Master's thesis
Master's Programme in Data Science

# Sub-sampled and Differentially Private Hamiltonian Monte Carlo

Lauri Lode

December 9, 2019

Supervisor(s): Doctor Antti Koskela

Examiner(s): Associate Professor Antti Honkela

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Koulutusohjelma — Utbildningsprogram — Degree programme |
|---|---|
| Faculty of Science | Master's Programme in Data Science |

Tekijä — Författare — Author

Lauri Lode

Työn nimi — Arbetets titel — Title

Sub-sampled and Differentially Private Hamiltonian Monte Carlo

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidantal — Number of pages |
|---|---|---|
| Master's thesis | December 9, 2019 | 58 |

Tiivistelmä — Referat — Abstract

Hamiltonian Monte Carlo is a powerful Markov Chain algorithm, which is able to traverse complex posterior distributions accurately. One of the method's disadvantages is it's reliance on gradient evaluations over the full data, which quickly becomes computationally costly when the data sets grow large. By mini-batching the data set for stochastic gradient approximations we can speed up the algorithm, albeit with a reduced posterior accuracy. We illustrate by using a toy example, that the stochastic version of the method is unable to explore the exact posterior, and we show how an added friction term greatly alleviates this, when the term is adjusted carefully.

We use the added stochastic error to our advantage, by turning the results differentially private. The randomness in the results masks the appearance of any single data point in the used data set, creating a way to more secure handling of sensitive data. In the case of stochastic gradient Hamiltonian Monte Carlo, we are able to achieve reasonable privacy bounds with little to no decrease in optimization performance, although finding a good the differentially private approximation of the target posterior becomes harder. In addition, we compare the previously considered privacy accounting methods to assay the privacy bounds to a new privacy loss distribution method, which is able to determine a tighter privacy profile than, for example, the moments accountant method.

ACM Computing Classification System (CCS):
Mathematics of computing → Metropolis-Hastings algorithm
Security and privacy → Privacy-preserving protocols
Security and privacy → Social aspects of security and privacy

Avainsanat — Nyckelord — Keywords

Differential privacy, Hamiltonian dynamics, Markov chain

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

# Contents

# 1. Introduction

Hamiltonian Monte Carlo (HMC) techniques [13] are one of the advanced sampling methods in use today. Originally discovered to model the dynamic molecular interactions in chemistry and physics, the method's usefulness in extracting information from probability distributions, such as posterior distributions of Bayesian models, was discovered shortly after. The energy conserving methods allow for more efficient exploration of the posterior distribution than standard random walk operations, and thus HMC needs less iterations than the other Markov chain implementations to explore the target set. However, for every iteration the gradients have to be evaluated over the whole data set, resulting in inefficient and slow iterations when the data set is large. To decrease and possibly distribute the computational burden, the gradients can be stochastically approximated from smaller batches of the original data [12].

Michael Betancourt has argued [7] that the gradients computed from naively subsampled batches cannot accurately simulate Hamiltonian dynamics and thus represent the posterior distribution. In this thesis we evaluate Betancourt's observation and demonstrate its impact on HMC sampling in different situations.

As the popularity of machine learning (ML) rises both in academia and in the industry, more and more sensitive data is being processed by ML algorithms. Data breaches are bound to happen and proper precautions should be taken to alleviate the damage. There are multiple ways to extract unprotected data from these algorithms and they should be secured accordingly.

Differential privacy (DP) aims to perturb the data or the algorithm outcome in a way this becomes difficult. The most common way is to perturb the data or algorithm outcome just so that the accuracy of the model is not compromised, but individual data points cannot be identified or deduced from the results. For example, we can calculate one person's weight from the average weight of a group of people, if the average weight of rest of the group is known exactly. The accuracy at which the person's weight can be computed decreases, when the means of these groups are perturbed according to DP – either by perturbing the weight data or the mean computations.

The second part of this thesis introduces the theory of differential privacy and different methods for computing privacy bounds of certain DP algorithms used to

implement the HMC sampling. Furthermore we compare these algorithms' privacy preserving capabilities and show how to compute the privacy guarantees tightly with newly discovered privacy accounting methods.

We briefly walk through the Markov chain sampling and other preliminaries, and then move on to Hamiltonian dynamics and show how it is used to approximate probability distributions in Chapter 2. In Chapter 3 we combine the stochastic gradient approximations with the Hamiltonian dynamics for increased computational performance. The concept of differential privacy and different ways to obtain and calculate privacy profiles for algorithms are introduced in Chapters 4 and 5. Finally, in Chapter 6 we implement DP to Hamiltonian methods, following by test results for both standard and private algorithms.

# 2. Hamiltonian Monte Carlo

This chapter introduces the Hamiltonian Monte Carlo algorithm and the potentially more efficient stochastic gradient Hamiltonian Monte Carlo (SGHMC) by Chen et al. [12]. We demonstrate the algorithms' strengths and weaknesses and walk through Betancourt's arguments as to why a mini-batched HMC can not model the posterior distribution accurately.

Hamiltonian Monte Carlo is a powerful tool to traverse a target set, e.g. a probability distribution without random-walk like behaviour, which can be also seen as waste of computing power. It is based on physical model consisting of potential and kinetic energy, which models the target as a geometrical representation of the potential energy. One can imagine playing mini-golf on an uneven field. Every time the ball is hit, it launches into desired direction at different speeds. The ball is prone to sliding along the curves of the terrain's topography, unless it is hit hard enough to overcome them. Regarding Hamiltonian dynamics, the terrain is the target set, and the typical use-case is to find the lowest point of the terrain.

## 2.1 Markov Chain Monte Carlo

Markov chain is a way to traverse a typical set stochastically. Commonly it is used to approximate the posterior distribution of a Bayesian model $\pi(q|X) \propto \pi(q)\Pi_{n=1}^{N}\pi(x_n|q)$, or

$$\pi(q|X) = \frac{\pi(q) \cdot \Pi_{n=1}^{N}\pi(x_n|q)}{\int \Pi_{n=1}^{N}\pi(x_n|q) \cdot \pi(q)d\pi}.$$

In complex models and for large data sets the integral quickly becomes nigh-impossible or too costly to evaluate. Therefore an accurate approximation of the posterior is needed to efficiently obtain posterior distributions for large models. Here $\pi(q)$ is the prior distribution of the parameter vector $q$ in a model space $\Omega$ and $\pi(x_n|q)$ is the likelihood for the $n$th data point $x$ in the set $X$ of $N$ items.

The typical Markov chain implementation is the Metropolis-Hastings algorithm, which has two phases: the proposal phase and the correction phase. For a vector of states $q = (q_1, \ldots, q_t)$, a new sample $q_{t+1}$ is drawn from a proposal probability density

$\mathcal{Q}(q'|q)$ and the new state vector that includes the new sample is marked as $q'$. For the random walk Metropolis, the Gaussian distribution is typically used as the proposal distribution with covariance matrix $\Sigma$:

$$\mathcal{Q}(q'|q) = \mathcal{N}(q'|q, \Sigma).$$

To ensure that the new sample doesn't stray too far from the typical set of the target distribution $\pi(q|X)$, each new proposal is accepted as a part of the chain with the probability

$$\alpha(q'|q) = \min\left(1, \frac{\mathcal{Q}(q|q')\pi(q')}{\mathcal{Q}(q'|q)\pi(q)}\right).$$

The Gaussian proposal distribution has a natural bias towards larger distribution volumes and the proposals from low-density areas of the model space are rejected. With enough iterations the chain will converge exactly to the target distribution $\pi(q|X)$ [10], however this rarely is feasible due to finite resources and slowness of the traversal. Especially in higher dimensions a close-enough approximation from appropriate sample size is used [8].

The position of the target set is typically unknown, and the chain is started at a random point. From there it usually takes the algorithm multiple iterations to converge to the target distribution and to start accepting samples drawn independent of the previous samples [8, 10]. These preliminary samples are typically discarded as part of a "burn-in" process to rid the sample collection of correlation. When the target is located, the initial, crude approximation of the target set is acquired quickly. Additional drawn samples increase the posterior accuracy, but at the same time the speed at which the accuracy improves decreases, until any meaningful progress demands multiple new samples [10].

### 2.1.1 Volumes in Higher Dimensions

In higher dimensions, the volume and density inside the target set will largely diminish compared to outside volume of the model space [8] (Fig. (2.1). In other words, the ratio between empty space and the area of interest grows larger. Due to random walk MCMC's bias to higher volumes, the vast majority of new proposals will be drawn from the outside of the target and rejected. The accepted samples are drawn close to starting point, leading to very slow exploration of the target distribution. We will next show that the Hamiltonian Monte Carlo method has capability of following the target set more efficiently, even in higher dimensions, and taking larger leaps towards the unexplored areas of the distribution [8].

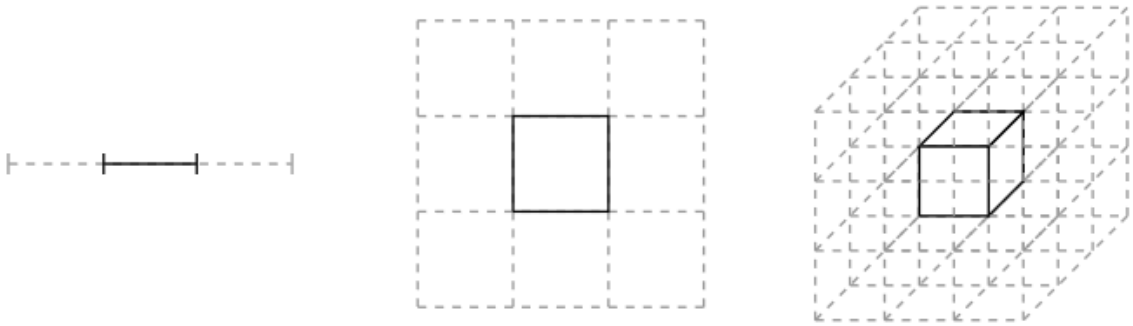

**Figure 2.1:** Increasing the target space dimensionality does not preserve the area/volume ratio of the target set and its complement. MCMC sampler is predisposed towards high volumes and as such biases outwards from the target set in higher dimensions. Figure by Michael Betancourt [8].

## 2.2 Hamiltonian Dynamics

Initially Hamiltonian dynamics were conjured to simulate molecular interactions and movement trajectories in physics. Later it was discovered that the trajectories of the molecules could be deterministically simulated by following Newton's law of momentum [3]. The same principles allow an efficient sampling of almost any probability distribution, and the modern Hamiltonian Monte Carlo Markov chain method follows this approach [8].

To mimic the conservation of energies in physical systems, Hamiltonian dynamics introduces a new, independent auxiliary variable $p$, the momentum, alongside the proposal sample $q$, the state or the position. We expand the $d$-dimensional parameter space to $2d$-dimensional phase space [8]:

$$q \to (q, p). \tag{2.1}$$

Continuing with the analogy, the energy (i.e. Hamiltonian) of the Hamiltonian system is marked $H(q, p)$ [13] and can usually be presented by the sum of potential energy $U(q)$ and kinetic energy $K(p, q)$ as follows:

$$H(q, p) = K(p, q) + U(q).$$

For brevity's sake we utilize the commonly used Euclidean metric to determine our kinetic energy, which resembles the "real-world" counterpart

$$E_k = \frac{mv^2}{2}.$$

When the kinetic energy is independent of the position and thus the state $q$, the Hamiltonian is called separable [7]. For example, the Euclidean metric operates like

this, and the Hamiltonian can be defined as

$$H(q,p) = K(p) + U(q).$$

In Hamiltonian dynamics, the change of $q$ and $p$ over time, $t$, is determined by $H$'s partial derivatives, which define the Hamilton's differential equations [8, 24]:

$$\begin{aligned} \frac{dq}{dt} &= \frac{\partial H}{\partial p} = \frac{\partial K}{\partial p} \\ \frac{dp}{dt} &= -\frac{\partial H}{\partial q} = -\frac{\partial K}{\partial q} - \frac{\partial U}{\partial q}. \end{aligned} \tag{2.2}$$

Here Both equations are gradients w.r.t $p$ and $q$:

$$\frac{\partial H}{\partial p} = \begin{bmatrix} \frac{\partial H}{\partial p_1} \\ \vdots \\ \frac{\partial H}{\partial p_n} \end{bmatrix} \quad \text{and} \quad \frac{\partial H}{\partial q} = \begin{bmatrix} \frac{\partial H}{\partial q_1} \\ \vdots \\ \frac{\partial H}{\partial q_n} \end{bmatrix}.$$

Together with the initial values $(q_0, p_0)$, the equations (2.2) give the Hamiltonian system. The partial derivatives form the Hamiltonian vector field. For separable Hamiltonians i.e. ones based on Euclidean metric, the vector field can also be decoupled to kinetic and potential vector fields:

$$\vec{H} = \frac{\partial H}{\partial p}\frac{\partial}{\partial q} - \frac{\partial H}{\partial q}\frac{\partial}{\partial p} \equiv \vec{K} + \vec{U}.$$

Next we define the mapping $\phi_t^H$, Hamiltonian flow, which maps the initial values of $q$ and $p$ to the solution at the moment of time $t$:

$$\phi_t^H(q,p) = (q(t), p(t)). \tag{2.3}$$

The joint canonical density distribution of the two variables $q$ and $p$ is [8]

$$\pi(q,p) = \pi(p|q) \cdot \pi(q). \tag{2.4}$$

As we are only interested in the state $q$, by marginalizing the momentum in the joint density distribution, we acquire the target distribution. After integrating over the Hamiltonian for some time $t$ i.e., applying the Hamiltonian flow onto the phase space (2.1)

$$(q,p) \to \phi_t^H(q,p),$$

we can project the resulting phase space back to the target space [7] to acquire a new sample for the Markov chain:

$$(q(t), p(t)) \to q(t).$$

The density of the canonical distribution does not depend on any choice of parametrization, and thus can be written in terms of invariant Hamiltonian function [8]

$$\pi(q,p) = e^{-H(q,p)}.$$

### 2.2.1 Base Properties of Hamiltonian Systems

One of the base properties of Markov chain is the reversibility of the chain. To keep the target distribution invariant, there should be a way to compute the previous state from any state in the chain. Hamiltonian dynamics complies with this requirement: when $K(p,q) = K(-p,q)$, we can negate the time derivatives in the Hamilton's equations and can so obtain the state $(q(t), p(t))$ from the state $(q(t+1), p(t+1))$ [24], and backtrack our way through the sampling execution.

#### The Invariancy of the Hamiltonian System

The joint density $\pi(q,p)$ does not depend on the parametrization i.e., the Hamiltonian dynamics is kept invariant for $d$-dimensional parameter vectors [8, 24]:

$$\begin{aligned}\frac{dH}{dt} &= \sum_{i=1}^{d}\left[\frac{dq_i}{dt}\frac{\partial H}{\partial q_i} + \frac{dp_i}{dt}\frac{\partial H}{\partial p_i}\right] \\ &= \sum_{i=1}^{d}\left[\frac{\partial H}{\partial p_i}\frac{\partial H}{\partial q_i} - \frac{\partial H}{\partial q_i}\frac{\partial H}{\partial p_i}\right] \\ &= 0.\end{aligned}$$

Thus we can set the Hamiltonian to equal the joint density function (2.4)[8]:

$$H(q,p) = -\log \pi(q,p)$$

i.e.,

$$\pi(q,p) = e^{-H(q,p)}.$$

From this follows that we can decompose joint density function as

$$\begin{aligned}H(q,p) &= -\log \pi(q,p) \\ &= -\log \pi(p|q) - \log \pi(q) \\ &= K(p,q) + U(q)\end{aligned}$$

Alternatively, in the Euclidean metric the kinetic energy can be defined in quadratic form as

$$K(p) = \frac{p^T M^{-1} p}{2}$$

for convenience's sake [24]. Here $M$ is a symmetric, positive-definite matrix resembling the mass of the object in the physical analogue. Typically $M$ is a diagonal identity matrix, and can be used to rotate and rescale the target distribution if necessary [24]. Although, in practice $M$ is seldom used beyond that, and thus we omit it from the equations for the rest of the thesis, for brevity's sake.

