \boldsymbol{z}$ which is added, but at the next iteration, each client computes a new update, perfectly deleting the local approximate likelihood term. Thus, this new update from the client releases information about the noise, which is a violation to post-processing of differential privacy. In addition, the privacy accounting method of the above algorithm is problematic. This algorithm distributes the overall noise of a fixed standard deviation across M clients, and computes the privacy cost in a way that as M increases, there is no additional privacy cost.

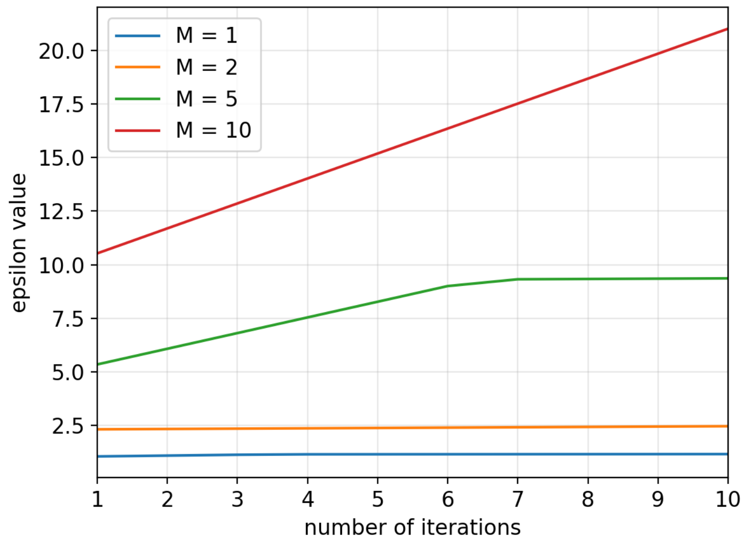


Fig. 3.3 Privacy costs (ε values) for several different numbers of clients.

Figure 3.3 illustrates how the privacy accounting method in algorithm 3 is problematic. We fix clipping bound $C = 1$, DP noise scale $\sigma = 1$, and target $\delta = 10^{-4}$, and then compute the values of ε using the moments accountant. If algorithm 3 is performed, no matter what the value of M is, the privacy costs will always be the blue curve, which is equivalent to $M = 1$. However, the correct privacy costs for $M = 2, 5, 10$ are illustrated by the orange, green, and red curves. Actually, if the number of clients is M, the effective DP noise scale should be $\sigma = 1/\sqrt{M}$.

Now, we specify two ways of fixing the technical problems in algorithm 3. The first idea is to remove the $\sqrt{M}$ term in equation 3.4, and it becomes:

$$\tilde{\Delta}\boldsymbol{\lambda}_m = \alpha \cdot \left[\frac{\Delta\boldsymbol{\lambda}_m}{\max(1, \|\Delta\boldsymbol{\lambda}_m\|_2/C)} + \sigma C \boldsymbol{z} \right] \text{ where } z_k \overset{iid}{\sim} \mathcal{N}(0,1) \tag{3.8}$$

At the central parameter server, the overall noise added will become $\sqrt{M} \cdot \sigma C \boldsymbol{z}$. This means that if the number of clients M is large, there will be a lot more noise added in each iteration, which will affect the performance of the algorithm.

Another idea of fixing the technical problems in algorithm 3 is to let the central server add noise, which is outlined in algorithm 4. However, algorithm 4 is still problematic because each client produces a clipped update, and assumes that this update is applied precisely. As a result, the local approximate likelihood terms are never noisy, and this algorithm results in pathological random walk behavior.

For a more detailed explanation, we can assume that all clients have the correct local approximate likelihood terms, but the central server has the wrong value of $q(\boldsymbol{\theta})$ due to the noise added. In this case, each client refines it's local approximate likelihood, but since this term is already correct, there is no "restoring-force" pushing $q(\boldsymbol{\theta})$ back towards the true optimum. The lack of "restoring force" can be viewed as that the approximate likelihood terms and the global approximate posterior become out-of-sync. Since the local approximate likelihood terms are never noisy, $q(\boldsymbol{\theta}) \neq p(\boldsymbol{\theta}) \prod_{m=1}^{M} t_m(\boldsymbol{\theta})$.

Here, we propose a possible remedy for the pathological random walk behavior. For the following discussion, assume that each approximate likelihood term is chosen from the same conjugate exponential family. Therefore, we have:

$$\boldsymbol{\lambda}_q = \boldsymbol{\lambda}_p + \sum_m \boldsymbol{\lambda}_{t_m} \tag{3.16}$$

where $\boldsymbol{\lambda}_q$ represents the (natural) parameters of the approximate posterior, $\boldsymbol{\lambda}_p$ represents the (natural) parameters of the prior, and $\boldsymbol{\lambda}_{t_m}$ represents the (natural) parameters of each un-normalized approximate likelihood term.

After T iterations of applying noise at the central server, the global approximate posterior is given by $\boldsymbol{\lambda}_q^{(T)}$. Since the approximate likelihood terms are not corrupted by the noise, $\boldsymbol{\lambda}_q^{(T)}$

Algorithm 4: Data-set Level DP-PVI (Central Noise)

```
   Input: data partition $\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_M\}$, prior $\boldsymbol{\lambda}^{(p)}$, learning rate $\alpha \in [0,1]$, clipping
          bound $C$, DP noise scale $\sigma$
 1 Initialize the approximate likelihoods:
       $$t_m^{(0)}(\boldsymbol{\theta}) = 1 \text{ for all } m = 1,2,...,M \quad (3.9)$$
 2 Initialize parameters of the approximate posterior:
       $$\boldsymbol{\lambda}^{(0)} = \boldsymbol{\lambda}^{(p)} \quad (3.10)$$
 3 for $i = 1,2,...$ until convergence do
 4     for $m = 1,2,...,M$ do
 5         Compute new parameters for this client:
               $$\boldsymbol{\lambda}_m = \operatorname*{argmax}_{\boldsymbol{\lambda}} \int q(\boldsymbol{\theta}|\boldsymbol{\lambda}) \ln \frac{q(\boldsymbol{\theta}|\boldsymbol{\lambda}^{(i-1)}) p(\mathcal{D}_m|\boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\lambda}) t_m^{(i-1)}(\boldsymbol{\theta})} d\boldsymbol{\theta} \quad (3.11)$$
 6         $\Delta\boldsymbol{\lambda}_m = \boldsymbol{\lambda}_m - \boldsymbol{\lambda}^{(i-1)}$
 7         Clip and corrupt update:
               $$\tilde{\Delta}\boldsymbol{\lambda}_m = \alpha \cdot \frac{\Delta\boldsymbol{\lambda}_m}{\max(1, \|\Delta\boldsymbol{\lambda}_m\|_2/C)} \quad (3.12)$$
 8         $\boldsymbol{\lambda}_m = \boldsymbol{\lambda}^{(i-1)} + \tilde{\Delta}\boldsymbol{\lambda}_m$
 9         Update the approximate likelihood:
               $$t_m^{(i)}(\boldsymbol{\theta}) = \frac{q(\boldsymbol{\theta}|\boldsymbol{\lambda}_m)}{q(\boldsymbol{\theta}|\boldsymbol{\lambda}^{(i-1)})} t_m^{(i-1)}(\boldsymbol{\theta}) \quad (3.13)$$
10     end
11     Compute global noise:
           $$\tilde{\Delta}\boldsymbol{\lambda}_s^{(i)} = \alpha \cdot \sigma C z \text{ where } z_k \overset{iid}{\sim} \mathcal{N}(0,1) \quad (3.14)$$
12     Compute new global parameters:
           $$\boldsymbol{\lambda}^{(i)} = \boldsymbol{\lambda}^{(i-1)} + \sum_{m=1}^{M} \tilde{\Delta}\boldsymbol{\lambda}_m + \tilde{\Delta}\boldsymbol{\lambda}_s^{(i)} \quad (3.15)$$
13     Update privacy cost using the moments accountant
14 end
```

can be written as:

$$\boldsymbol{\lambda}_q^{(T)} = \boldsymbol{\lambda}_p + \sum_m \boldsymbol{\lambda}_{t_m}^{(T)} + \boldsymbol{z}^{(1)} + \cdots + \boldsymbol{z}^{(T)} \tag{3.17}$$