### Symplecticity

The Hamiltonian system $H$ is symplectic i.e., for the Jacobian matrix $A$ of the exact flow of the mapping $\phi_t^H$

$$A = \frac{\partial \phi_t^H (q,p)}{\partial (q,p)}$$

holds true

$$A^T J A = J,$$

where

$$J = \begin{bmatrix} 0_d & I_d \\ -I_d & 0_d \end{bmatrix}.$$

Essentially, the properties possessed by the phase space $(q,p)$ will retain these properties even after being transformed by the Hamiltonian $\phi_t^H$. Especially the volume of $(p,q)$ is preserved throughout the computation, as

$$\det(A^T)\det(J)\det(A) = \det(J)$$

which implies that

$$\det(A)^2 = 1, \tag{2.5}$$

since $\det(J) = 1$ and $\det(A) = \det(A^T)$. From (2.5) follows that $\det(A)$ is either 1 or $-1$, due the mapping $\phi_t^H$ being continuous function w.r.t the time variable $t$, and initially at $t = 0$ the mapping is an identity matrix:

$$\phi_0^H = I_d.$$

Thus, the determinant can not jump from positive to negative [17] and the mapping $\phi_t^H$ is a symplectic mapping.

### The Choice of Kinetic Energy and the Energy Conservation

Euclidean metric is the classical choice for the simulation of energies, though a model employing Riemannian manifolds tends to converge faster and more accurately at the expense of greatly increased computational complexity per iteration [16, 32]. Given the need one could craft their own kinetic energy metric to the exact need of their model, but it would be resource-intensive to find a properly working equation, and the either of the previous Gaussian-based metrics typically give good performance in almost any model [8, 24].

The canonical distribution also ensures that likewise in the physical dynamics model, the conservation of energies are kept intact: the volume of energy consisting of the $2d$-dimensional phase space $(q,p)$ stays constant throughout the process [8, 24].

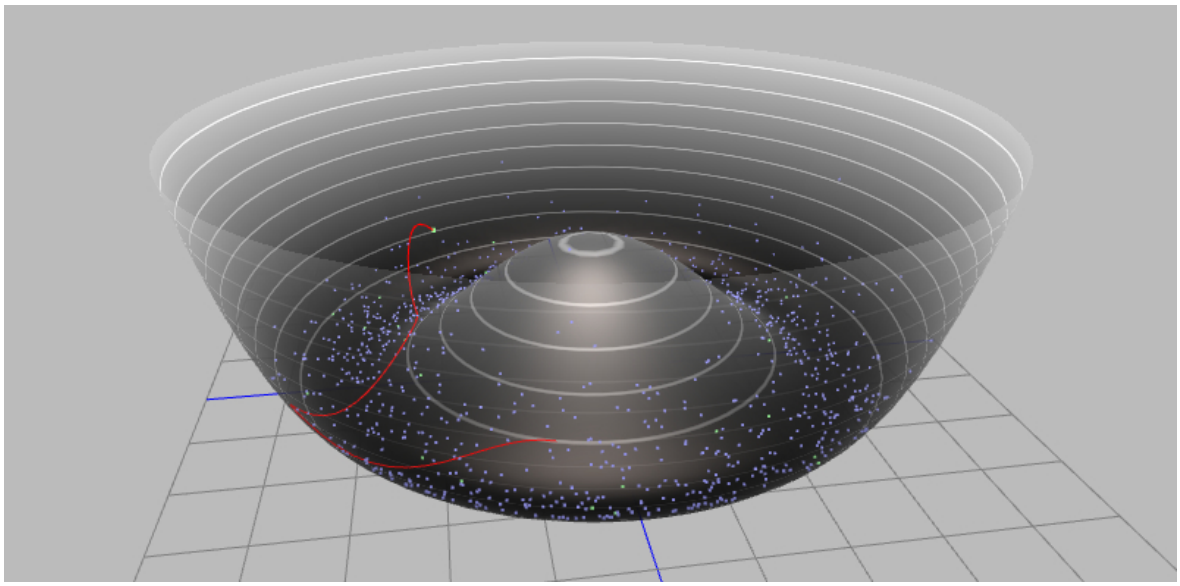

**Figure 2.2:** HMC models the target set as a construct of potential energy. The red line resembles integrator trajectory traversing the set and the dots are drawn samples from the set. Figure by Alex Rogozhnikov [27].

When the $(q, p)$ space stretches along either of the variables, e.g. the potential energy increases, the space will equally shrink along the other, as the kinetic energy decreases [8]:

$$dq(t)dp(t) = dq(0)dp(0).$$

A vector field with zero divergence is known to preserve volume [5], and this property can be shown with the vector field defined by the Hamilton's equations [24]:

$$\sum_{i=1}^{d}\left[\frac{\partial}{\partial q_i}\frac{dq_i}{dt} + \frac{\partial}{\partial p_i}\frac{dp_i}{dt}\right] = \sum_{i=1}^{d}\left[\frac{\partial}{\partial q_i}\frac{\partial H}{dp_i} - \frac{\partial}{\partial p_i}\frac{\partial H}{dq_i}\right] = \sum_{i=1}^{d}\left[\frac{\partial^2 H}{\partial q_i \partial p_i} - \frac{\partial^2 H}{\partial p_i \partial q_i}\right] = 0.$$

The conservation of energy restricts every Hamiltonian trajectory to an energy level set (notation due [8])

$$H^{-1}(E) = \{q, p | H(q, p) = E\},$$

that are compact $2d-1$-dimensional surfaces in the phase space. Between the trajectory simulations the momentum $p$ is sampled again and the algorithm transcends from one energy set to another, as presented in Figure (2.3).

## 2.3 HMC in Practice

As per the previous sections, the HMC is compatible with Markov chain requirements. However, as numeric procedures are prone to introduce stacking error to the calcula-

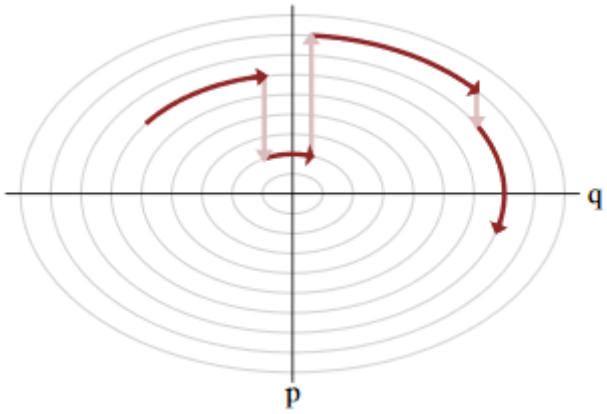


**Figure 2.3:** Four integrator steps simulated consecutively. With the re-sampling of the momentum $p$, the phase space switches from a energy level to another on the energy level set. After a set length the phase space $(q, p)$ is projected back to target space. Figure by Michael Betancourt [8].

tions, the trajectories in the Hamiltonian vector field need a special numerical integrator. Even these sophisticated methods introduce a small amount of error, and we should adjust for that.

### 2.3.1 Symplectic Integrators and the Leapfrog Method

In practice, the exact Hamiltonian flow trajectories are usually not feasible to calculate due to non-linearity of the involved differential equations. Also the error induced by the commonly used numerical integrators will compound and bias these trajectories, so that they will drift away from the actual trajectories in the Hamiltonian vector field. A group of integrators, called symplectic integrators, are tailored for Hamiltonian systems to preserve the symplecticness and so the accuracy of the Hamiltonian transition [8, 24].

The numerical integrator is defined by a one-step mapping

$$y_{i+1} = \phi_h(y_i).$$

The integrator is symplectic, if the mapping $\phi_h(y)$ is symplectic, meaning that its Jacobian is a symplectic matrix:

$$\left(\frac{\partial \phi_h(y)}{\partial y}\right)^T J \left(\frac{\partial \phi_h(y)}{\partial y}\right) = J.$$

For the Hamiltonian system, established in Section (2.2.1) it stands that the mapping $\phi_t^H$ is indeed a symplectic mapping.

Due to numerical constraints the integration time has to be discretized into small intervals of length $\eta$, commonly known as the step size. The trajectory is constructed by applying the integrator $L$ times onto the system $H$.

One typically used second-order symplectic integrator is the leapfrog integrator, where the Hamiltonian flow is split into two parts in terms of potential and kinetic energy. The first a half-step in time is taken along the potential flow and the momentum $p$ updated. The state $q$ is updated along the kinetic flow, with the new $p$. Finally the second half-step in time is taken and the momentum $p$ updated again, with the new $q$. In practice, $p$ and $q$ are updated as below:

$$\begin{aligned} p(t+\eta/2) &= p(t) - (\eta/2)\frac{\partial U}{\partial q}q(t) \\ q(t+\eta) &= q(t) + p(t+\eta/2)\eta \\ p(t+\eta) &= p(t+\eta/2) - (\eta/2)\frac{\partial U}{\partial q}q(t+\eta) \end{aligned} \tag{2.6}$$

The resulting $p$ and $q$ would ideally yield the exact solutions for the mapping $\phi_t^H$, however a small numerical error is introduced in the time stepping: while the integrator itself is exact, approximating the continuous time with discrete steps cause deviation from the actual trajectory. The leapfrog integrator approximately preserves the energy of the system $H(q,p)$ [8, 24]. This implies small numerical errors in long time integrations [17]. Being symmetric in nature, it also preserves the reversibility requirement of the Markov chain: by negating $p$ we are able to backtrack the leapfrog steps to the earlier position [24].

The HMC algorithm is tuned by the step-size of $\eta$ and the amount of the integrator steps $L$. A good starting point for the amount of leapfrog iterations tends to be $L = \lfloor T/\eta \rfloor$, where $T$ is the total amount of algorithm iterations $t$. Smaller step size leads to more thorough, but slower exploration of the target, and a larger step size to faster traversal around it. The same stands for the trajectory length $L$ [8]. The longer trajectories have a high probability of straying farther away from the starting point than short trajectories, which drastically lowers the autocorrelation between the samples [7].

### 2.3.2 Accuracy of Symplectic Integrators

For separable Hamiltonians

$$H(q,p) = K(p) + U(q),$$

the subsystems, potential and kinetic energies, are then the exact solutions of the system

$$\begin{aligned} (q,p) &= \left(\frac{\partial K}{\partial p}, 0\right) \\ (q,p) &= \left(-\frac{\partial H}{\partial q}, 0\right) \end{aligned}$$

and the method can be presented as a symmetric composition in terms of Strang splitting [7] i.e., the leapfrog method:

$$\begin{aligned}\phi_\eta^H &= \phi_{\eta/2}^U \circ \phi_\eta^K \circ \phi_{\eta/2}^U \\ &= e^{\frac{\eta}{2}\vec{U}} \circ e^{\eta\vec{K}} \circ e^{\frac{\eta}{2}\vec{U}},\end{aligned}$$

where $\phi_{\eta/2}^U$ and $\phi_\eta^K$ give the exact flows corresponding to these subsystems, respectively. By applying the Baker-Campbell-Hausdorff formula [6] on the equation, which will give the solution for the equation

$$e^X e^Y = e^Z,$$

for lie groups [28] $X$, $Y$ and $Z$. This composition yields [7]

$$\begin{aligned}&e^{\frac{\eta}{2}\vec{U}} \circ e^{\eta\vec{K}} \circ e^{\frac{\eta}{2}\vec{U}} \\ &= \exp(\frac{\eta}{2}\vec{U}) \circ \exp\left(\eta\vec{K} + \frac{\eta}{2}\vec{U} + \frac{\eta^2}{4}\left[\vec{K},\vec{U}\right]\right) + \mathcal{O}(\eta^3) \\ &= \exp\left(\frac{\eta}{2}\vec{U} + \eta\vec{K} + \frac{\eta}{2}\vec{U} + \frac{\eta^2}{4}\left[\vec{K},\vec{U}\right] + \frac{1}{2}\left[\frac{\eta}{2}\vec{U}, \eta\vec{K} + \frac{\eta}{2}\vec{U} + \frac{\eta^2}{4}\left[\vec{K},\vec{U}\right]\right]\right) + \mathcal{O}(\eta^3) \\ &= \exp\left(\eta\vec{H} + \frac{\eta^2}{4}\left[\vec{K},\vec{U}\right] + \frac{\eta^2}{4}\left[\vec{U},\vec{K}\right] + \frac{\eta^2}{8}\left[\vec{U},\vec{U}\right]\right) + \mathcal{O}(\eta^3) \\ &= e^{\eta\vec{H}} + \mathcal{O}(\eta^3),\end{aligned}$$

where we denote

$$\left[\vec{K},\vec{U}\right] = \vec{K}\vec{U} - \vec{U}\vec{K}.$$

The compounding error from the exact Hamiltonian flow is bound by $\mathcal{O}(\eta^2)$, when this composition is composed with itself $L = \lfloor T/\eta \rfloor$ times. Here $T$ is the total amount of iterations, or the integration time [7]:

$$\begin{aligned}\phi_{\eta,T}^{\tilde{H}} &\equiv \left(\phi_{\eta/2}^U \circ \phi_\eta^K \circ \phi_{\eta/2}^U\right)^L \\ &= \left(\exp(\eta\vec{H}) + \mathcal{O}(\eta^3)\right)^L \\ &= \exp((L\eta)\vec{H}) + (L\eta)\mathcal{O}(\eta^2) \\ &= \exp(T\vec{H}) + T\mathcal{O}(\eta^2) \\ &= e^{T\vec{H}} + \mathcal{O}(\eta^2).\end{aligned}$$

### 2.3.3 Implementing the HMC Algorithm

We can run HMC as a implementation of a MCMC algorithm. We propose the state $q$ as a part of the Markov chain, while the momentum $p$ is discarded.

In the proposal stage we sample $p$ independently of $q$ from a Gaussian distribution with zero mean, and the variance $M$ from the kinetic energy quadratic form: $p \sim$

$\mathcal{N}(0, M)$. The independence of $p$ implies that the canonical joint distribution $\pi(q, p)$ stays invariant [24].

After simulating the leapfrog trajectories (or any other reversible symplectic integrator that preserves the volume of the phase space $(q, p)$) $L$ times, we obtain the new proposal $q'$ (see Fig. (2.2)). Due to the residual error left from the symplectic integrators, the samples $q'$ would bias the resulting Markov chain and cannot be accepted outright: the MCMC-like acceptance step is still crucial for an accurate approximation [7, 9]. The new state is accepted with a probability:

$$\begin{aligned}\alpha(q,p) &= \min\left(1, \exp\left(H(q,p) - H \circ \phi^{\tilde{H}}_{\eta,T}(q,p)\right)\right) \\ &= \min\left(1, \exp\left(H(q,p) - H(q',p')\right)\right).\end{aligned}$$

Here $(q, p)$ is the initial state and $(q', p')$ the new proposed state. In case the proposal is rejected, a new proposal is solved from the initial state again. However, the probability for an accepted proposal tends to still be high, as the injected noise is quite small. Thus, an optimal acceptance rate is around 65-80% [8, 24], whereas for random walk MCMC the asymptotically optimal ratio should be nearing 23% [26]. For efficiency's sake, typically the acceptance range of 23-50% is deemed acceptable [11, 24, 26], as the complexity of the model will also affect the acceptance ratio – in some cases too low acceptance might leave some areas of the target distribution unexplored [11, 26].

After the leapfrog integrations the momentum $p$ should be reversed to make the HMC Markov chain symmetrical. In practice though, is usually is not relevant, as we can show that $K(p) = K(-p)$, the momentum is rarely needed afterwards, and it is re-sampled in the next iteration cycle before it is utilised again [24]. The long deterministic trajectories eliminate the random-walk behaviour and the samples are less correlated with each other. This means the algorithm finds the target set faster and we can manage with smaller amount of samples discarded by the burn-in process.

The step size $\eta$ does not need to be constant throughout the run [24]. As we later see, there is some benefit for decreasing step size: after converging to the typical set, long trajectories with the length $\eta L$ deviate from the actual vector field $H$ more and cause lower acceptance rates [24]. Slight variations in the step size enhances the convergence properties of the algorithm, and it is good practise to perturb $\eta$ for a small amount at the start of a new trajectory [24]. With variable step size the phase space $(q, p)$ cannot be evaluated at consistent time points any more, but this will not affect the performance of the algorithm or is typically needed for evaluation [24].

### 2.3.4 Strengths and Weak Spots of HMC Method

One of the bigger advantages of the Hamiltonian Monte Carlo is its divergence from random walk behaviour. Long trajectories following the Hamiltonian vector field traverse the target set usually much faster, and with a higher acceptance rate leading to few wasted computing cycles. The burn-in phase can also be reduced, as the distances between consecutive samples are typically larger and the trajectories are aimed at the target set [8][24]. It is shown that HMC is better equipped sampling from a set of correlated distributions, over many other Markov chain implementations [24].

However, the improved performance comes at the cost: the gradients for the trajectories must be calculated at every step of the numerical integration. Typically the performance is still better than with random walk MCMC due to faster convergence rates – only a fraction of iterations are needed for the same accuracy. As the model complexity grows and the gradients get slower to evaluate, likewise the step size for random walk MCMC has to be set low due to its tendency to aim for the higher volumes [8] (see Section (2.1.1)).

In general, MCMC algorithms have problems traversing areas of high curvature, e.g. sharp corners in the target set. The pathology stands true to HMC, but fortunately also for the symplectic integrators. At these points the integrators fail catastrophically and diverge greatly from the actual Hamiltonian trajectories – by following the simulated trajectories the problematic areas can be identified and the algorithm tuned accordingly. Mild pathologies can be remedied by lowering the step size $\eta$, though major problems usually require adjusting or rethinking the prior, or even the target distribution to avoid biases in the resulting estimate [8].

As with all gradient-dependent algorithms, multi-modal distributions cause problems, when the algorithm sets to converge into one of the modes. When the sampler dives into an isolated local minimum, it might not have enough momentum to cross the energy ridge and leave rest of the target set unexplored. Proper tempering of the algorithm will alleviate the problem to the extent [24].

# 3. Stochastic Gradient Hamiltonian Monte Carlo

Hamiltonian Monte Carlo is a powerful sampling tool, however the gradients must be computed in each iteration cycle. When the model complexity increases, or the amount of data increases, the calculations take higher and higher toll on the computer. When implementing HMC next to a data stream, the whole data set cannot be utilized outright. With the advent of Big Data, the possible dataset grow impossibly huge and cannot be evaluated efficiently at once.

In this chapter we show the stochastic gradient Hamiltonian Monte Carlo (SGHMC) algorithm. By dividing the data into smaller batches and applying stochastic gradient descent (SGD) on them, we can lighten the computational load per iteration. The basic idea of mini-batch approximation of gradients is given in Section (3.1). To counteract the error resulting from the stochastic gradients, an additional friction term can be added to the symplectic integrator, so that the trajectories would follow the Hamiltonian flow more closely [12]. The idea of the friction term is introduced in Section (3.1.1).

## 3.1 Implementing the SGHMC Algorithm

For the mini-batch operations we assume that all data $X$ of size $N$ is independently and identically distributed (i.i.d.) and as such adhering to central limit theorem [7, 12]. In Bayesian statistics, the posterior distribution is presented as a product of likelihoods for each data point $x_i \in X$:

$$\pi(q|X) \propto \pi(q) \prod_{i=1}^{N} \pi(x_i|q).$$

Thus the potential energy can also be decomposed into a sum. We can represent the potential energy as

$$\begin{aligned} U(q) &= -\log\left(\pi(X|q)\pi(q)\right) = -\log\pi(X|q) - \log\pi(q) \\ &= -\sum_{n=1}^{N}\log\pi(x_n|q) - \log\pi(q). \end{aligned}$$

The data is divided into batches of size $m$ randomly and uniformly [12], so there will be $N/m$ sub-sets of data. The full potential energy can then be approximated as

$$\begin{aligned} \tilde{U}(q) &= -\frac{N}{m}\sum_{i=1}^{m}\log\pi(x_i|q) - \log\pi(q) \\ &= -\frac{N}{m}U_j(q), \end{aligned}$$

where $U_j(q)$ is the potential energy of one mini-batch. We denote the formed mini-batches with $B_j$, where $j \in J = \{1, \ldots, N/m\}$ is the index of a mini-batch. The stochastic gradient for $\partial U/\partial q$ is calculated as

$$\nabla U(q) = -\sum_{x\in X}\nabla\log\pi(x|q) - \nabla\log\pi(q),$$

and similarly the total gradient approximated from one mini-batch will be [12]

$$\nabla\tilde{U}(q) = -\frac{N}{m}\sum_{x\in B_j}\nabla\log\pi(x|q) - \nabla\log\pi(q).$$

Due to the assumption that the data complies with the central limit theorem, this noisy gradient can be approximated [12] as

$$\nabla\tilde{U}(q) \approx \sum_{x\in B_j}\nabla\log\pi(x|q) - \nabla\log\pi(q) + \mathcal{N}\left(0, \mathrm{cov}(q)\right).$$

Compared to actual gradient, the stochastic gradient injects the error of $\mathcal{N}\left(0, \eta\cdot\mathrm{cov}(q)\right)$ to the results. Here $\mathrm{cov}(q)$ is the covariance of the parameter vector $q$ [12]. Naturally, a large batch size $B$ gives more accurate results than smaller batches.

During the symplectic integration phase, the error enables the entropy to rise inside the Hamiltonian system $H$. The canonical joint distribution $\pi(q, p)$ is no longer invariant, as the volume of the system's energy is supposed to be preserved inside the system. The resulting estimate distribution biases towards uniform distribution, which could be far from the actual target distribution [12]. The longer the simulated trajectories are, the more the trajectories differ from the actuals and entropy is increased.

Because of the bias, the proposals have to be validated, and we again have a need for Metropolis-Hastings acceptance step. The acceptance is computed over the whole

data set, which makes it a computationally heavy operation and thus we prefer not to run it after only short trajectory simulations. On the other hand, longer simulations get more biased, are likely to be rejected and we again lose the potential computational gains as more iterations are needed for convergence [12].

### 3.1.1 Friction Term

To counteract the bias from the stochastic noise, Chen et. al [12] propose adding a friction term to the integration. Properly tuned, it negates the error and also retains the joint distribution $\pi(q, p)$ as an invariant. In a real-world analogue, the friction between a rolling ball and the surface will keep the ball on track and more resistant to winds that would otherwise change its course. The friction decreases the total energy of the Hamiltonian $H(q, p)$: as the entropy is shown to strictly increase [12], the additional energy loss will balance the system's energy by stalling the ratio at which the energy increases.

The friction term implemented into the system could be seen as a partial momentum update, and it originates from second-order Langevin dynamics [12]. It does not improve the standard HMC method with noiseless gradients, however it is crucial in counteracting the injected noise in stochastic gradients. With Hamiltonian system $H'$, augmented with an appropriately selected friction term, the dynamics give the unique stationary distribution

$$\pi(q, p) \propto e^{-H(q,p)}.$$

(Theorem 3.2 [12]).

We introduce the friction term $\mathcal{B}M^{-1}p$ to the trajectory calculations, such that the updates for the momentum $p$ are changed:

$$p(t) = p(t - t') - \frac{\eta}{2}\left(\frac{\partial U}{\partial q}(q(t - t')) - \mathcal{B}p(t - t') + \xi\right), \quad \xi \sim \mathcal{N}(0, 2\mathcal{B}).$$

Here $\mathcal{B} = (\eta/2) \cdot \mathrm{cov}(q)$ is the diffusion matrix contributed by the gradient noise, called the noise model. The states $(t - t')$ refer to the previous state of the parameters $p$ and $q$, whether they were the state of the first or the second half-step in the integrator (2.6) [12].

In practice the exact $\mathcal{B}$ is typically not known, and we have to use an approximate $\tilde{\mathcal{B}}$. We introduce a user-specified model $\mathcal{C} \succeq \mathcal{B}$ to have more control over the noise reduction [12]. Here $A \succeq B$ denotes that $A - B \geq 0$ i.e., $A - B$ is a positive semi-definite matrix. $A \succ B$ indicates similarly that $A - B > 0$ and is a positive definite matrix.

The stochastic noise $\mathcal{B}$ will eventually reach zero, as $\eta \to 0$, and so the substitute $\mathcal{C}$ and the injected noise will dominate over $\mathcal{B}$ and the stochastic error, when we use a

strictly decreasing sequence of step sizes [12, 33]. The integration step (2.6) is updated with the new equation for the momentum updates as follows:

$$\begin{aligned} p(t) &= p(t-t') - \frac{\eta}{2}\left(\frac{\partial U}{\partial q}(q(t-t')) - \mathcal{C}M^{-1}p(t-t') + \xi + \xi'\right) \\ &\equiv p(t-t') - \frac{\eta}{2}\left(\frac{\partial U}{\partial q}(q(t-t')) - \mathcal{C}p(t-t') + \xi\right), \end{aligned} \tag{3.1}$$

where $\xi \sim \mathcal{N}\left(0, 2(\mathcal{C} - \tilde{\mathcal{B}})\right)$ and $\xi' \sim \mathcal{N}(0, 2\mathcal{B})$. As $\mathcal{C} \succeq \mathcal{B}$, likewise $\xi \succeq \xi'$ and $\xi$ dominates over other Gaussian errors. To omit the MH-acceptance step, the SGHMC algorithm uses a strictly decreasing series of step sizes $\eta$, which satisfy the Robbins-Monro conditions [25, 33]

$$\sum_i \eta_i^2 < \infty \text{ and } \sum_i \eta_i = \infty, \qquad \eta_i \in \eta.$$

The same step-size decay is implemented in stochastic gradient Langevin dynamics (SGLD) [12, 33]. These conditions make sure that the parameter vector $q$ eventually reaches the high probability regions of the target set and that the model will converge to the mode [33]. We typically can tolerate a small error, so the step size is bounded by a small non-zero value instead of dropping down to zero. The trade-off is between the efficiency of the algorithm and the accuracy: when $\eta \to 0$, the algorithm is sampling the exact posterior and the error becomes a non-issue, but the trajectories also become increasingly short – many more iterations are needed to wholly traverse the target distribution [12, 33].

Additionally, if $\tilde{\mathcal{B}} = \mathcal{B}$, then the integrator (3.1) results in stationary distribution $\pi(q,p) \propto e^{-H(q,p)}$, as the momentum update would simplify to

$$p(t) = p(t-t') - \frac{\eta}{2}\left(\frac{\partial U}{\partial q}(q(t-t')) - \mathcal{C}p(t-t') + \xi\right), \quad \xi \sim \mathcal{N}(0, 2\mathcal{C}). \tag{3.2}$$

However, this would be incredibly unlikely in practice [12].

Our approximation of the error model $\mathcal{B}$ can be either a scalar value or a diagonal matrix, and the user-defined model $\mathcal{C}$ can very well be arbitrary, as long as the equation $\mathcal{C} \succeq \mathcal{B}$ is satisfied. Chen et al. suggest [12] that $\tilde{\mathcal{B}}$ can be approximated as zero and $\mathcal{C}$ as a small scalar value without affecting the outcome of the algorithm. Typically, the step sizes are decreased polynomially

$$\eta_t = a(b+t)^{-\gamma},$$

where $\gamma \in [0.5, 1]$ and $t$ is the current iteration. Values $a$ and $b$ are used to scale the steps between desired values [33].

**Approximating the Error $\mathcal{B}$**

In case we need an accurate representation of the estimated error $\tilde{\mathcal{B}}$, we can approximate it via the diffusion matrix $\tilde{\mathcal{B}} = \frac{1}{2}\eta V(q)$, where $V(q)$ is the empirical Fisher information i.e., the covariance of the gradients w.r.t. parameter $\theta$ [2]

$$V(\theta|B_j) = \frac{1}{m-1}\sum_{i=1}^{m}\Big(g_i(\theta) - \bar{g}_m(\theta)\Big)\Big(g_i(\theta) - \bar{g}_m(\theta)\Big)^T.$$

Here $B_j$ is the $j$th mini batch of $m$ data points $\{x_1, \ldots, x_m\}$, $g(\theta)$ is the score i.e., the gradient of the log likelihood w.r.t data-point $x_i$:

$$g_i(\theta) = g_i(\theta|x_i \in B_j) = \nabla_\theta \log \pi(\theta|x_i)$$

and $\bar{g}(\theta) = \frac{1}{m}\sum_{i=1}^{m} g(\theta)$ is the score average.

If it happens that $\tilde{\mathcal{B}} = \mathcal{B}$ and the eq. (3.2) stands, the initially non-zero error $\mathcal{B} = \frac{1}{2}\eta V \to 0$ as $\eta \to 0$, and our estimation $\tilde{\mathcal{B}} = 0$ is still valid. Thus this bears little significance in our use case. We choose to omit the error approximation for the rest of the thesis, as it adds another parameters to consider for differential privacy calculations in the next chapters, without a significant advantage in the model precision.

## 3.2 The Implications of Naive Sub-sampling

In addition to the entropy increasing bias found by Chen et al. [12], Michael Betancourt argues [7] that the naively sub-sampled data sets will almost always introduce irreducible error to the estimate target distribution. Only when the data is redundant or tall i.e., either the data points overlap or there are only few dimensions per data point and the data points are more numerous than the data dimensions, may the sub-sampling yield acceptable results.

We assume again that the data is i.i.d. and that the total potential energy is the sum of all potential energies of the mini-batches

$$U(q) = \sum_{j \in J} U_j(q),$$

where $J$ denotes the index of the $N/m$ mini-batches. The full potential energy can then be approximated with the help of one mini-batch [7] as

$$\tilde{U}(q) \approx |J| U_j(q).$$

As we are now using only a fraction of the original data to calculate the potential energy, the symplectic integrators are compromised with a bias that interferes with the energy-preserving property of Hamiltonian dynamics. Two different approaches

are possible – either the data is sub-sampled between the simulated trajectories or repeatedly for every time-step inside the trajectory. In the spirit of not utilizing the data-set as whole, the MH-acceptance step is omitted for both methods [7].

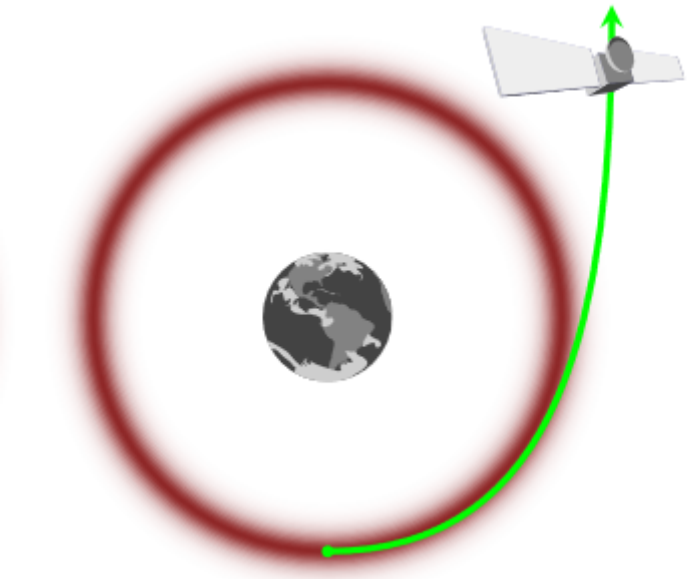

**Figure 3.1:** The error in the trajectory simulations exemplified. Without the exact gradients or corrective manoeuvres the trajectory of the sample (the satellite) strays from the actual Hamiltonian vector and biases the outcome, in this case striking the satellite out of the orbit. Figure by Michael Betancourt [7].