Now, since the approximate likelihood terms stored at the clients are wrong, we can reset those likelihood terms as follows:

$$\boldsymbol{\lambda}_{t_m}^{(T)} = \frac{\boldsymbol{\lambda}_q^{(T)} - \boldsymbol{\lambda}_p}{M} \tag{3.18}$$

where M is the number of clients.

This resetting approach does not violate differential privacy, because the approximate likelihood terms are set using public knowledge only and the privacy accounting does not need to be modified. After the resetting takes place, the noise will be incorporated into the local approximate likelihood terms, and then the clients will be able to restart refining these likelihoods, hopefully pushing $\boldsymbol{\lambda}_q$ back towards the global optimum.

The advantage of this approach is that it essentially reset the random walk noise every time the local approximate likelihoods are reset. It also has several disadvantages. Firstly, we need to figure out how to choose the interval between each resetting. Moreover, this approach is unlikely to work well for inhomogeneous data. Last but not least, it does not completely remove the pathological random walk behavior.

We have now constructed two types of DP-PVI algorithms. In the next chapter, we will apply DP-PVI algorithms to specific models and provide the results of our experiments.

Chapter 4

Experiments and Results

In this chapter, we apply DP-PVI algorithms to Bayesian multi-dimensional linear regression models. Specifically, we focus on the data-point level DP-PVI algorithm, applying DP-SGD to perform the local free energy optimization step in PVI. We mainly conduct experiments on mini-batching and adaptive clipping bounds, investigating how they affect the performance of the data-point level DP-PVI algorithm.

4.1 Preliminaries

4.1.1 Model Definition

We consider the following multi-dimensional linear regression model:

$$y = \boldsymbol{\theta}^\top \boldsymbol{x} + \varepsilon \quad \text{where} \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_D \end{bmatrix} \quad \varepsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma_e^2) \tag{4.1}$$

$\boldsymbol{\theta}$ are fixed unknown parameters of this probabilistic model, and σ_e is assumed to be known. The training data set $\mathcal{D}$ is partitioned into M groups of data points $\{\mathcal{D}_1, ..., \mathcal{D}_M\}$, where each $\mathcal{D}_m = \{(\boldsymbol{x}_m^{(n)}, y_m^{(n)})\}_{n=1}^{N_m}$. $\boldsymbol{\theta}$ has a Gaussian prior:

$$p(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_p, \sigma_p^2 \boldsymbol{I}) \tag{4.2}$$

The approximate likelihood $t_m(\boldsymbol{\theta})$ for each client m are un-normalized Gaussian distributions, and thus the approximate posterior $q(\boldsymbol{\theta})$ is also Gaussian. We aim to find a $q(\boldsymbol{\theta})$ which is a good approximation of the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$.

Exponential family distributions take the following form:

$$p(\boldsymbol{x}|\boldsymbol{\eta}) = h(\boldsymbol{x}) \exp\left(\boldsymbol{\eta}^\top \boldsymbol{T}(\boldsymbol{x}) - A(\boldsymbol{\eta})\right) \tag{4.3}$$

where $\boldsymbol{\eta}$ is the vector of *natural parameters* and $\boldsymbol{T}(\boldsymbol{x})$ is the vector of *sufficient statistics*. It is useful to express the multivariate Gaussian distribution as an exponential family distribution in terms of natural parameters:

$$\begin{aligned}
\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) &= \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \\
&= \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right] \\
&= \frac{1}{(2\pi)^{k/2}} \exp\left[-\frac{1}{2}\operatorname{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\boldsymbol{x}^\top) + \boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - (\frac{1}{2}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{2}\ln|\boldsymbol{\Sigma}|)\right] \\
&= \frac{1}{(2\pi)^{k/2}} \exp\left[-\frac{1}{2}\operatorname{vec}(\boldsymbol{\Sigma}^{-1})^\top \operatorname{vec}(\boldsymbol{x}\boldsymbol{x}^\top) + \boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - (\frac{1}{2}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{2}\ln|\boldsymbol{\Sigma}|)\right]
\end{aligned} \tag{4.4}$$

where $\operatorname{vec}(\cdot)$ is the vectorization operator, which converts a matrix into a column vector by stacking the columns of the matrix on top of one another. Comparing equation 4.4 to the form of exponential family distributions, we can get:

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ \operatorname{vec}(\boldsymbol{\Sigma}^{-1}) \end{bmatrix} \tag{4.5}$$

$$\boldsymbol{T}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} \\ -\frac{1}{2}\operatorname{vec}(\boldsymbol{x}\boldsymbol{x}^\top) \end{bmatrix} \tag{4.6}$$

$$A(\boldsymbol{\eta}) = \frac{1}{2}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{2}\ln|\boldsymbol{\Sigma}| \tag{4.7}$$

$$h(\boldsymbol{x}) = \frac{1}{(2\pi)^{k/2}} \tag{4.8}$$

Note that $\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ is known as the natural mean, and $\boldsymbol{\Sigma}^{-1}$ is known as the precision matrix.

This representation allows the products and quotients of multivariate Gaussian distributions to be written easily as follows:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\eta}_1) \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\eta}_2) \propto \mathcal{N}(\boldsymbol{x}|\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2) \tag{4.9}$$

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\eta}_1) / \mathcal{N}(\boldsymbol{x}|\boldsymbol{\eta}_2) \propto \mathcal{N}(\boldsymbol{x}|\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2) \tag{4.10}$$

4.1.2 Analytical Update Equations for PVI

For this multi-dimensional linear regression model, the local free energy maximization (equation 2.4) and the local approximate likelihood update (equation 2.5) in the PVI algorithm are analytical. Equation 2.7 rewrites the local free energy maximization as the KL minimization between $q(\boldsymbol{\theta})$ and the tilted distribution $\widehat{p}_m^{(i)}(\boldsymbol{\theta})$ where m is the client index. This KL divergence is minimized when $q(\boldsymbol{\theta})$ is exactly equal to $\widehat{p}_m^{(i)}(\boldsymbol{\theta})$.