### 3.2.1 Data Sub-sampling Between the Trajectories

When the total potential energy is approximated with sub-sampled $U_j(q)$, the approximate Hamiltonian is likewise $H_j = K(p) + |J|U_j(q)$ [7]. This leads to an similar proportioned error on the every step of the integration process:

$$\begin{aligned} e^{\frac{\eta}{2}\vec{U}_j} \circ e^{\eta\vec{K}} \circ e^{\frac{\eta}{2}\vec{U}_j} &= e^{\eta\vec{H}_j} + \mathcal{O}(\eta^3) \\ &= e^{\eta\vec{H}_j - \eta\overrightarrow{\Delta V}_j} + \mathcal{O}(\eta^3). \end{aligned}$$

Here

$$\overrightarrow{\Delta U}_j = (\vec{U} - |J|\vec{U}_j) = -\left(\frac{\partial U}{\partial q} - |J|\frac{\partial U_j}{\partial q}\right)\frac{\partial}{\partial p}.$$

When the composition is again composed with itself $L$ times, the error over the whole trajectory is of second order. The constant in front of the $\mathcal{O}(\eta)$ term depends only on

$T = \eta L$:

$$\begin{aligned}\phi^{\tilde{H}_j}_{\eta,T} &\equiv \left(\phi^{U_j}_{\eta/2} \circ \phi^{K}_{\eta} \circ \phi^{U_j}_{\eta/2}\right)^L \\ &= e^{T\vec{H}_j} + \mathcal{O}(\eta^2) \\ &= e^{T(\vec{H} - \overrightarrow{\Delta U}_j)} + \mathcal{O}(\eta^2).\end{aligned}$$

As the target dimensions grow, the more the gradient approximation parts from the true gradient. This will result in either diminishing acceptance ratio for the potential acceptance step or biased estimates [7]. Highly redundant data might alleviate the bias, because the sample could then contain more of the original meaningful information, but this is not a realistic assumption for many cases.

The Biased trajectories compromise the preservation of the volume in the phase-space $(q, p)$, leading to malformed estimated target level set (see Fig. (3.1) for example). As opposed to error resulting from the symplectic integrators itself, the error from the approximate gradient is invariant to the step-size $\eta$ [7] and thus cannot be fixed by denser time discretization.

The difference between true and stochastic gradient rise as the sample size $B$ decreases, meaning higher batch sizes yield more accurate results. Betancourt concludes, that only batch sizes $B = N/2$ and up may give us acceptable results [7]. Here $N$ is the total amount of data-points.

### 3.2.2 Data Sub-sampling During a Trajectory

The second approach is to sample the original data set $L$ times during the trajectory calculations, one batch per iteration. We hope that the biases from different samples cancel each other out as the trajectory is computed, as is done during the SGHMC algorithm [12], before adding the friction term.

Similarly to the previous case, where one batch was used per trajectory, the simulated trajectories are biased by the errors [7]

$$\begin{aligned}B_1 &= -\frac{1}{L}\sum_{l=1}^{L} \overrightarrow{\Delta U}_{jl} \\ &= \vec{U} - \frac{|J|}{L}\sum_{l=1}^{L} \vec{U}_{jl} \\ &= -\left(\frac{\partial U}{\partial q} - \frac{|J|}{L}\sum_{l=1}^{L}\frac{\partial U_j}{\partial q}\right)\frac{\partial}{\partial p}\end{aligned}$$

and

$$B_2 = \eta\left(\left[\vec{H}, \overrightarrow{\Delta U}_{jl}\right] - \left[\vec{H}, \overrightarrow{\Delta U}_{jL}\right]\right).$$

While $B_2$ can be alleviated by having the last iteration of the trajectory simulation use the same batch of data as the first iteration, $j_L = j_1$ [7], the error $B_1$, that is due the energy difference between the actual gradient and the stochastic gradient, is harder to deal with. To completely erase the error, the average gradient of the used batches must yield the gradient of the full potential energy. Using all the sampled batches exactly twice during the trajectory will eradicate both errors $B_1$ and $B_2$, but we lose all the computational efficiency we are after by using stochastic gradients [7].

# 4. Differential Privacy

In this chapter we introduce the concept of differential privacy (DP) and means to find privacy bound for differentially private algorithms, focusing on the new method for computing the privacy guarantees tightly by Koskela et al. [19]. We also show that only minor modifications are necessary for SGHMC-algorithms to achieve differentially private sampling results.

When processing for example medical information of a group of patients in a hospital, or other similarly intimate or classified information, we do not want the adversaries to obtain the exact results. Even if the obtained data set cannot alone be used to identify people, today there are countless otherwise innocuous data sets available to public, and situationally these sets could be combined to infer identifying information with high probability.

Netflix published a large, anonymized data set of movie ratings to the public, for a competition with a goal to improve movie recommendation engine. The data set alone was free of outright identifying information, but by comparing the ratings and their timestamps to Internet Movie Database's (IMDB) user-given ratings, researchers were able to uniquely identify 99% of the $500,000$ users whose rating information was published. Further, the review and the movie watching history of the individual can be analysed to deduce more private informations, like political or religious beliefs [23]. Of course false-positives are bound to happen with such a limited auxiliary information, though the sheer size of the privacy break make a large part of these users prone to data breaches.

Users might use the same alias for multiple services, or in the worst case use their own name online. A single positive identifier could open the flood gates to considerable security breach, as the modern computers have the capability and resources to gather information from multiple giant data sets. Large companies such as Apple have started to adopt differential privacy in some of their systems [4], to protect the user data they gather from device analytics. This DP data is then used e.g. for better text and emoji prediction algorithms without compromising single users preferences.

In this chapter we define the concept of differential privacy, and introduce the common methods guaranteeing, that an output of a model does not compromise the

privacy of the used data.

## 4.1 The Definition of Differential Privacy

The $(\epsilon, \delta)$-differential privacy ($(\epsilon, \delta)$-DP) is defined with the help of remove/add neighbouring relation [19, 30]. We say that the data sets $X, Y \in \mathcal{X}^n$ are neighbours (denoted by $X \simeq_R Y$) when $Y$ is obtained from $X$ by removing or adding a data point. For example, otherwise identical matrices $X$ and $Y$ would differ by one additional row. These data sets are then called adjacent. For the substitution relation, see [19, 30].

The differential privacy is measured in variables $\epsilon$ and $\delta$. $\epsilon$ denotes the acceptable privacy loss or privacy budget – the probability of results drawn from adjacent sets $X$ and $Y$ differing greatly. We are also allowed to slightly diverge from this requirement, by exceeding the budget with the probability $\delta$ [14, 15, 31].

Mathematically, differential privacy is defined as follows [15, 31]: a randomized algorithm $\mathcal{A}$, also called the privacy mechanism, is $(\epsilon, \delta)$-differentially private within the domain $\mathcal{X}^n$, if for all measurable sets $S \subset \text{Range}(\mathcal{A})$ and for all $X, Y \in \mathcal{X}^n$ such that $X \simeq_R Y$, we have

$$\mathbb{P}(\mathcal{A}(X) \in S) \leq \exp(\epsilon)\mathbb{P}(\mathcal{A}(Y) \in S) + \delta. \tag{4.1}$$

The intuition here is that as long as the parameters $\epsilon$ and $\delta$ are small enough, the two datasets should output close to same results, even if one data point does not exist in one of them, i.e., the end result does not depend on any single data point. The potential attackers cannot recognize which data set was used to obtain the results, and the outcome is then much less useful for trying to infer any information about any one data point, or even the existence of the point altogether. When we set $\delta = 0$, we do not allow any variation exceeding our privacy budget and hence have stronger privacy guarantees, and we call the algorithm $(\epsilon)$-differentially private.

The definition is also robust against post-processing, and a composition of DP-algorithms preserves the DP-property [15]:

- If $\mathcal{A}$ is an $(\epsilon, \delta)$-DP algorithm, then for all algorithms $\mathcal{A}'$, the composition $(\mathcal{A}' \circ \mathcal{A})$ is also $(\epsilon, \delta)$-DP algorithm [31].
- If $\mathcal{A}'$ is $(\epsilon', \delta')$-DP, then the composition $(\mathcal{A}' \circ \mathcal{A})$ is $(\epsilon + \epsilon', \delta + \delta')$-DP algorithm [31].

This guarantees that the privacy protection cannot be circumvented or reversed by another algorithm. Also no auxiliary information available to the attacker affects the privacy guarantees. While the resulting $(\epsilon, \delta)$-DP is always the sum of associated

privacy parameters, this allows us to access sensitive data multiple times without losing all the privacy.

When the neighbouring datasets $X$ and $Y$ differ by size, i.e., one of the sets has more data points than the other, we call the achieved differential privacy unbounded. When the datasets are equal in size, but one or more data points have been modified between them, we achieve bounded differential privacy. Although most of the theory is interchangeable between the two, potentially achievable privacy bounds differ by a small margin. In this thesis we concentrate on the unbounded privacy bounds and the details on the bounded differential privacy can be found in [30].

One common strategy to achieve differential privacy is to inject the results with a controlled amount of noise, usually Laplace or Gaussian in nature [15, 31]. Increasing the amount of noise will give us more secure results, but the output accuracy will inevitably suffer. Thus finding the smallest amount of noise to achieve the given $(\epsilon, \delta)$-bounds is important. We will notice, that this is a surprisingly compatible method achieving privacy for sub-sampled HMC algorithms.

## 4.2 Gaussian Mechanism

The simplest way to ensure differential privacy of the model output is to add random noise to the results. To adjust the noise injected to the model, it is calibrated to its $\ell_2$-sensitivity. for an arbitrary $d$-dimensional function $f : \mathcal{X}^n \to \mathbb{R}^d$, the sensitivity is defined as [1, 31]

$$\Delta_2 f = \sup_{X \simeq_R Y} ||f(X) - f(Y)||_2,$$

To achieve $(\epsilon, \delta)$-DP for the function $f$, we define the Gaussian Mechanism output of that function as

$$\tilde{f}(X) = f(X) + \mathcal{N}(0, \sigma^2).$$

When we select the standard deviation $\sigma$ such that

$$\sigma \geq \Delta_2 f \frac{\sqrt{2 \log(1.25/\delta)}}{\epsilon},$$

the output $\tilde{f}(X)$ is $(\epsilon, \delta)$-differentially private [15, 31]. While fast and straight-forward to implement, with this approach we use all the available privacy budget at once, after the algorithm has finished the execution. We can reduce the amount of injected noise and thus increase the accuracy of the algorithm by taking into account the actual privacy loss iteration by iteration [1, 19]. Moreover, the privacy loss can be reduced by using sub-sampling of the data [30, 34] at each iteration. This also corresponds to the mechanism of the DP-SGD algorithm.

## 4.3 Moments Accountant Method

For the moments accountant (MA) method, we implement an "accountant" procedure to the computations. The accountant calculates the privacy cost at every access of the data set and accumulates them up to the privacy budget, allowing us to find tighter privacy bounds for given $\epsilon$ and $\delta$ [1]. The composability properties of $(\epsilon, \delta)$-DP functions enables us to add this new procedure without compromising the privacy.

The privacy loss for the output $o$ of an $(\epsilon, \delta)$-DP algorithm $\mathcal{A}$, adjacent data sets $X$ and $Y$ and arbitrary auxiliary input AUX, that does not depend on the data, is defined as

$$\mathcal{L}(o|\mathcal{A}, \text{AUX}, X, Y) \triangleq \log \frac{\mathbb{P}(\mathcal{A}(\text{AUX}, X) = o)}{\mathbb{P}(\mathcal{A}(\text{AUX}, Y) = o)},$$

or more concisely,

$$\mathcal{L}_{X/Y}(t) = \log \frac{f_X(t)}{f_Y(t)}, \tag{4.2}$$

where the functions $f_X$ and $f_Y$ are probability density functions of the algorithm $\mathcal{A}$ for the datasets $X$ and $Y$, respectively, at any point $t \in \mathbb{R}$ [1, 19]. Though various kinds of algorithms can be differentially private, for simplicity's sake in this thesis we only consider algorithms, that have $d$-dimensional real-valued output, vector $\mathcal{A} : \mathcal{X}^n \to \mathbb{R}^d$. The differentially private SGD implemented for mechanisms in $\mathbb{R}^d$ can be analyzed using one-dimensional density functions [1, 19].

Moments accountant method utilizes the advanced composition lemma [1, 15]: for all $\epsilon, \delta, \delta' \geq 0$, the class of $(\epsilon, \delta)$-DP algorithms satisfy $(\epsilon', k\delta + \delta')$-DP under $k$-fold adaptive composition for

$$\epsilon' = \sqrt{2k \log(1/\delta')}\epsilon + k\epsilon(e^\epsilon - 1).$$

The lemma allows us to sacrifice some amount of $\delta$ for considerable improvement in the privacy budget $\epsilon$ [15, 31]. For a slightly higher $\delta$, we net a smaller $\epsilon$.

The moments accountant is itself defined as follows: for a randomized algorithm $\mathcal{A}$, two adjacent data sets $X$ and $Y$ and the auxiliary input AUX, that is independent of the data sets, the moments accountant with an integer parameter $\lambda$ is

$$\alpha_{\mathcal{A}}(\lambda) \triangleq \max_{\text{AUX}, d, d'} \alpha_{\mathcal{A}}(\lambda|\text{AUX}, X, Y),$$

where the function

$$\alpha_{\mathcal{A}}(\lambda|\text{AUX}, X, Y) \triangleq \log \mathbb{E}\left(\exp(\lambda \mathcal{L}(\mathcal{A}, \text{AUX}, X, Y))\right)$$

is the logarithm of the function at $\lambda$, which generates the moment i.e., the $\lambda$th moment of the privacy loss variable [1, 21]. The moments are composable: for an algorithm

$$\mathcal{A} = \mathcal{A}_1 \circ \mathcal{A}_2 \circ \ldots \mathcal{A}_k$$

and any integer $\lambda$ stands

$$\alpha_{\mathcal{A}}(\lambda) = \sum_{i=1}^{k} \alpha_{\mathcal{A}_i}(\lambda).$$

Here we consider all previous outputs of moments $\alpha_{\mathcal{A}_j}$, $j < i$, as auxiliary input for the moment $\alpha_{\mathcal{A}_i}$ [1, 21].

For any $\epsilon > 0$, the algorithm $\mathcal{A}$ is $(\epsilon, \delta)$-DP for [1, 21]

$$\delta = \min_{\lambda} \exp(\alpha_{\mathcal{A}}(\lambda) - \lambda\epsilon).$$

With these properties, we can calculate the privacy bounds. Let $f$ be a function $f : \mathcal{X}^n \to \mathbb{R}^d$ such that $||f(\cdot)||_2 \leq 1$. Let $\sigma \geq 1$, and B the size of a mini-batch with sampling probability $q = B/N$. If $q < 1/(16\sigma)$, for any integer-valued $\lambda$ such that

$$0 < \lambda \leq \sigma^2 \log(1/(q\sigma)),$$

the algorithm

$$\mathcal{A}(X) = \sum_{i \in B} f(x_i) + \mathcal{N}(0, \sigma^2) \tag{4.3}$$

satisfies [1, 21]

$$\alpha_{\mathcal{A}}(\lambda) \leq \frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}(q^3).$$

Abadi et al. [1] have created an algorithm to calculate the privacy bound for $\epsilon$ and $\delta$ according to $\lambda$.