The tilted distribution can be expressed as follows:

$$\begin{aligned}\widehat{p}_m^{(i)}(\boldsymbol{\theta}) &\propto \frac{q^{(i-1)}(\boldsymbol{\theta})}{t_m^{(i-1)}(\boldsymbol{\theta})} p(\mathcal{D}_m|\boldsymbol{\theta}) \\ &\propto \underbrace{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\eta}_q^{(i-1)} - \boldsymbol{\eta}_m^{(i-1)})}_{\text{“prior”}} \underbrace{p(\mathcal{D}_m|\boldsymbol{\theta})}_{\text{likelihood}} = \mathcal{N}(\boldsymbol{\theta}|\tilde{\boldsymbol{\eta}}) \end{aligned} \tag{4.11}$$

Since the above equation is equivalent to standard Bayesian linear regression, $\tilde{\boldsymbol{\eta}}$ can be computed analytically as follows:

$$\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta}_q^{(i-1)} - \boldsymbol{\eta}_m^{(i-1)} + \begin{bmatrix} \frac{\boldsymbol{X}_m^\top \boldsymbol{Y}_m}{\sigma_e^2} \\ \\ \text{diag}(\frac{\boldsymbol{X}_m^\top \boldsymbol{X}_m}{\sigma_e^2}) \end{bmatrix} \tag{4.12}$$

where

$$\boldsymbol{X}_m = \begin{bmatrix} — & \boldsymbol{x}_m^{(1)} & — \\ & \vdots & \\ — & \boldsymbol{x}_m^{(N_m)} & — \end{bmatrix} \quad \boldsymbol{Y}_m = \begin{bmatrix} y_m^{(1)} \\ \vdots \\ y_m^{(N_m)} \end{bmatrix} \tag{4.13}$$

where diag$(\cdot)$ takes a matrix as input and returns a column vector of the diagonal entries of this matrix. Note that since we perform *mean field* PVI here, we only keep diagonal entries of the precision matrices of the prior, the approximate posterior, and the approximate likelihood terms, so we have this $\text{diag}(\boldsymbol{X}_m^\top \boldsymbol{X}_m/\sigma_e^2)$ operation when computing $\tilde{\boldsymbol{\eta}}$.

Therefore, the local free energy maximization step is equivalent to setting $q^{(i)}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\tilde{\boldsymbol{\eta}})$ (i.e. $\boldsymbol{\eta}_q^{(i)} = \tilde{\boldsymbol{\eta}}$). Then the local approximate likelihood update (equation 2.5) is straightforward to compute in terms of natural parameters:

$$\boldsymbol{\eta}_m^{(i)} = \boldsymbol{\eta}_q^{(i)} - \boldsymbol{\eta}_q^{(i-1)} + \boldsymbol{\eta}_m^{(i-1)} \tag{4.14}$$

Note that since the approximate posterior $q(\boldsymbol{\theta})$ must normalize, there is no need to keep track of the normalizing constants of the approximate likelihood terms.

4.1.3 Gradient of Local Free Energy

Since data-point level DP-PVI algorithm is constructed by performing the local free energy optimization step (equation 2.4) in PVI (algorithm 1) using DP-SGD (algorithm 2), we now derive the gradients of the local free energy (definition 2.1.1) with respect to the mean vector $\boldsymbol{\mu}_q$ and the covariance matrix $\boldsymbol{\Sigma}_q$ of $q(\boldsymbol{\theta})$.

For client m, the local free energy can be expressed as follows:

$$\begin{aligned}\mathcal{F}_m^{(i)}\big(q(\boldsymbol{\theta})\big) &= \int q(\boldsymbol{\theta}) \ln \frac{q^{(i-1)}(\boldsymbol{\theta}) p(\mathcal{D}_m|\boldsymbol{\theta})}{q(\boldsymbol{\theta}) t_m^{(i-1)}(\boldsymbol{\theta})}\, d\boldsymbol{\theta} \\ &= \mathcal{H}[q] + \int q(\boldsymbol{\theta}) \ln p(\mathcal{D}_m|\boldsymbol{\theta})\, d\boldsymbol{\theta} + \int q(\boldsymbol{\theta}) \ln \Big(\mathcal{C} \cdot \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\eta}_q^{(i-1)} - \boldsymbol{\eta}_m^{(i-1)})\Big)\, d\boldsymbol{\theta}\end{aligned} \tag{4.15}$$

where $\mathcal{H}[q]$ is the *differential entropy* of q and $\mathcal{C}$ is some constant (which does not have a fixed value in the following analysis). The differential entropy for a multivariate Gaussian distribution has an analytical form (Bishop, 2006):

$$\mathcal{H}[q] = \frac{D+1}{2}(1 + \ln 2\pi) + \frac{1}{2} \ln |\boldsymbol{\Sigma}_q| \tag{4.16}$$

The second term in equation 4.15 can be written as follows:

$$\begin{aligned}&\int q(\boldsymbol{\theta}) \ln p(\mathcal{D}_m|\boldsymbol{\theta})\, d\boldsymbol{\theta} \\ &= \mathcal{C} - \frac{1}{2\sigma_e^2} \int q(\boldsymbol{\theta}) \Big[(\boldsymbol{Y}_m - \boldsymbol{X}_m\boldsymbol{\theta})^\top (\boldsymbol{Y}_m - \boldsymbol{X}_m\boldsymbol{\theta})\Big]\, d\boldsymbol{\theta} \\ &= \mathcal{C} - \frac{1}{2\sigma_e^2} \int q(\boldsymbol{\theta}) \Big[\boldsymbol{\theta}^\top \boldsymbol{X}_m^\top \boldsymbol{X}_m \boldsymbol{\theta} - 2\boldsymbol{Y}_m^\top \boldsymbol{X}_m \boldsymbol{\theta}\Big]\, d\boldsymbol{\theta} \\ &= \mathcal{C} - \frac{1}{2\sigma_e^2} \int q(\boldsymbol{\theta}) \Big[\mathrm{tr}(\boldsymbol{X}_m^\top \boldsymbol{X}_m \boldsymbol{\theta}\boldsymbol{\theta}^\top) - 2\boldsymbol{Y}_m^\top \boldsymbol{X}_m \boldsymbol{\theta}\Big]\, d\boldsymbol{\theta} \\ &= \mathcal{C} - \frac{1}{2\sigma_e^2} \int q(\boldsymbol{\theta}) \Big[\mathrm{vec}(\boldsymbol{X}_m^\top \boldsymbol{X}_m)^\top \mathrm{vec}(\boldsymbol{\theta}\boldsymbol{\theta}^\top) - 2\boldsymbol{Y}_m^\top \boldsymbol{X}_m \boldsymbol{\theta}\Big]\, d\boldsymbol{\theta} \\ &= \mathcal{C} - \frac{1}{2\sigma_e^2} \Big[\mathrm{vec}(\boldsymbol{X}_m^\top \boldsymbol{X}_m)^\top \mathrm{vec}(\boldsymbol{\Sigma}_q + \boldsymbol{\mu}_q\boldsymbol{\mu}_q^\top) - 2\boldsymbol{Y}_m^\top \boldsymbol{X}_m \boldsymbol{\mu}_q\Big] \\ &= \mathcal{C} - \frac{1}{2\sigma_e^2} \Big[\mathrm{vec}(\boldsymbol{X}_m^\top \boldsymbol{X}_m)^\top \mathrm{vec}(\boldsymbol{\Sigma}_q) + \boldsymbol{\mu}_q^\top \boldsymbol{X}_m^\top \boldsymbol{X}_m \boldsymbol{\mu}_q - 2\boldsymbol{Y}_m^\top \boldsymbol{X}_m \boldsymbol{\mu}_q\Big]\end{aligned} \tag{4.17}$$

where the first and second moments of the Gaussian distribution $q(\boldsymbol{\theta})$ have been directly substituted in.