All in all, the moments accountant method gives the $(\epsilon, \delta)$-differential privacy of the mechanism (4.3) after $T$ iterations. The theoretical results of [1] show that overall the algorithm $\mathcal{A}$ achieves $\mathcal{O}\left(q\epsilon\sqrt{T}, \delta\right)$-differential privacy [1]. As the totalling differential privacy is dependent on many hyper-parameters including the model batch size and iteration time, tuning the algorithm for maximum privacy can be difficult [1].

## 4.4 Computing the Privacy Profiles Tightly via Privacy Loss Distribution

To compute tight values for privacy parameters $\epsilon$ and $\delta$, we re-define the differential privacy (4.1). Along with the privacy loss (4.2) we define the privacy loss distribution (PLD) of the randomized algorithm $\mathcal{A}(X)$ over $\mathcal{A}(Y)$, $X \simeq_R Y$ as $\omega : \Omega \to \mathbb{R}$ [19]:

$$\omega(S) = \int_{\left\{t \in \mathbb{R} | \mathcal{L}_{X/Y}(t) \in S\right\}} f_X(t)dt,$$

where $\omega$ is a set of all measurable subsets of $\mathbb{R}$ and $S \in \Omega$. $f_X$ and $f_Y$ are the probability density functions of the algorithms $\mathcal{A} : \mathcal{X}^n \to \mathbb{R}$ for the sets $X$ and $Y$, respectively.

When $\mathcal{L}_{X/Y}$ is a continuously differentiable and bijective function, for a measurable set $S \subset \mathbb{R}$ it holds that [19]

$$\omega(S) = \int_S \frac{d\omega}{dy} dy, \tag{4.4}$$

where

$$\frac{d\omega}{dy} = \begin{cases} f_X\left(\mathcal{L}^{-1}_{X/Y}(y)\right) \dfrac{d\mathcal{L}^{-1}_{X/Y}(y)}{dy}, & \text{if } y \in \mathcal{L}_{X/Y}(\mathbb{R}) \\ 0, & \text{otherwise.} \end{cases}$$

This is proven by selecting $t = \mathcal{L}^{-1}_{X/Y}(y)$ [19]:

$$\begin{aligned} \omega(S) &= \int_{\{t \in \mathbb{R} | \mathcal{L}_{X/Y}(t) \in S\}} f_X(t) dt \\ &= \int_{S \cap \mathcal{L}_{X/Y}(\mathbb{R})} f_X\left(\mathcal{L}^{-1}_{X/Y}(y)\right) \frac{d\mathcal{L}^{-1}_{X/Y}(y)}{dy} dy. \blacksquare \end{aligned}$$

For one-dimensional distributions, the randomised algorithm $\mathcal{A}$ satisfies $(\epsilon, \delta)$-differential privacy, when

$$\int_S f_X(t)dt \leq e^\epsilon \int_S f_Y(t)dt + \delta \quad \text{and} \quad \int_S f_Y(t)dt \leq e^\epsilon \int_S f_X(t)dt + \delta,$$

for all sets $S \subset \mathbb{R}$ and for all adjacent data sets $X$ and $Y$ [19]. The algorithm $\mathcal{A}$ is called tightly $(\epsilon, \delta)$-DP, when no $\delta'$ satisfies $\delta' < \delta$ such that $\mathcal{A}$ satisfies $(\epsilon, \delta')$-DP [19]. For tightly $(\epsilon, \delta)$-DP algorithm, the parameter $\delta$ is computed as [19]

$$\delta = \max_{X \simeq Y} \left\{ \int_{\mathbb{R}} \max\{f_X(t) - e^\epsilon f_Y(t), 0\} dt, \int_{\mathbb{R}} \max\{f_Y(t) - e^\epsilon f_X(t), 0\} dt \right\}. \tag{4.5}$$

The previous equation (4.5) can be represented by integrals:

$$\delta = \max_{X \simeq Y} \max\left\{\delta_{X/Y}, \delta_{Y/X}\right\},$$

where

$$\delta_{X/Y} = \int_{\mathcal{L}_{X/Y}(\mathbb{R}) \cap [\epsilon, \infty[} (1 - e^{\epsilon - y}) f_X\left(\mathcal{L}^{-1}_{X/Y}(y)\right) \frac{d\mathcal{L}^{-1}_{X/Y}(y)}{dy} dy$$

and

$$\delta_{Y/X} = \int_{\mathcal{L}_{Y/X}(\mathbb{R}) \cap [\epsilon, \infty[} (1 - e^{\epsilon - y}) f_Y\left(\mathcal{L}^{-1}_{Y/X}(y)\right) \frac{d\mathcal{L}^{-1}_{Y/X}(y)}{dy} dy.$$

With these equations, we represent the parameter $\delta$ for one-dimensional, continuous output of a tightly $(\epsilon, \delta)$-DP algorithm $\mathcal{A}$ as

$$\delta = \max_{X \simeq Y} \max\left\{\delta_{X/Y}, \delta_{Y/X}\right\}, \tag{4.6}$$

where

$$\delta_{X/Y} = \int_\epsilon^\infty (1 - e^{\epsilon - y}) \frac{d\omega_{X/Y}}{dy} dy \quad \text{and} \quad \delta_{Y/X} = \int_\epsilon^\infty (1 - e^{\epsilon - y}) \frac{d\omega_{Y/X}}{dy} dy.$$

The distribution functions $\frac{d\omega_{X/Y}}{dy}$ and $\frac{d\omega_{Y/X}}{dy}$ are defined similarly as in (4.4) [19]. To utilize this representation for a composition of algorithms, we also need to evaluate the privacy loss distribution for that composition.

Let $\mathcal{A}, \mathcal{A}' : \mathcal{X}^n \to \mathbb{R}$ be independent random algorithms that output one-dimensional random variables with continuous densities. The privacy loss density function of the composition of $\mathcal{A}$ and $\mathcal{A}'$ (be it either $\mathcal{A} \circ \mathcal{A}'$ or $\mathcal{A}' \circ \mathcal{A}$) is defined

$$\frac{d\omega^{c}_{X/Y}}{du}(y) = \int_{-\infty}^{\infty} \frac{d\omega_{X/Y}}{dy}(t) \frac{d\omega_{X'/Y'}}{dy}(y-t)dt. \tag{4.7}$$

Here the derivative $d\omega_{X'/Y'}/dy$ stems from the algorithm $\mathcal{A}'$ [19]. The proof for the above equations can be found in [19].

With the help of the definitions (4.6) and (4.7), we obtain an integral representation for $\delta(\epsilon)$ [19]. For $k$ consecutive applications of an algorithm $\mathcal{A}$ and for some $\epsilon > 0$, the composition of the algorithms is tightly $(\epsilon, \delta)$-DP for $\delta$ given by the function

$$\delta = \max_{X \simeq Y} \max \left\{ \delta_{X/Y}, \delta_{Y/X} \right\},$$

where

$$\delta_{X/Y} = \int_{\epsilon}^{\infty} (1 - e^{\epsilon - y}) \left( \frac{d\omega_{X/Y}}{dy} *^{k} \frac{d\omega_{X/Y}}{dy} \right)(y) dy. \tag{4.8}$$

Here $(d\omega_{X/Y}/dy) *^{k} (d\omega_{X/Y}/dy))(y)$ is shorthand for the privacy loss density function, which is the result of compositing the algorithm $\mathcal{A}$ with itself $k$ times [19].

This results enables us to numerically compute a tight bound of $\delta$ for an algorithm with appropriately pre-set $\epsilon$, or vice-versa. This technique is shown to give tighter privacy bounds than both moments accountant and Rényi accountant methods [19]. The inverse of (4.8), i.e., the function $\epsilon(\delta)$, can be solved e.g. by Newton's method by solving for $\tilde{\delta}$ such that $|\delta(\epsilon_l) - \tilde{\delta}| \leq tol$, for Newton's iteration step $l$ and pre-set tolerance value $tol$ [19]. We skip the details of the numerical method to evaluate (4.8) and refer to [19] for more details.

# 5. Implementing the Differential Privacy to HMC

The sub-sampled HMC methods all add some amount of stochastic error to the algorithm. Incidentally, we can use this inherent noise in order to achieve differential privacy for our model, leaving us only small adjustments to make [21, 31]. Mainly we focus on the correct step size sequencing such that we minimize the stochastic noise per iteration, while still having large enough error to mask our results. To bound the effect of single data points on the Hamiltonian trajectories, we norm-clip the gradients using a constant $L$:

$$\nabla \tilde{U} = \frac{\nabla U}{\max\left(1, \frac{||\nabla U||_2}{L}\right)}. \tag{5.1}$$

This operation does not impact the convergence rates of the algorithms, as in practice its effect is similar to using adaptive step size [21, 29]. Furthermore, the ideal step-size sequence $\eta$ would be scaled by $L$ too [21]. If a too small value for $L$ is used, the added error will dominate the actual gradients and we are unable to achieve good results [29]. The gradient-clipping operation both satisfies the Lipschitz condition and ensures that the computed gradients satisfy the Lipschitz condition

$$|\nabla \tilde{U}(x) - \nabla \tilde{U}(x')| \leq L|x - x'| \quad \forall x, x' \in X$$

for the constant $L$ [21]. The constant $L$ is bound to the models log-likelihood $\pi(x_i|q)$ and is typically small for continuous distributions, which do not increase or decrease at exponential rate at any point [31].

Due to usage of adaptive step sizes and restricted maximum iteration time, which is one parameter for the privacy parameter computations [1], it is good practice to search for the good starting point for the algorithm. If the location of the target set is unknown, we would waste the iterations by converging to the set and then not have enough time to explore the target set adequately. One proposed method for finding the starting point near the target set is differentially private one posterior sample method [31], where only one $(\epsilon, \delta)$-DP sample is used from the sampler.

In the next sections we adjust the stochastic gradient Hamiltonian Monte Carlo to be compliant with $(\epsilon, \delta)$-differential privacy requirements.

## 5.1 Naive Privacy Analysis with the Gaussian Mechanism

During the HMC algorithm, we only access the source data for the gradient calculations [24]. The SGHMC is very similar to stochastic gradient descent with momentum and stochastic gradient Langevin dynamics -algorithms: we are able to achieve DP with the same technique [31].

As we sample the dataset one sub-set at the time and without the MH acceptance step, we are bound to inject some stochastic error to the algorithm, which is partially dependent on the step sizes $\eta_t$ at any iteration $t$. By scaling the decreasing step sizes $\eta$ appropriately, we mask the results with this noise while still preserving the computational accuracy. While the privacy mechanism doesn't demand it, we still must reduce the step size to zero to ensure accurate sampling from the posterior distribution [12, 31].

The one of requirements for the functional SGHMC algorithm is the injected error model $\mathcal{C}$, which should dominate the estimate of the actual error model $\tilde{\mathcal{B}}$. The error $\mathcal{C}$ is set such that, along with the step sizes $\eta$, it will also dominate the Gaussian mechanism error at every iteration, which would otherwise be added to the algorithm [31]. As suggested by Wang et al. [31] we achieve $(\epsilon, \delta)$-DP:

$$\begin{aligned} & 2(\mathcal{C} - \tilde{\mathcal{B}})\eta_t \succ \frac{128NTL^2}{B\epsilon^2} \log\left(\frac{2.5NT}{B\delta}\right) \log\left(\frac{2}{\delta}\right) \eta_t^2 \\ \Leftrightarrow & \frac{2(\mathcal{C} - \tilde{\mathcal{B}})}{\eta_t} \succ \frac{128NTL^2}{B\epsilon^2} \log\left(\frac{2.5NT}{B\delta}\right) \log\left(\frac{2}{\delta}\right) \qquad \text{for all } \eta_t \in \eta, \end{aligned} \tag{5.2}$$

where $L$ is the Lipschitz constant and $B$ the size of the mini-batch. This holds true, when $\eta_t \to 0$ [31]. Then, the injected noise from the error models $\tilde{\mathcal{B}}$ and $\mathcal{C}$ will dominate the stochastic error accordingly.

It is essential for the accuracy of the algorithm that the step sizes reach zero. This leads to unpractically low bounds for the step size and usually we have to tolerate some amount of residual error and set $\eta_t = \max(\eta_t, \eta_{min})$, for some floor value $\eta_{min}$ [31].

## 5.2 Moments Accountant

To achieve $(\epsilon, \delta)$-DP for SGHMC, we only have to switch to a sequence of the step sizes $\eta$ instead of a constant step size. It is shown that the step size sequence of $\eta_t = \mathcal{O}(t^{-\alpha})$ yields the optimal values $\eta_t$ regarding the mean square error bound [21].

If we select the decrease rate for $\eta$ to be of $\mathcal{O}(t^{-1/3})$, there exists two positive constants $c_1$ and $c_2$. Assume the sampling probability $q = m/N$, Lipschitz constant $L$ and the number of total iterations $T$. SGHMC algorithm achieves $(\epsilon, \delta)$-DP for any $\epsilon < c_1 q^2 T^{2/3}$, when the step size at iteration $t$ satisfies [21]:

1. $\eta_t \leq \dfrac{N}{L^2}$
2. $\eta_t > \dfrac{q^2 N}{256 L^2}$
3. $\eta_t < \dfrac{\epsilon^2 N t^{-1/3}}{c_2^2 L^2 T^{2/3} \log(1/\delta)}$

The first condition is satisfied in practice, as typically $N/L^2 \gg \eta_t$, especially when the data set is large. With appropriately selected $q$ and $L$, also the second condition is satisfied [21]. Empirically, in this case it is shown that $c_2 < 128$, to differ from the method by Wang et al. [21, 31].

Along with the new upper bound (similarly to the previous method (5.2)), in order to have the injected friction noise to dominate the other error sources, which in this case is the injected $\mathcal{N}(0, \eta_t/N)$[21], we set

$$\frac{2(\mathcal{C} - \tilde{\mathcal{B}})}{\eta_t} \succ \frac{\eta_t}{N} I_d \quad \Leftrightarrow \quad \frac{2(\mathcal{C} - \tilde{\mathcal{B}})}{\eta_t^2} \succ \frac{1}{N} I_d \quad \text{for all } \eta_t \in \eta, \tag{5.3}$$

where $I_d$ is the identity matrix of size $d \times d$.

In practice this holds true with any moderate sized data set and mainly restricts the maximum i.e., the first couple step-sizes $\eta_0, \eta_1, \ldots$. Considering that the lack of corrective MH acceptance step present in SGHMC, high step sizes predispose the model to biases. Thus this affects the model mostly when the distance between target set and the starting point is known to be large: the long leap frog trajectories converge to the target set with less iterations. The parameters $c_1$ and $c_2$ and thus the privacy parameters $\epsilon$ and $\delta$ can then be computed numerically with the Moments accountant algorithm by Abadi et al. [1] *.

The moments accountant method achieves the same privacy requirements than the original method by Wang et al. with higher bound for the step size sequence $\eta_t$, or reciprocally generates tighter bounds for $\epsilon$ and $\delta$ with the same step size.

*Code available at https://github.com/ageron/tensorflow-models/tree/master/differential_privacy

## 5.3 Tightly Computed Privacy Profile

Similarly to moments accountant method, due to custom noise model, it is enough to set the injected $\mathcal{C}$ large enough to dominate the other error sources. As the step sizes decrease ($\eta_t \to 0$), the injected noise is scaled accordingly. The prerequisite for the error models $\mathcal{C}$ and $\tilde{\mathcal{B}}$ is the same as above (5.3). Koskela et al. have provided code* for calculating the privacy profile via the PLD method [19].

## 5.4 Computing the Privacy Bounds

Both Moments accountant and privacy loss distribution method are partially invariant to the actual sampling algorithm. Initially the privacy mechanisms introduced here are based on stochastic gradient descent algorithm, which is close relation to sampling algorithms introduced in the thesis [12, 31]. Thus all, both Abadi's method and the improved, analytical moments accountant by Wang et al. [30] and the PLD method by Koskela et al. [19], calculate the privacy parameters $(\epsilon, \delta)$ similarly: by the sampling probability $q = m/N$ and the standard deviation $\sigma$ in the privacy mechanism (4.3)

$$\mathcal{A}(X) = \sum_{i \in B} \tilde{\nabla} f(x_i) + \mathcal{N}(0, \sigma^2).$$

The totalling parameters $\epsilon$ and $\delta$ for an algorithm are determined by times the input data is accessed, which in this case is determined by the leapfrog trajectory lengths and the amount of total iterations: $T \cdot J$. The mechanism $\mathcal{A}$ is composited with itself as many times in both moments accountant and privacy loss distribution methods.

For moments accountant and PLD, it stands that $\sigma \sim \mathcal{O}\left(\frac{1}{\sqrt{\eta}}\right)$ [1, 19], and the mechanism is at most an unit vector: $||\tilde{\nabla} f(x_i)||_2 \leq 1$. To achieve this form, we scale the algorithm with the sequence of step sizes $\eta_t$, and the standard deviation $\sigma$ is scaled accordingly. Thus the scaled the mechanism yields

$$\mathcal{A}'(X) = \eta_t \sum_{i \in B} \tilde{\nabla} f(x_i) + \mathcal{N}\left(0, 2(\mathcal{C} - \tilde{\mathcal{B}})\right) \sqrt{\eta_t}.$$

As we clip the gradients (5.1) in the SGHMC algorithm, and as such scale the mechanism by the constant $L$. We have $||\tilde{\nabla} f(x_i)||_2 \leq L$ and the mechanism is scaled accordingly:

$$\mathcal{A}'(X) = \eta_t L \left( \sum_{i \in B} \tilde{\nabla} f(x_i) + \mathcal{N}(0, \sigma^2) \right),$$

where

$$\sigma^2 = \frac{2(\mathcal{C} - \tilde{\mathcal{B}})}{\eta_t L^2}$$

*Code available at https://github.com/DPBayes/PLD-Accountant/

and further

$$\sigma = \sqrt{\frac{2(\mathcal{C} - \tilde{\mathcal{B}})}{\eta_t L^2}}.$$

This $\sigma$ will be used to determine the privacy parameters $\epsilon$ and $\delta$.

Previously, in Sec. 3.1, we established that differentially private algorithms are robust against post-processing. Thus, as scaling the mechanism equate to post-processing, we are able to scale algorithm with the clipping constant without losing the privacy.

# 6. Experiments

We test the introduced HMC implementations in three different ways: first, we demonstrate the biases in the phase space $(q, p)$ introduced by the sub-sampling of the data, and then we will evaluate the performance of the methods in sampling a normal distribution, and in a multi-class classification task using softmax regression. Finally, we compute the achieved differential privacy in the classification task for the stochastic gradient HMC and compare the achieved privacy bounds between moments accountant and tight bound from the PLD method.

## 6.1 Conservation of the Phase Space Volume

We repeat Betancourt's experiment [7] and generate $N = 500$ data-points from one-dimensional Gaussian distribution $\mathcal{N}(1, 2^2)$ in order to approximate its mean as a posterior distribution with the prior $\pi(\mu) = \mathcal{N}(\mu|m, s^2)$, where $m = 0$ and $s = 1$. The log-likelihood for the data set is thus

$$\pi(\mathbf{y}|\mu) = \sum_{n=1}^{N} \mathcal{N}(y_n|\mu, \sigma^2)$$

with $\mu = 1$ and $\sigma = 2$. For each batch $B_j$, $j \in J$ (see Sec. (3.1)) the potential energy $U_j$ can then be calculated as

$$U_j = \pi(\mathbf{y}^j|\mu) + \pi(\mu) = \frac{\sigma^2 + Ns^2}{\sigma^2 s^2} \left( \mu - \frac{\left(\frac{1}{B}\sum_{n=(j-1)B+1}^{jB} x_n\right) Ns^2 + m\sigma^2}{\sigma^2 + Ns^2} \right)^2,$$

where $\mathbf{y}^j$ is the vector of data points belonging to $j$th batch. Thus, we have the gradient of the potential energy

$$\nabla U_j = -2 \left( \frac{\sigma^2 + Ns^2}{\sigma^2 s^2} \right) \left( \mu - \frac{\left(\frac{1}{B}\sum_{n=(j-1)B+1}^{jB} x_n\right) Ns^2 + m\sigma^2}{\sigma^2 Ns^2} \right).$$

#### Bias Inside One Trajectory

We map the phase space $(q, p)$ after a single leapfrog trajectory of length $L = 500$ from the whole data set and sub-sampled sets iterations with step-size $\eta = 0.05$. When a

trajectory inside the Hamiltonian vector field is calculated from a single sample, with the batch-size $B = 10$, the difference between the phase spaces of sub-sampled and full data is evident, and the energy retained by the sub-sample is dependent on the data-points assigned to that batch – The results will differ considerably depending on how closely the sub-sampled gradient resembles that of the full data. Larger batch sizes diminish the difference, but cannot mimic the full data set in full. The Figures (6.1) were chosen to present the typical scenario. Larger (Fig. (6.2)) or smaller step size does not change the outcome drastically.

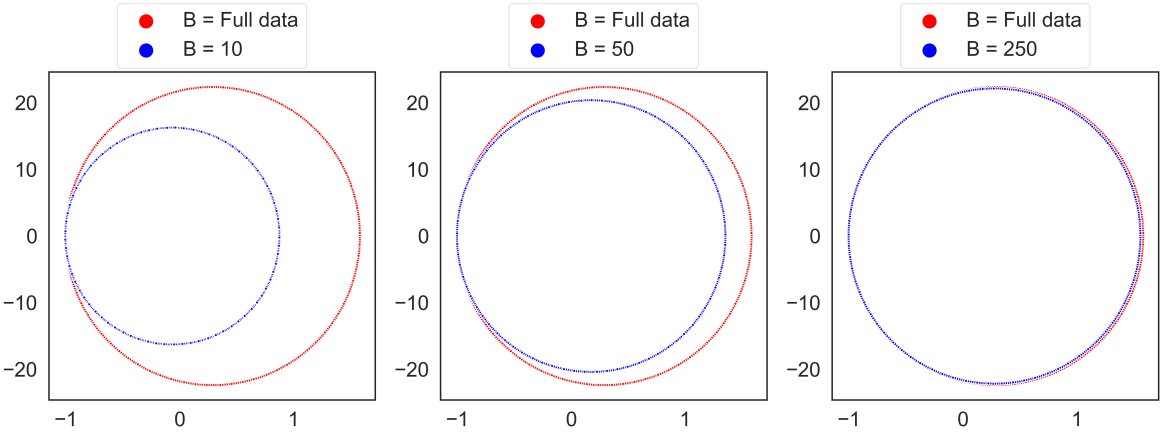


**Figure 6.1:** Phase space $(q, p)$ evaluated over one integration step with subsampled data. At the step size $\eta = 0.05$ the difference between the actual energy and the sub-sampled volume decreases with larger batch sizes.

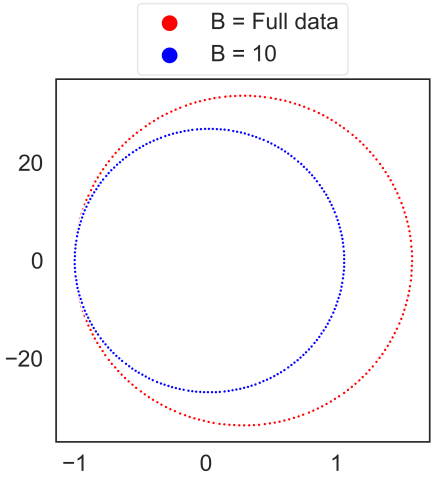


**Figure 6.2:** Phase space $(q, p)$ evaluated for the full data and Batch size of 10 data points. Here $\eta = 0.1$ for comparison.

### Bias Over Multiple Trajectories

When simulating the Hamiltonian flow with the leapfrog integrator of length $L = N/B$ for $T = 500$ iterations, the mini-batched trajectories stray far from the actual phase space in the Figures (6.3). Again, with larger sub-sample sizes the error decreases greatly, but will not reach the same accuracy as the whole data set. In this scenario lowering the step size $\eta$ gives us considerable gains in phase space accuracy (Fig. (6.4)). However, with small $\eta$ we would need more iterations $T$ to traverse the target set, losing the potential gains in the computational performance.

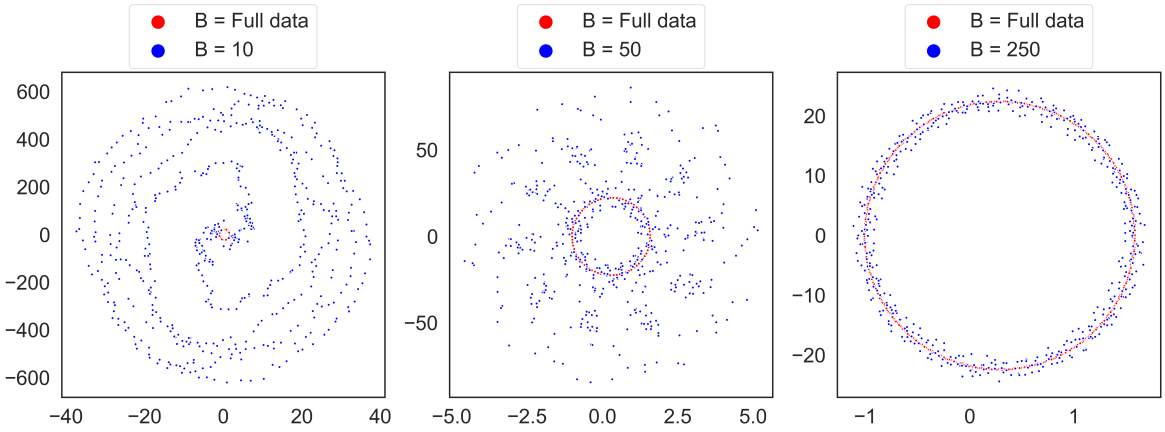


**Figure 6.3:** Phase space $(q, p)$ evaluated over the leapfrog integrator with subsampled data. At the step size $\eta = 0.05$ the difference between the actual energy and the sub-sampled volume decreases with larger batch sizes. Note that the actual phase space i.e., the red circle, stays the same size in every image.

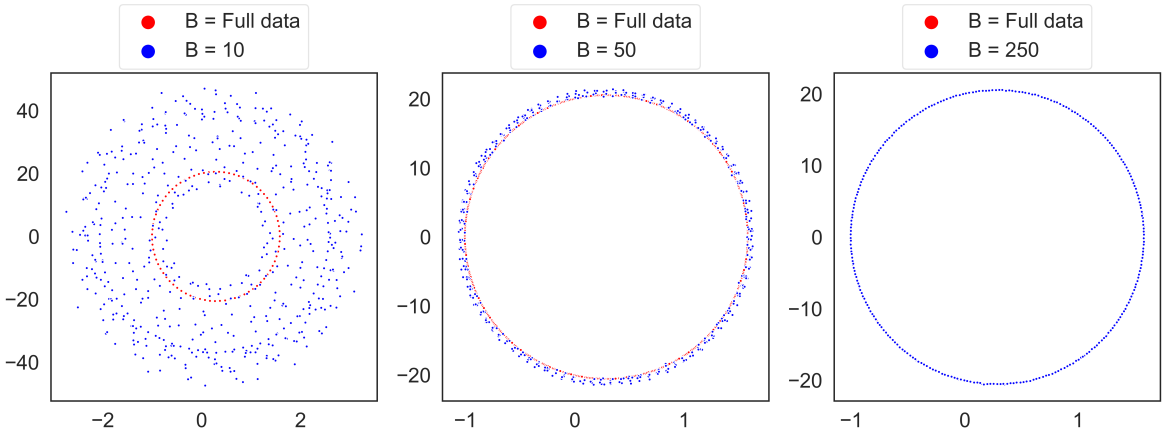


**Figure 6.4:** Phase space $(q, p)$ evaluated over the leapfrog integrator with subsampled data. At the step size $\eta = 0.005$ the differences between the actual energy and the sub-sampled volume are not as radical. Note that the actual phase space i.e., the red circle, stays the same size in every image.

When the friction term with $\mathcal{C} = 8 \cdot 10^{-4}$ is introduced to leapfrog method (Figure (6.5), it balances the outcome by forcing more trajectories to end up inside the actual

phase space i.e., having less total energy than the baseline. A singular trajectory will still stray from the target, but the mean of all trajectories will be closer to baseline.

The amount of friction needed is dependent on the other parameters of the model – here we fit the friction to accommodate batch size $B = 50$. At larger batch sizes the friction starts to have a detrimental effect, as the algorithm starts to put samples inside the target phase space rather than outside, implying that adjusting the friction term to the batch size is necessary for good accuracy. At the step size $\eta = 0.05$ the results from the friction term are between the standard leapfrog implementations at step sizes $\eta = 0.005$ and $\eta = 0.05$.

However, SGHMC with friction term also fails to accurately reproduce the exact phase space of the standard HMC algorithm. It brings the mean of all energy states closer to the center of the actual phase space, which can only help reduce the sub-sampling bias. Thus, we cannot consider it an analytically robust method for eliminating the said bias, and Betancourt's [7] assertion stands. In the next sections we see how the SGHMC algorithm performs in practice against the standard HMC method.

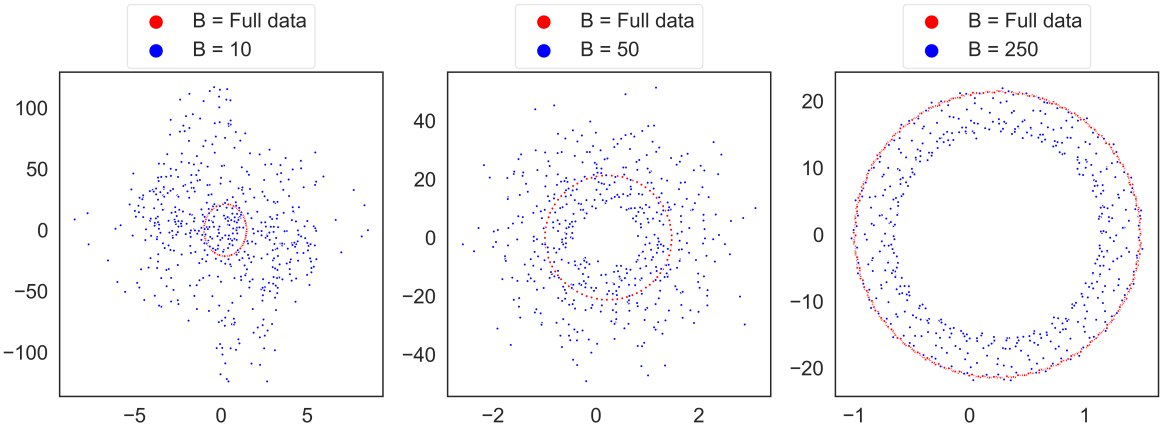


**Figure 6.5:** Phase space $(q, p)$ evaluated over the leapfrog integrator with the friction term and sub-sampled data. At similar step size $\eta = 0.05$ the differences between the actual energy and the sub-sampled volume are smaller at $B = 50$, but too high friction quickly induces negative bias when the batch size increases. Similarly a too small friction term cannot control the bias at the lower batch size. Note that the actual phase space i.e., the red circle, stays the same size in every image.