Let $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Sigma}}$ denote the mean vector and covariance matrix of $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\eta}_q^{(i-1)} - \boldsymbol{\eta}_m^{(i-1)})$. Then the last term in equation 4.15 can be written as follows:

$$\begin{aligned}
&\int q(\boldsymbol{\theta}) \ln \Big(\mathcal{C} \cdot \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\eta}_q^{(i-1)} - \boldsymbol{\eta}_m^{(i-1)})\Big)\, d\boldsymbol{\theta} \\
&= \mathcal{C} - \frac{1}{2} \int q(\boldsymbol{\theta}) \left[(\boldsymbol{\theta} - \tilde{\boldsymbol{\mu}})^\top \tilde{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{\theta} - \tilde{\boldsymbol{\mu}})\right] d\boldsymbol{\theta} \\
&= \mathcal{C} - \frac{1}{2} \int q(\boldsymbol{\theta}) \left[\boldsymbol{\theta}^\top \tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\theta} - 2\tilde{\boldsymbol{\mu}}^\top \tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\theta}\right] d\boldsymbol{\theta} \\
&= \mathcal{C} - \frac{1}{2} \int q(\boldsymbol{\theta}) \left[\operatorname{tr}(\tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\theta}\boldsymbol{\theta}^\top) - 2\tilde{\boldsymbol{\mu}}^\top \tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\theta}\right] d\boldsymbol{\theta} \\
&= \mathcal{C} - \frac{1}{2} \int q(\boldsymbol{\theta}) \left[\operatorname{vec}(\tilde{\boldsymbol{\Sigma}}^{-1})^\top \operatorname{vec}(\boldsymbol{\theta}\boldsymbol{\theta}^\top) - 2\tilde{\boldsymbol{\mu}}^\top \tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\theta}\right] d\boldsymbol{\theta} \\
&= \mathcal{C} - \frac{1}{2} \left[\operatorname{vec}(\tilde{\boldsymbol{\Sigma}}^{-1})^\top \operatorname{vec}(\boldsymbol{\Sigma}_q + \boldsymbol{\mu}_q\boldsymbol{\mu}_q^\top) - 2\tilde{\boldsymbol{\mu}}^\top \tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\mu}_q\right] \\
&= \mathcal{C} - \frac{1}{2} \left[\operatorname{vec}(\tilde{\boldsymbol{\Sigma}}^{-1})^\top \operatorname{vec}(\boldsymbol{\Sigma}_q) + \boldsymbol{\mu}_q^\top \tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\mu}_q - 2\tilde{\boldsymbol{\mu}}^\top \tilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\mu}_q\right] \qquad (4.18)
\end{aligned}$$

Combining the above expressions, we can compute the gradients of the local free energy with respect to mean $\boldsymbol{\mu}_q$ and covariance $\boldsymbol{\Sigma}_q$ of $q(\boldsymbol{\theta})$ as follows:

$$\frac{\partial \mathcal{F}_m^{(i)}(q(\boldsymbol{\theta}))}{\partial \boldsymbol{\mu}_q} = -\frac{1}{\sigma_e^2}(\boldsymbol{X}_m^\top \boldsymbol{X}_m \boldsymbol{\mu}_q - \boldsymbol{X}_m^\top \boldsymbol{Y}_m) - (\tilde{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\mu}_q - \tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}) \qquad (4.19)$$

$$\frac{\partial \mathcal{F}_m^{(i)}(q(\boldsymbol{\theta}))}{\partial \boldsymbol{\Sigma}_q} = \frac{1}{2}\boldsymbol{\Sigma}_q^{-1} - \frac{1}{2\sigma_e^2}\boldsymbol{X}_m^\top \boldsymbol{X}_m - \frac{1}{2}\tilde{\boldsymbol{\Sigma}}^{-1} \qquad (4.20)$$

Note that since we seek to maximize the local free energy, DP-SGD should be performed using the negative of the above gradients. Moreover, DP-SGD need to clip the gradients of each data point, so we rewrite the above gradients in terms of each data point in the local data shard $\mathcal{D}_m = \{(\boldsymbol{x}_m^{(n)}, y_m^{(n)})\}_{n=1}^{N_m}$ as follows:

$$\boldsymbol{g}_{\boldsymbol{\mu}_q}\left(\boldsymbol{x}_m^{(n)}, y_m^{(n)}\right) = \frac{1}{\sigma_e^2}\left(\boldsymbol{x}_m^{(n)}\boldsymbol{x}_m^{(n)\top}\boldsymbol{\mu}_q - \boldsymbol{x}_m^{(n)} y_m^{(n)}\right) + \frac{1}{N_m}\left(\tilde{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\mu}_q - \tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}\right) \qquad (4.21)$$

$$\boldsymbol{g}_{\boldsymbol{\Sigma}_q}\left(\boldsymbol{x}_m^{(n)}, y_m^{(n)}\right) = \frac{1}{2\sigma_e^2}\boldsymbol{x}_m^{(n)}\boldsymbol{x}_m^{(n)\top} - \frac{1}{2N_m}\left(\boldsymbol{\Sigma}_q^{-1} - \tilde{\boldsymbol{\Sigma}}^{-1}\right) \qquad (4.22)$$

where $\boldsymbol{x}_m^{(n)}$ is a column vector. These gradients can then be used in DP-SGD.

4.1.4 Assessing Performance

To assess the performance of a DP-PVI algorithm, we use the KL divergence between the approximate posterior obtained from the algorithm and the posterior distribution computed non-privately using the entire combined data set and the exact analytical equations for Bayesian linear regression i.e. $\mathrm{KL}\big(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D})\big)$.

Intuitively, since differential privacy limits the contribution of any one single data point on the probabilistic model, we may expect that it reduces *overfitting*, which occurs a training procedure causes the model to correspond too closely or exactly to a particular set of data, and improves *model generalization*, the prediction performance on unseen data (Sharma, 2019). Therefore, the above KL divergence performance metric is not perfect. However, since we are using a simple linear probabilistic model and the data are also generated from a linear model, the KL divergence is a suitable performance metric here.

4.1.5 Data Generation

Each input $\boldsymbol{x}$ is generated as follows:

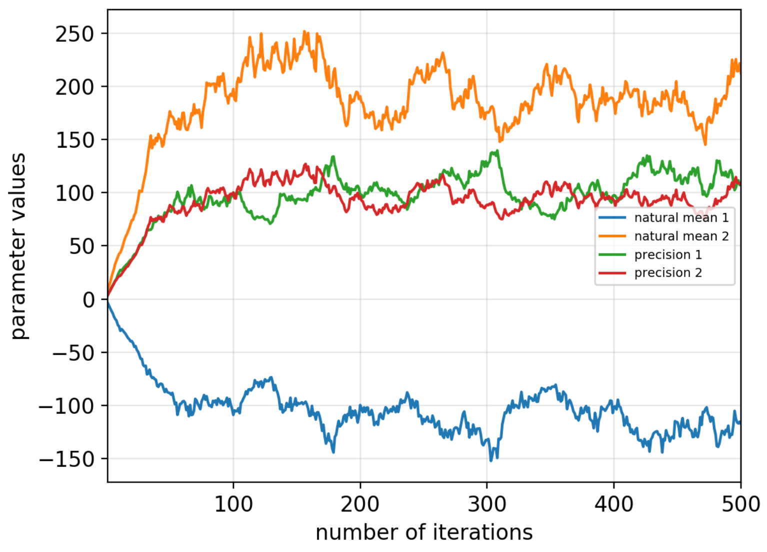
$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{bmatrix} \text{ where } x_d \overset{iid}{\sim} \mathcal{N}(0,1) \text{ for each } d = 1,...,D \tag{4.23}$$

Then given fixed values of $\boldsymbol{\theta}$ and σ_e, the local data shard at each client is generated using the model definition 4.1 with the above generated $\boldsymbol{x}$. Note that the data set we use in this project is homogeneous (i.e. the underlying data distribution for each client is the same).

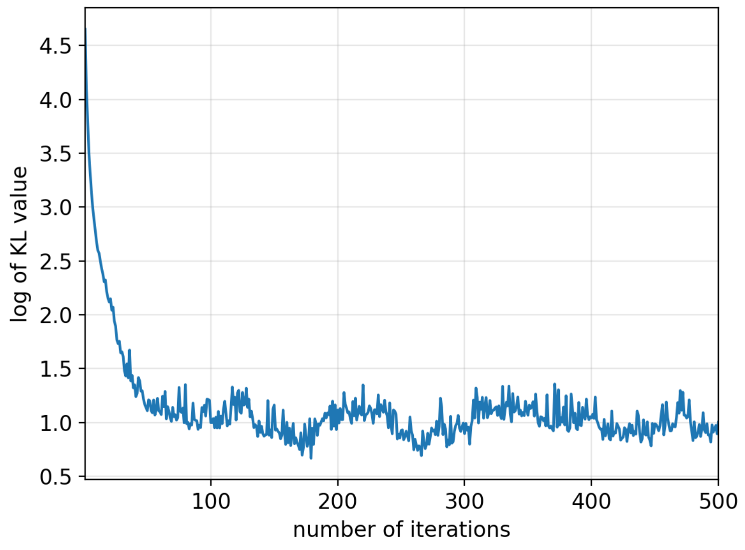
4.2 Data-point Level DP-PVI

4.2.1 DP-SGD

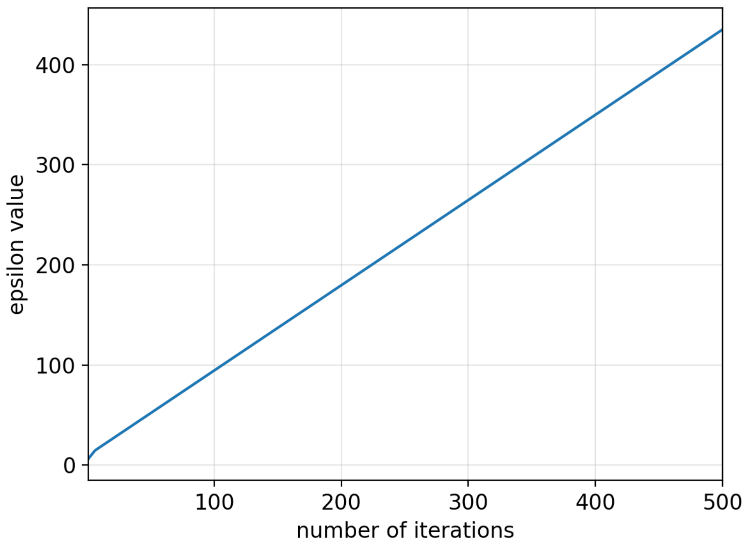
Recall that data-point level DP-PVI algorithm can be constructed by performing the local free energy optimization step (equation 2.4) in PVI (algorithm 1) using DP-SGD (algorithm 2). In our implementation, gradient descent is performed on the mean and the log of diagonal entries of the covariance matrix, in order to ensure that the variances of the marginals remain positive. Otherwise, large learning rate can cause negative variances, and thus the algorithm fails.



(a) Evolution of natural parameters of $q(\boldsymbol{\theta})$.



(b) Evolution of the log of $\mathrm{KL}\big(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D})\big)$.



(c) Evolution of the privacy parameter ε.

Fig. 4.1 Typical results of the data-point level DP-PVI algorithm implemented using DP-SGD.

Figure 4.1 illustrates a set of typical results of the data-point level DP-PVI algorithm implemented using DP-SGD. Here, we are using 5 clients each with a local data shard of 200 data points. The true parameters and hyperparameters are set as follows: $\boldsymbol{\theta} = [-1,2]^\top, \sigma_e = 3, \boldsymbol{\mu}_p = \mathbf{0}, \sigma_p = 1$, learning rate $\alpha = 0.1$, clipping bound $C = 5$, DP noise scale $\sigma = 1$, lot size $L = 20$, and target $\delta = 10^{-4}$. Each client performs 50 iterations of DP-SGD per one server update, and 500 iterations of server update have been applied.

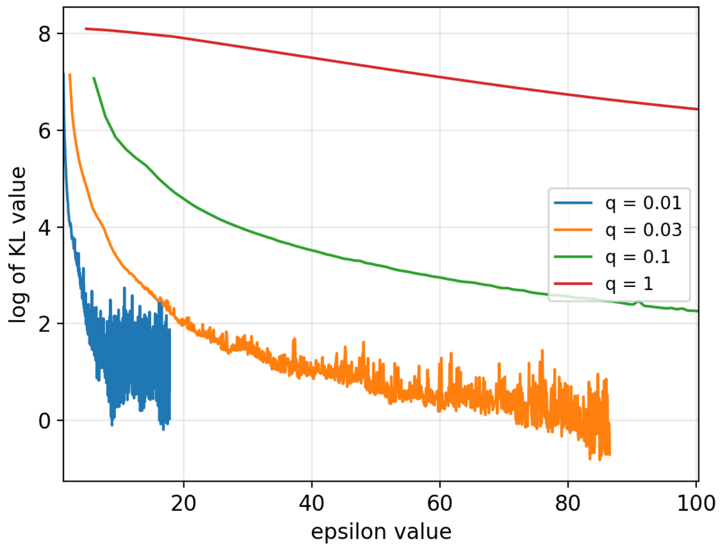
We can see from figure 4.1a that the natural parameters of the approximate posterior almost converge after 100 iterations of server update. After that, they slightly oscillate around due to the DP noise applied. Figure 4.1b shows that the log of the KL divergence between $q(\boldsymbol{\theta})$ and the true posterior is quite low, which reassures us that the performance of this DP-PVI algorithm is quite good. From figure 4.1c, we can see that the ε value goes above 400 after 500 iterations of server update, which far exceeds the maximum value $\varepsilon \approx 10$ found in Hsu et al. (2014). However, $q(\boldsymbol{\theta})$ almost converges after 100 iterations of server update, and the ε value at that point is about 100, which is not particularly bad.

4.2.2 Mini-batching

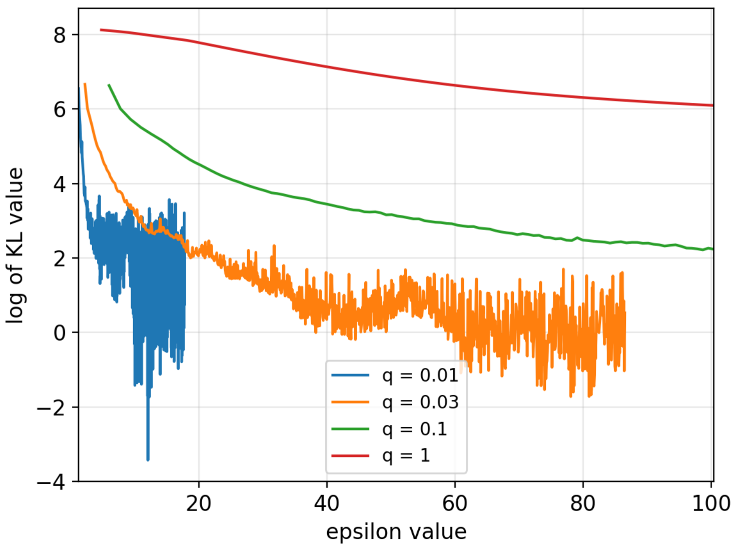
In this subsection, we investigate the effect of mini-batching (i.e. varying the sampling probability $q = L/N$ in DP-SGD) on the performance of the data-point level DP-PVI algorithm. Specifically, we aim to explore how the KL divergence performance metric and the privacy parameter ε change as q varies.