## 6.2 Normal Distribution

We use the same distribution and prior to gauge the accuracy of different algorithms in a simple optimization task by finding the mean of the target distribution. The gradients are clipped with the constant $L = 0.7$ and the error model for SGHMC is set to $\mathcal{C} = 0.005$. For the standard HMC we set the step size $\eta = 0.1$ and the amount of

leapfrog steps $J = 15$. For the stochastic algorithms we set $J = N/B$ and the sequence of step size as

$$\eta_t = \beta \cdot t^{-1/3}, \quad t \in T,$$

with the multiplier set $\beta = 1$. This is the same implementation as Betancourt's implementation [7], where each sub-sample is used once in the leapfrog integrator per iteration. The amount of total iterations is set $T = 500$. 50% of the drawn samples is discarded as the burn-in data.

While standard HMC implementation samples the mean with high accuracy, the naive SGHMC without the friction term biases towards uniform distribution and fails to find an accurate mean, until the step size multiplier is lowered to 0.002 and the sampling probability is set to $q = 0.10$. The SGHMC implementation also has problems with low sampling ratios, but the results improve with $q = 0.10$ and we begin to see consistent accurate estimates. Considering that with $q = 0.01$ the size of one batch is only five samples, this is unsurprising.

With no Metropolis-Hastings acceptance step, both stochastic models tend to wider standard deviations than the basic HMC algorithm. With optimization task this might be a non-factor, but e.g. confidence intervals from the approximate posterior might not be accurate. The sampled values, root-mean-square errors and standard errors are seen in the table (6.1).

| **Algorithm** | **Mean** | **Sample** $s$ | **Std. error** | **RMSE** |
|---|---|---|---|---|
| Actual | 1.0011 | - | - | - |
| Standard HMC | 0.9932 | 0.0789 | 0.0038 | 0.0004 |
| Naive SGHMC | 71 707 | 5079 | 329 | 4535 |
| SGHMC w/ $q = 0.01$ | 0.4203 | 32.7348 | 2.0662 | 0.0367 |
| Naive HMC w/ $q = 0.1$, $\beta = 0.002$ | 1.0537 | 0.0307 | 0.0019 | 0.0033 |
| SGHMC w/ $q = 0.1$ | 1.0596 | 2.1496 | 0.1356 | 0.0037 |

**Table 6.1:** The mean and sample standard deviations $s$ of different HMC algorithms sampling a normal distribution with a prior $\pi(\theta) \sim \mathcal{N}(0, 1)$. We measure the accuracy of the estimate with standard error and the precision with the root mean square error. Naive and SGHMC implementations are first listed with same sampling parameters and then with parameters yielding accurate results.

## 6.3 Logistic Regression

The performance of the HMC algorithms is evaluated by multi-class logistic regression (i.e. softmax regression) against stochastic gradient descent augmented with limited-memory bfgs optimizer from Python's Scikit Learn library. The dataset used is Radon

dataset from the collection by Måns Magnusson [22] and we predict the radon floor measure value from two log-variables.

### Softmax Function and Cross-Entropy Loss

The dataset of size $N$ is classified into $K = 5$ classes $k$. We denote the class labels as matrix $Y$, which we encode into $N \times K$ sized one-hot matrix. We use two predictive covariates for each data point, formed into matrix $X^{N\times 2}$. Thus our regression model for the $n$th data point belonging in the class $k$, so $d = (n, k)$, is

$$z_d = \theta_{(1,k)} + \theta_{(2,k)}x_{(n,2)} + \theta_{(3,k)}x_{(n,3)}.$$

Here $x \in X$ denotes the feature vector of the $n$th row of the matrix $X$. To accommodate the intercept parameter $\theta_1$, we concatenate a column of ones, $x_1$, with the matrix $X$ and can compute the net input matrix $Z$ for the $n$th row and $k$:th class:

$$Z_n = \theta_{(1,k)}x_{(n,1)} + \theta_{(2,k)}x_{(n,2)} + \theta_{(3,k)}x_{(n,3)} = \sum_{m=1}^{M} \theta_{(m,k)}x_{(n,m)}$$

and for the whole matrix

$$Z = X\theta.$$

With $M = 2 + 1$ features and five classes, the 15-dimensional weight matrix $\theta$ will be $M \times K = 3 \times 5$ in size. For each data point, we compute the probabilities for the data point of belonging to each of the $K$ classes via the softmax function $\Phi$:

$$\mathcal{P}(y = k|Z) = \Phi(Z) = \frac{e^Z}{\sum_{j=1}^{K} e^{Z_j}}.$$

Thus, the softmax probability matrix $\Phi(X)$ will be $N \times K$ in size.

We calculate the cross-entropy loss function to determine the loss for row

$$\mathcal{F}(Z, Y_i, \theta_i) = -\sum_{m=1}^{K} Y_{i,m} \cdot \Big(\log \Phi(Z) + \log \pi(\theta_{i,m})\Big),$$

where $\log \pi(\theta_{i,m})$ denotes the logarithm of the prior knowledge $\pi$. We then minimize the total loss function $\mathcal{L}$, the mean of losses over all training samples:

$$\mathcal{L}(\theta, X, Y) = \frac{1}{N}\sum_{i=0}^{N} \mathcal{F}(Z, Y_i, \theta),$$

where $N$ is the number of training samples.

### Classification Results

Each variable of the input data is standardized to mean $\mu = 0$ and standard deviation $s = 1/2$, and the training and testing sets are split 80%:20% of the total data. The data is shuffled after every full trajectory and the batches are formed without replacement, although with this data set the sampling method does not affect the accuracy of the model.

The following parameters are set for the SGHMC algorithm: the clipping constant $L = 1.0$, the scalar-valued error models $\mathcal{C} = 0.15$ and $\tilde{\mathcal{B}} = 0$, sampling probability $q = 0.01$, total iterations $T = 1500$. The sequence of step-sizes for SGHMC is set as

$$\eta_t = \beta \cdot t^{-1/3}, \quad t \in T,$$

with the multiplier set $\beta = 1.7$. The amount of integrator steps is set $J = N/B$, which mirrors the Betancourt's implementation of stochastic HMC-algorithm [7] with data sub-sampling during the trajectory computation.

For standard HMC implementation the hyper-parameters are set as $\eta = 3$ and $J = 23$ for 60% acceptance ratio. The two tested prior distributions are set to $\pi_1 \sim \mathcal{N}(0, 10)$ and $\pi_2 \sim \mathcal{U}(-1000, 1000)$. For the predictions we discard the first 75% of the samples as burn-in and use the means of the posteriors from the remaining samples as parameters for logistic regression. The mean accuracy and standard deviation are calculated from 15 simulations and the results are shown in the table (6.2).

Furthermore, we also test the SGHMC-algorithm with only a fraction of the available data by setting the trajectory length $J = 10$, as having $J = (N/B)$ does not give us any computational gains over the standard HMC [7]. The batch size stays the same, so we only use 10 batches worth of data. With well optimized hyper-parameters, this does not affect the accuracy of the model noticeably. We notice that the sampling probability $q = 0.01$ is enough for accurate stochastic gradients – having $q = 0.05$ does not gain us significantly better results, but compromises the differential privacy of the model. With the lower $J$ value, the SGHMC algorithm is $(9.789, 10^{-5})$-DP with the set parameters and is on par with the other algorithms w.r.t the classification accuracy.

SGHMC consistently achieves similar classification accuracy as the standard HMC and SGD algorithms. The naive implementation without the friction term has a higher sample variance and cannot produce dependable results. The classification results from the robust algorithms are close enough together for this data set, that the differences between them could be attributed to random variation. Thus we cannot consider the results statistically significant, however we note that over multiple executions SGHMC had consistently a slightly higher accuracy than the differentially private SGD algorithm while retaining a better privacy profile.

The difference could be explained by Hamiltonian Monte Carlo's better robustness

against overfitting, due to random nature of Markov chains. As the dataset is heavily biased to two of the five possible classes, the better results could possibly be achieved by more thorough optimization of the regression weight matrix. The choice of the prior does not affect the results considerably.

| Method | Accuracy | Sample std. dev. |
|---|---|---|
| **SGD** | 0.731 | 0.001 |
| $(10.727, 10^{-5})$**-DP SGD** | 0.711 | 0.004 |
| **HMC $\mathcal{N}$ target** | 0.731 | 0.004 |
| **HMC $\mathcal{U}$ target** | 0.732 | 0.002 |
| $(9.789, 10^{-5})$**-DP SGHMC, $\mathcal{N}$ target** | 0.730 | 0.003 |
| $(9.789, 10^{-5})$**-DP SGHMC, $\mathcal{U}$ target** | 0.726 | 0.006 |
| **Naive SGHMC $\mathcal{N}$, var. $\eta$** | 0.716 | 0.021 |
| **Naive SGHMC $\mathcal{U}$, var. $\eta$** | 0.722 | 0.013 |
| **Naive SGHMC $\mathcal{N}$, const. $\eta$** | 0.616 | 0.122 |
| **Naive SGHMC $\mathcal{U}$, const. $\eta$** | 0.728 | 0.009 |

**Table 6.2:** Means of classification accuracy and the sample standard deviation for different algorithms after 15 executions. Here $\mathcal{N}$ denotes the normal distribution prior and $\mathcal{U}$ the uniform distribution prior.

## 6.4 Posterior Accuracy of SGHMC

We measure the differentially private 15-dimensional posterior features of the above softmax regression against a long running standard HMC implementation. As it is known, that a long running HMC eventually converges to the exact posterior of the model [8, 24], and when $\eta \to 0$, the SGHMC also converges to the exact target posterior [12, 31]. However, then the algorithm's computational performance will suffer, as we take only minuscule steps during the integration. Thus, we have to balance the posterior accuracy and the minimum step size in our model, with the sequence of $\eta_t = \beta \cdot t^{-1/3}, \quad t \in T$, with the multiplier $\beta = 1$ and $T = 1500$, will be $\eta_{1500} \approx 0.0874$.

We measure the posterior accuracy by calculating the absolute differences between the HMC and SGHMC posterior standard deviations $s$ for each dimension and denote their mean as $|s_{hmc} - s_{sghmc}|$, for conciseness. This is not a watertight measure, as the shape of the posteriors can vary and each execution finds slightly different solutions for the softmax regression [20]. However, we deem it adequate to showcase the achieved posterior accuracy.

The naive implementation fails to accurately approximate the exact posterior,

even with small step sizes, as there is no MH-acceptance step or friction to restrain the sample deviation from the mean/median. Thus we exclude it from the comparison. We compare the standard deviations of each dimension of the SGHMC algorithm against those from standard HMC algorithm run for 5000 iterations (Fig. (6.6)). As each implementation converges to slightly different solution for the regression parameters $\theta$, there is little point comparing those values. With the softmax regression the difference between parameters is not a problem, as due the parameter redundancy there are multiple values that fit the given data [20].

Our tests show that regardless of parametrisation the classification results from SGHMC are comparable to those of SGD or HMC methods (Fig. (6.7)): when the hyper-parameters are approximately in the area which yields the correct results, the classification accuracy is more often than not on par with the baseline SGD algorithm. Similarly the difference in the mean posterior accuracy stays low when a large enough $\beta$ is used.

Chen et al. recommend keeping the scalar-valued noise model $\mathcal{C}$ rather low at 0.01 or less [12]. However we notice that with a large enough step size multiplier the algorithm yields correct results even with higher amounts of injected noise. The results stay acceptable until $\mathcal{C} = 0.66$, when at high step size multipliers both accuracy meters drop catastrophically. However lower multipliers $\beta$ are still usable.

The SGHMC algorithm converges to a solution in under $T = 100$ iterations. For purely optimization tasks, the algorithm is thus able to compete against SGD in speed and accuracy. The measured posterior accuracy is relatively good, however a good approximation of the posterior naturally needs a larger sample size.

## 6.5 Differential Privacy

We compute the moments accountant and the tight privacy profile for the classification task above. For MA calculation, we use analytical Rényi differential privacy (RDP) by Wang et. al [30], who has provided the code for calculations*. The analytical RDP would give more accurate results than previous implementations of moments accountant (where applicable). The tight bounds for the privacy parameters are calculated via PLD-method by Koskela et al. [19], who have also provided the code for calculations†.

*Code available at https://github.com/yuxiangw/autodp

†Code available at https://github.com/DPBayes/PLD-Accountant/

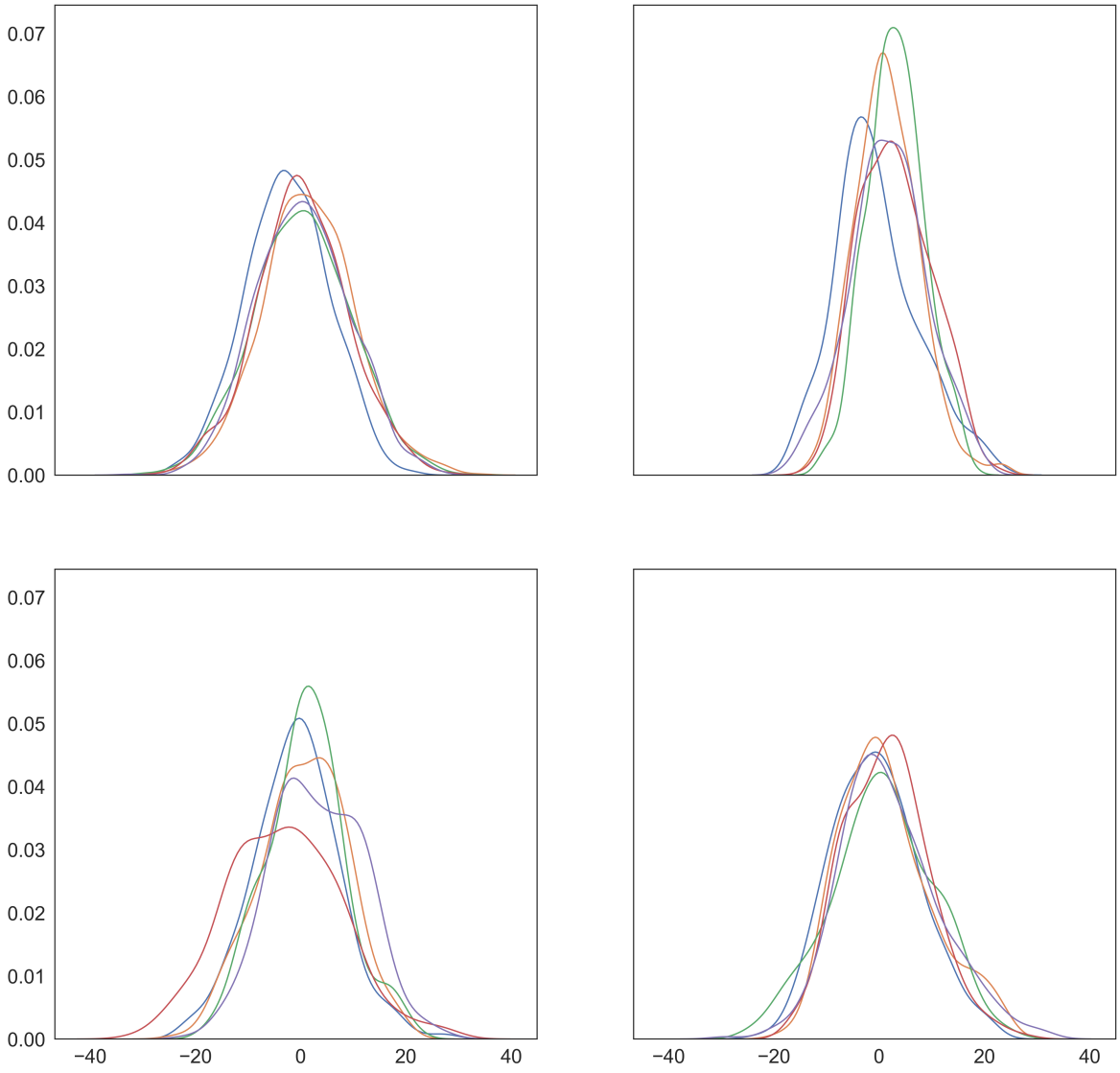


**Figure 6.6:** Five of the total 15 dimensions plotted for HMC and SGHMC. The posterior accuracy is measured by the average absolute distance from the posterior sampled by the standard HMC algorithm $|s_{HMC} - s_{algorihm}|$. **Top left**: the posteriors of a standard HMC algorithm run for 5000 iterations. **Top right**: SGHMC with the error model $\mathcal{C} = 0.05$: $|s_{hmc} - s_{sghmc}| \approx 1.68$. **Bottom left**: SGHMC with the error model $\mathcal{C} = 0.03$: $|s_{hmc} - s_{sghmc}| \approx 0.87$. With the smaller error, at the expense of the privacy parameter $\epsilon$, we gain improvement in the posterior accuracy. **Bottom right**: SGHMC with no regard for differential privacy: the model is optimised to approximate the actual posterior: $|s_{hmc} - s_{sghmc}| \approx 0.65$