Since there are many hyperparameters we can tune in this algorithm, we fix $\boldsymbol{\theta} = [-1,2]^\top, \sigma_e = 1, \boldsymbol{\mu}_p = \mathbf{0}, \sigma_p = 1$, learning rate $\alpha = 0.03$, DP noise scale $\sigma = 1$, and target $\delta = 10^{-4}$. Since mini-batching of DP-SGD happens locally in each client, without loss of generality, we use one client with 1000 data points (i.e. $N = 1000$) here. The client performs 50 iterations of DP-SGD per one server update, and the server can perform a maximum of 1000 iterations of update. Since the maximum ε value found in literature is $\varepsilon \approx 10$, the performance of the algorithm when ε is very large does not have much value, so we stop our algorithm after ε reaches 100.

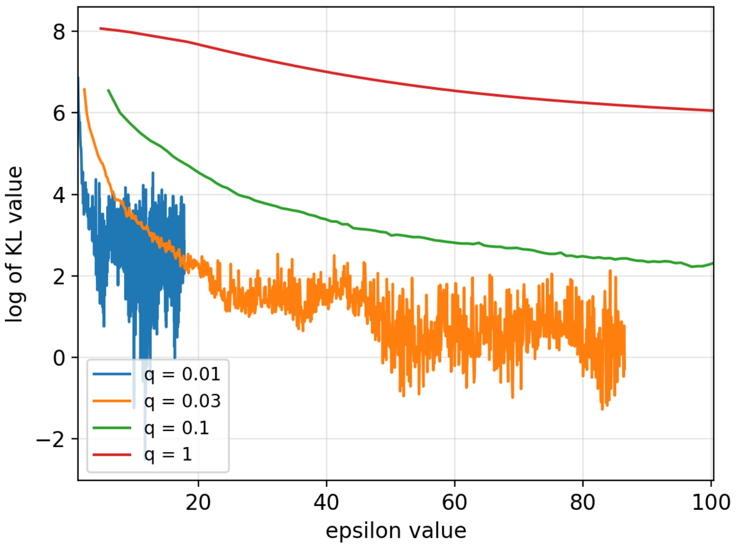
Figure 4.2 shows the results of mini-batching for three different clipping bounds. All three sub-figures exhibit similar behaviors. First of all, the privacy parameter ε has a very large dependence on the value of sampling probability q. After running effectively $50 \times 100 = 5000$ iterations of DP-SGD, the ε value is less than 90 when $q = 0.03$, and is even less than 20 when $q = 0.01$. On the other hand, when $q = 0.1$, the ε reaches 100 after about 110 iterations of server update. When $q = 1$, the situation is even worse: ε reaches 100 after only 2 iterations of server update if we perform 50 iterations of DP-SGD per server update. Thus, the curves corresponding to $q = 1$ are generated by running only 1 iteration of



(a) Log of KL versus ε value for different q values when $C = 2$.



(b) Log of KL versus ε value for different q values when $C = 5$.



(c) Log of KL versus ε value for different q values when $C = 8$.

Fig. 4.2 Results of mini-batching for three different clipping bounds C.

DP-SGD per server update, and the algorithm usually stops after less than 100 iterations of server update.

Moreover, when sampling probability q is smaller, the algorithm achieves smaller KL divergence values, but it is less stable and the parameters of $q(\boldsymbol{\theta})$ will oscillate around. Specifically, when $q = 0.01$ and $q = 0.03$, the algorithm achieves better KL values but is very unstable. On the other hand, when $q = 0.1$ and $q = 1$, the algorithm is stable but has worse performance. This kind of behavior is because larger lot size L (i.e. larger q) will reduce the relative effect of the added DP noise.

We propose two ways of mitigating the instability when q is small. One is to set the learning rate to be adaptive, starting at a large value and reducing over time. The other is to apply a gradient thresholding step which limits the magnitude of each gradient to a set value.

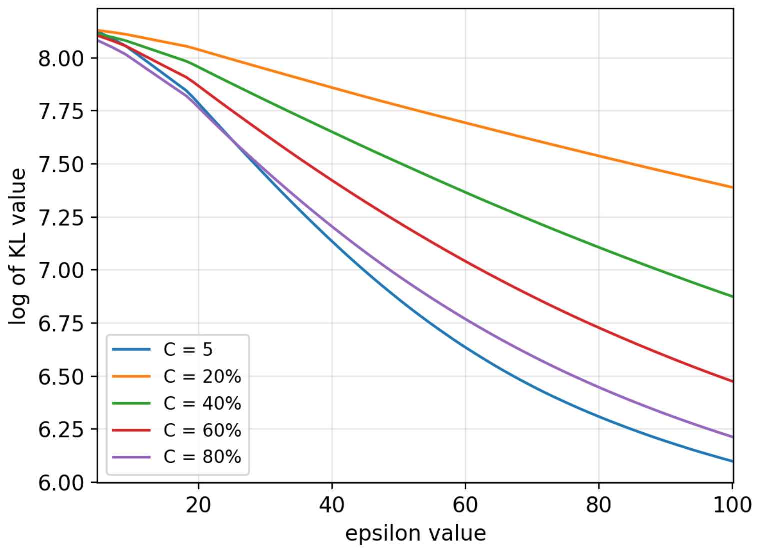
In summary, using a very large sampling probability q is not good because its privacy cost is too high; using a very small q is also not good because it is extremely unstable. If any of the two methods proposed above is used, then we can choose a relatively small q to benefit from lower privacy cost and also ensure stability. Abadi et al. (2016) suggested that lot size $L \approx \sqrt{N}$ is probably the best.

4.3 Adaptive Clipping Bounds

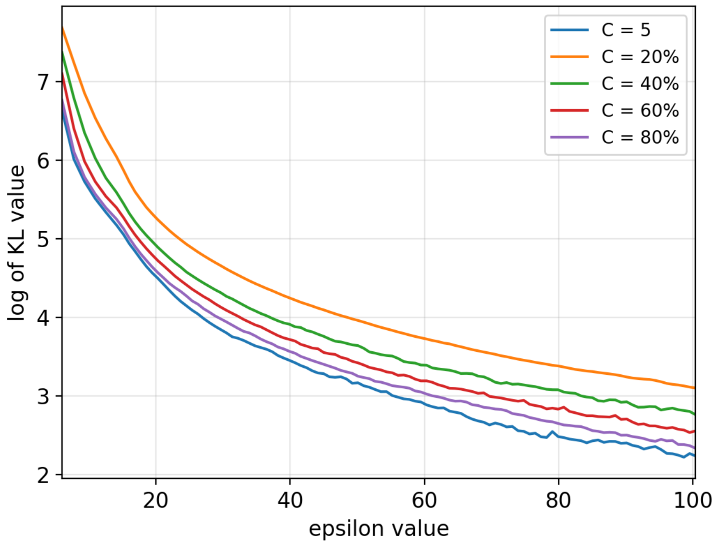
In this subsection, we investigate the idea of adaptive clipping bounds and how it affects the performance of the data-point level DP-PVI algorithm. To begin with, we explain what a good value of clipping bound C should be, and why we want to use adaptive clipping bounds.

If the clipping bound is chosen too large, too much noise will be added to the gradient as the noise scales with σC, and thus the *signal-to-noise ratio (SNR)* of the DP mechanism is needlessly small. On the other hand, if the clipping bound is chosen too small, the clipped gradient may be very different from the true gradient, and thus more iterations are required for the algorithm to converge, resulting in worse privacy guarantees. Therefore, a good value of clipping bound should clip some but not all gradients. Abadi et al. (2016) suggested that C can be chosen as the median of the norms of the unclipped gradients over the course of training, which seems to be a good value to start with.

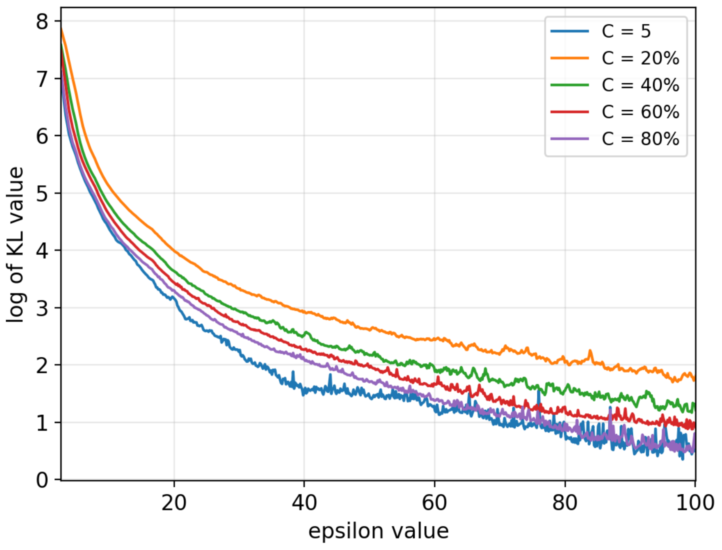
There are two main reasons that we want to use adaptive clipping bounds. Firstly, gradient magnitudes change across the duration of training. When the algorithm approaches convergence, gradient magnitudes are expected to be relatively small compared to those at the start. Secondly, unless we already have significant knowledge about the data set, it is difficult to set an appropriate clipping bound beforehand. In fact, using adaptive clipping bounds will save us a lot of time in hand tuning this hyperparameter.



(a) Log of KL versus ε value for adaptive C when $q = 1, \alpha = 0.03$.



(b) Log of KL versus ε value for adaptive C when $q = 0.1, \alpha = 0.03$.



(c) Log of KL versus ε value for adaptive C when $q = 0.03, \alpha = 0.01$.

Fig. 4.3 Results of adaptive clipping bounds for three different settings.

Here we explore a possible idea of setting adaptive clipping bounds. For each iteration of DP-SGD, we compute the ℓ_2 norms of the unclipped gradients of all data points in this mini-batch, and set the clipping bound C to be at some target percentile.

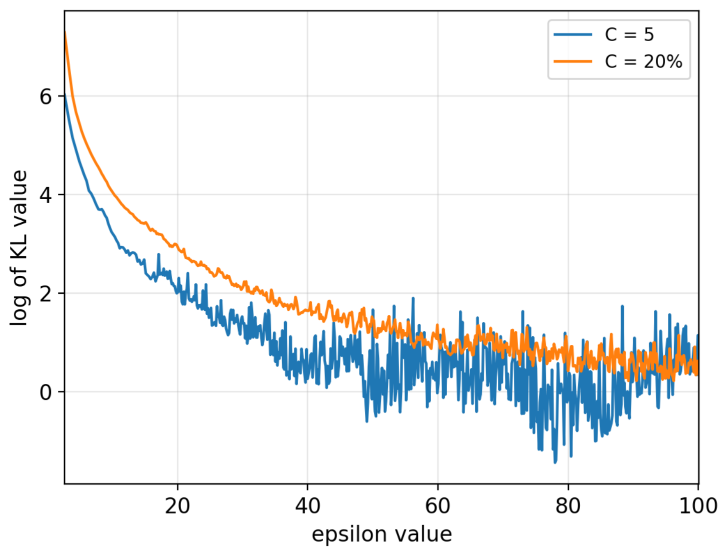
We first use a data set to find out which percentile works the best for different sampling probability q and learning rate α. For our experiments, we fix $\boldsymbol{\theta} = [-1,2]^\top, \sigma_e = 1, \boldsymbol{\mu}_p = \mathbf{0}, \sigma_p = 1$, DP noise scale $\sigma = 1$, and target $\delta = 10^{-4}$. Since adaptive clipping of DP-SGD happens locally in each client, without loss of generality, we use one client with 1000 data points (i.e. $N = 1000$) here. The server can perform a maximum of 1000 iterations of update, and the client performs 100/50/1 iteration(s) of DP-SGD per one server update when $q = 0.03/0.1/1$. As what we did before, we stop the algorithm after ε reaches 100.

Figure 4.3 illustrates the results of adaptive clipping bounds for three different settings. The adaptive clipping bound C is chosen to be 20th, 40th, 60th, and 80th percentile, and is compared to a fixed value $C = 5$. All three sub-figures exhibit similar behaviors. First of all, the privacy guarantees are not affected by the clipping bounds, because in each setting, the same number of iterations of server update is performed for all five experiments when ε reaches 100. Moreover, the performance of adaptive clipping bounds seems to be not as good as that of the fixed clipping bound. In addition, we can observe the pattern that the performance becomes better when the percentile is larger.

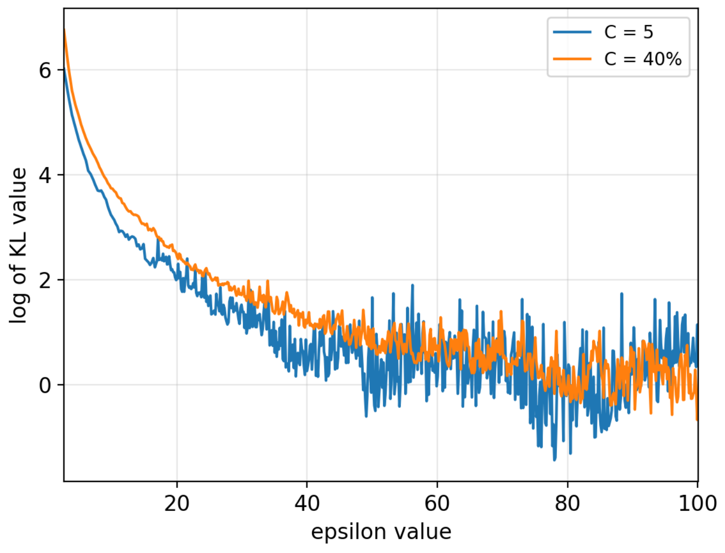
Figure 4.4 shows the case where adaptive clipping bounds are truly useful. In this setting, sampling probability $q = 0.03$ and learning rate $\alpha = 0.03$, so the algorithm will be unstable and the parameters of the approximate posterior will oscillate around when using a fixed clipping bound $C = 5$. Here, adaptive clipping bounds mitigate the instability of the algorithm. Specifically, larger percentile achieves a little bit smaller values of KL divergence, but smaller percentile better reduces the instability. Note that the privacy guarantees are still not affected by the clipping bounds.

Next, we test adaptive clipping bounds on another data set to see whether the percentiles we have used generalize across data sets. We modify the true parameters of the model, setting $\boldsymbol{\theta} = [5,-2]^\top, \sigma_e = 3$. Then we test adaptive clipping bounds on the newly generated data set using 20th, 40th, and 60th percentile versus the same fixed value $C = 5$.

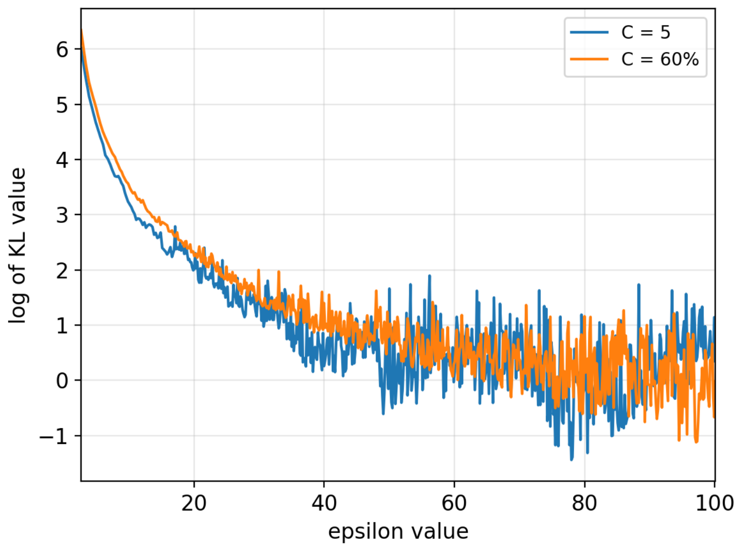
Figure 4.5 shows the results of adaptive clipping bounds on this new data set. We can see that the patterns we observe generalize across different data sets. Adaptive clipping bounds make the algorithm more stable, and smaller percentile better mitigates the instability while has slightly worse performance.



(a) Log of KL versus ε value for C equal to 20th percentile.



(b) Log of KL versus ε value for C equal to 40th percentile.



(c) Log of KL versus ε value for C equal to 60th percentile.

Fig. 4.4 Results of adaptive clipping bounds when $q = 0.03, \alpha = 0.03$.

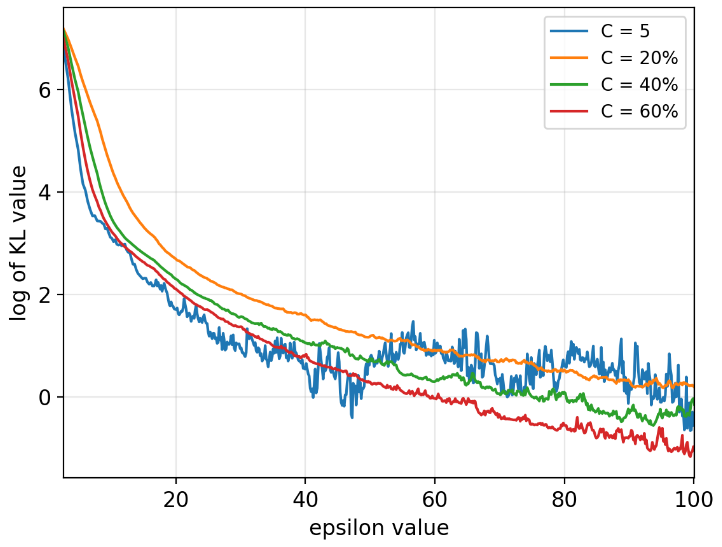


Fig. 4.5 Results of adaptive clipping bounds for a new data set.

We have applied data-point level DP-PVI algorithm to Bayesian multi-dimensional linear regression models, and have explained all experiments and results. In the next chapter, we provide our conclusions.

Chapter 5

Conclusions

In this chapter, we provide our conclusions and elaborate on possible future work.

The Partitioned Variational Inference algorithm is extended to support private federated learning using differential privacy techniques, resulting in two types of DP-PVI algorithms, namely data-point level DP-PVI and data-set level DP-PVI. Data-point level DP-PVI makes fewer assumptions about the credibility of external parties (e.g. the central server, other clients) and enables each client to tune its individual privacy settings. On the other hand, data-set level DP-PVI is the natural scheme to apply when each local data shard corresponds to the data of a single user, but this algorithm has technical problems that need to be fixed in future work.

Data-point level DP-PVI algorithm is applied to Bayesian multi-dimensional linear regression models to assess its performance. In general, the algorithm converges and the KL divergence performance metric shows it works well, but the privacy guarantee achieved is not very ideal. In addition, experiments on mini-batching show that the sampling probability q should be chosen wisely. The privacy cost is too high for large values of q, while the algorithm is unstable for small values of q. Lastly, using adaptive clipping bounds makes the algorithm more stable, and the metric of setting those clipping bounds generalize across different data sets.

5.1 Future Work

The first piece of work that can be done in the future is to solve the technical problems in algorithm 3. Letting the central server add noise seems to be a promising approach, but we will need to find a way that completely remove its pathological random walk behavior.

Performance of the DP-PVI algorithms largely depends on the hyperparameter settings, but it is not clear how to choose the optimal hyperparameter setting. In fact, it takes a lot of time to hand tune all those hyperparameters. In addition, the optimal hyperparameter setting for a specific data set often does not generalize to other data sets. In this project, we have explored the idea of adaptive clipping bounds. For other important hyperparameters like learning rate and DP noise scale, we can also investigate how to make them adaptive in the future. If more hyperparameters are adaptive, the problem of tuning hyperparameters will be alleviated.

Now we have only applied the DP-PVI algorithms to linear models, so another possible direction of future work is to test the algorithms on non-linear models such as Bayesian neural networks. In addition, we have only conducted experiments using homogeneous data. Intuitively, the performance of the DP-PVI algorithms will be worse on inhomogeneous data, so it is worth investigating whether the DP-PVI algorithms perform well on inhomogeneous data.

Last but not least, we can move from our current differential privacy techniques to Rényi DP (Mironov, 2017) in the future.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bui, T. D., Nguyen, C. V., Swaroop, S., and Turner, R. E. (2018). Partitioned variational inference: A unified framework encompassing federated and continual learning. *arXiv preprint arXiv:1811.11206*.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Dwork, C. and Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM.

Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., and Ristenpart, T. (2014). Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32.

Gymrek, M., McGuire, A. L., Golan, D., Halperin, E., and Erlich, Y. (2013). Identifying personal genomes by surname inference. *Science*, 339(6117):321–324.

Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C., and Roth, A. (2014). Differential privacy: An economic method for choosing epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410. IEEE.

Jälkö, J., Dikmen, O., and Honkela, A. (2016). Differentially private variational inference for non-conjugate models. *arXiv preprint arXiv:1610.08749*.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*.

Mironov, I. (2017). Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE.

Narayanan, A. and Shmatikov, V. (2008). Robust de-anonymization of large datasets (how to break anonymity of the netflix prize dataset). *University of Texas at Austin*.

Sharma, M. (2019). Differential privacy & approximate bayesian inference. Master's thesis, University of Cambridge.