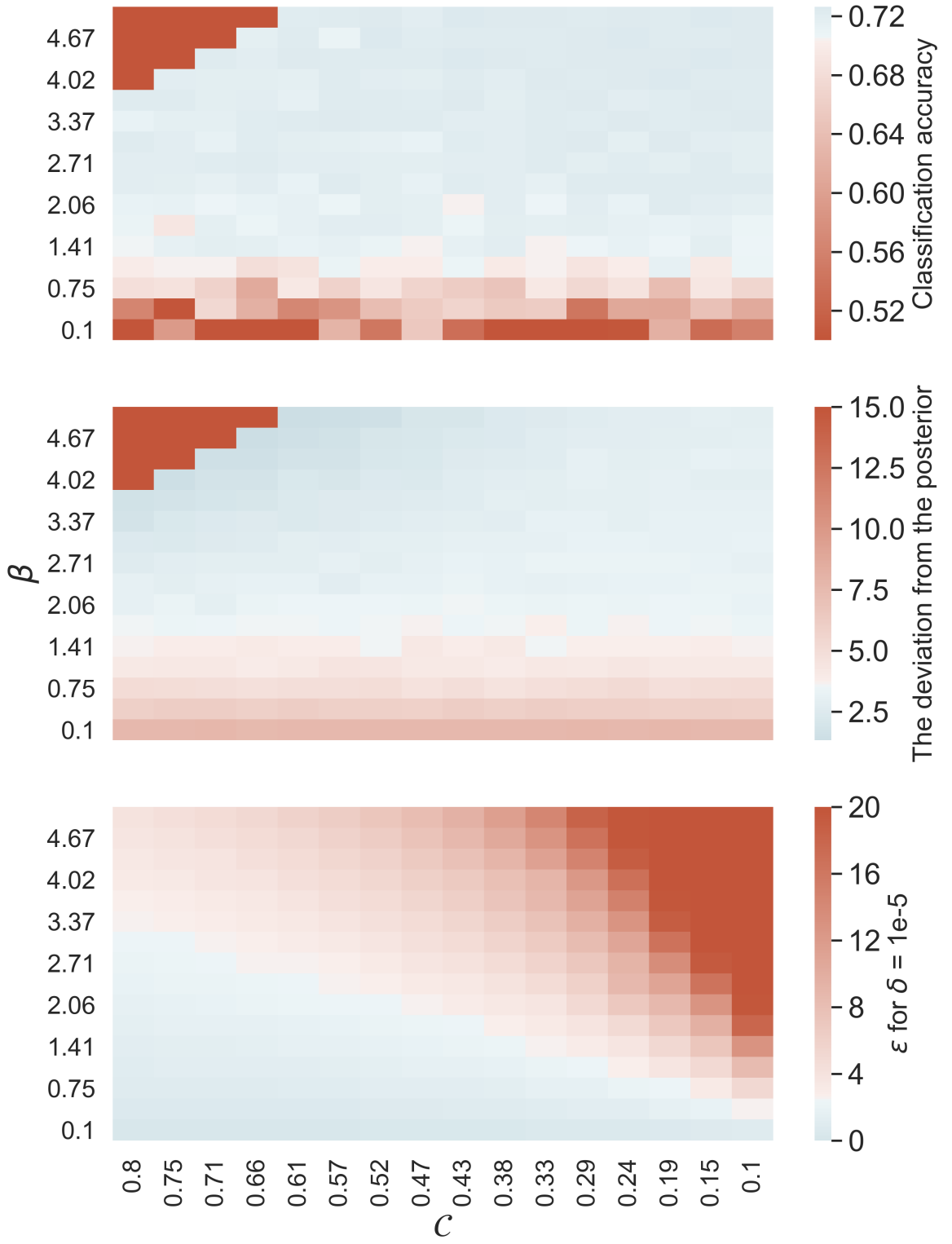


**Figure 6.7:** Classifcation performance, deviation from the target posterior and $(\epsilon, 10^{-5})$-DP privacy profile for SGHMC algorithm at different step size multipliers $\beta$ and error models $\mathcal{C}$ after $T = 750$ iterations with $J = 10$ integration steps. Here the clipping constant is set to $L = 1$. The $\epsilon$ keeps improving with the rising error profile, and the results stay the same until $\mathcal{C} > 0.66$, when the injected noise starts to interfere with both classification and posterior accuracy. The posterior and classification accuracy are mean results from 10 executions.

### Bounded and Unbounded Differential Privacy

Here we compute the unbounded privacy profile $(\epsilon, \delta)$ [19] i.e., $X \simeq_R Y$ and the mini-batches are formed by Poisson subsampling [34]. $X \simeq_R Y$ denotes that one data point has been added or removed from either of the sets. Poisson subsampling means that each data point is individually and independently collected on the mini-batch via binomial sampling $\gamma \sim \text{Binom}(q)$: $B = \{x_i|\gamma = 1, i \in N\}$. Although to keep consistency with previous results [1, 12, 30], we likewise approximate the Poisson subsampling by sampling $q \cdot |N|$ sized mini-batches without replacement.

The unbounded [19] privacy profile is used by both PLD and MA methods of calculating differential privacy. We can reduce the unbounded profile into bounded profile: for unbounded $(\epsilon, \delta)$ we have bounded profile $(2\epsilon, (1 + e^{\epsilon})\delta)$. When $\epsilon$ is small, the bounded profile can be approximated as $(2\epsilon, 2\delta)$ [19, 30]. The bounded privacy profile is determined by the relation $X \simeq_S Y$ i.e. $|X| = |Y|$ and the data sets differ by one modified data point.

### PLD Versus Moments Accountant

We obtain the sequence of sigmas

$$\sigma = \sqrt{\frac{2(\mathcal{C} - \tilde{\mathcal{B}})}{\eta L^2}}$$

for computing the privacy bounds (Sec. (5.4)). The following parameters are set for the SGHMC algorithm: the clipping constant $L = 0.7$, the scalar-valued error models $\mathcal{C} = 1.0$ and $\tilde{\mathcal{B}} = 0$, sampling probability $q = 0.01$, total iterations $T = 200$. The sequence of step-sizes for SGHMC is set as

$$\eta_t = \beta \cdot t^{-1/3}, \quad t \in T,$$

with the multiplier set $\beta = 3$. Here the integrator steps are set $J = 10$. Thus the total amount of iterations to consider w.r.t the privacy bounds is $T \cdot J$. The results are shown in Fig. (6.8) and in table (6.3). To keep consistent with the previous results [1, 34] we set these parameters to have $\sigma \geq 1$.

Overall the PLD method shows a considerable improvement over the privacy profile, especially with higher $\delta$ values. When tested against the amount of iterations, with constant $\delta = 10^{-5}$, the difference between the methods increases linearly with the amount of iterations.

### Differential Privacy in Hamiltonian Monte Carlo

The problems arise when trying to retain meaningful differential privacy along with accurate test results. We show that SGHMC is able to obtain both accurate classifi-

**(a)** $\epsilon$ against variable $\delta$

| $\delta$ | $\epsilon$-PLD | $\epsilon$-MA |
|---|---|---|
| $10^{-6}$ | 0.881 | 2.254 |
| $10^{-5}$ | 0.763 | 2.061 |
| $10^{-4}$ | 0.629 | 1.848 |
| 0.001 | 0.473 | 1.607 |
| 0.01 | 0.273 | 1.324 |

**(b)** $\epsilon$ against iteration time with constant $\delta$

| # of iterations $J \cdot T$ | $\epsilon$-PLD | $\epsilon$-MA |
|---|---|---|
| 1000 | 0.609 | 1.664 |
| 2000 | 0.763 | 2.061 |
| 5000 | 1.040 | 2.800 |
| 10000 | 1.324 | 3.541 |

**Table 6.3:** **(a)** Privacy parameter $\epsilon$ computed by given parameter $\delta$ for logistic regression after $T \cdot J = 2000$ iterations. The PLD profile is considerably smaller than the results from MA. **(b)** The effect of total iteration time to the privacy bounds. The values $\epsilon$ were computed with constant $\delta = 10^{-5}$.

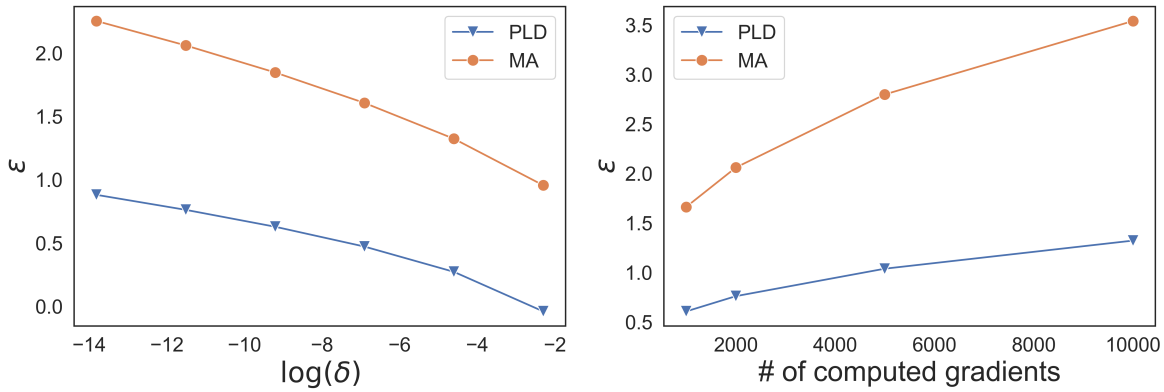


**Figure 6.8: Left:** Calculated parameters $\epsilon$ w.r.t. the parameter $\delta$ after $T \cdot J = 2000$ total iterations. $\log(\delta)$ is used for plotting for improved readability. **Right:** Calculated parameters $\epsilon$ and constant $\delta = 10^{-5}$ w.r.t. number of iterations.

cation results and closely approximated posterior with many different combination of hyper-parameters (Sec. (6.4)). At recommended amount of injected noise [12] we are only able to achieve loose privacy bounds, but the model nevertheless stays accurate until very high values $\mathcal{C}$ is used. Thus we can inject high amounts of noise into the model without compromising the accuracy.

Fig. (6.7) shows the model accuracy and DP-bounds interaction. For comparison, we run two executions with comparable classification and posterior accuracy to long-running HMC (Fig. (6.9)). Both run for 1500 iterations with trajectory length $J = 10$ and sampling probability $q = 0.01$. The algorithm with set parameters $\beta = 2$, $\mathcal{C} = 0.05$ and $L = 0.7$ is $(19.811, 10^{-5})$-DP. With parameters $\beta = 3$, $\mathcal{C} = 0.7$ and $L = 0.8$ the algorithm is $(2.248, 10^{-5})$-DP. They perform similarly, but have vastly different privacy profiles.

Moments accountant method might not give us reliable privacy bounds in the typical use-case of SGHMC algorithm: the hyper-parameters that give good results are

usually set such that $\sigma < 1$ for at least at the start of the sequence. For MA method it is required that $\sigma \geq 1$ [1] and it is implicated that Rényi accountant performs better with this constraint as well [34, 30]. Additionally, MA converges to better privacy bounds when the number of iterations is high [1]. While a good approximation of the posterior demands a large sample size, we see that SGHMC is able to converge to good optimization solution with a low amount of iterations, and thus MA might not give good privacy bounds for $\epsilon$ and $\delta$ in such use cases.

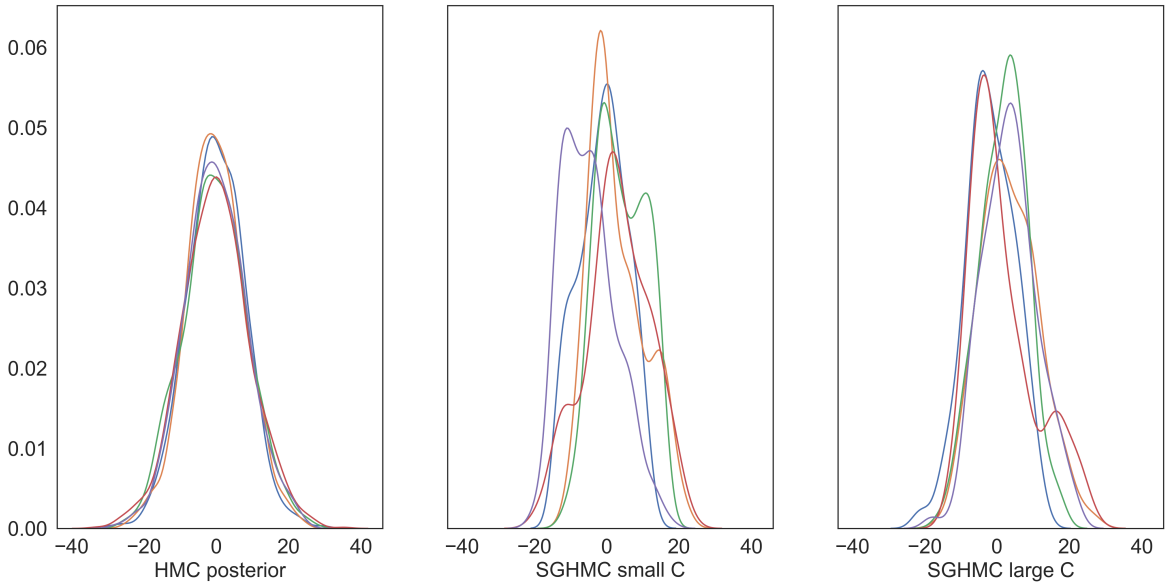


**Figure 6.9:** Comparison between posterior distributions with similar classification accuracy. **Left**: a group of distributions from the posterior of a standard HMC algorithm run for 5000 iterations. **Center**: the same distributions from SGHMC with $(19.811, 10^{-5})$-DP. $|s_{hmc} - s_{sghmc}| \approx 1.67$. **Right:** SGHMC with $(2.248, 10^{-5})$-DP, using high $\mathcal{C} = 0.7$, achieving $|\sigma_{hmc} - \sigma_{sghmc}| \approx 1.57$. Even though the deviation between the posteriors is relatively small, by visual inspection one of the dimensions has a different shape than it's actual counterpart.

# 7. Conclusions

While powerful and versatile tool, Hamiltonian Monte Carlo struggles with huge datasets due to its reliance on gradient evaluations over the whole data set. Stochastic gradient Hamiltonian Monte Carlo improves the computational efficiency by using only fractions of the original dataset to approximate these gradients.

Due to biases caused by integrating over the stochastic gradients, the preservation of the system's energy does not hold in theory. However, this bears little significance in our experiments: a well-adjusted friction term constrains the uncontrolled entropy increase in the algorithm and the posterior accuracy is preserved in the 15-dimensional test setting. However, in even higher dimensions the sub-sampling bias might increase and skew the results, as the friction term cannot conserve the volume of the energy exactly: it merely stops the samples straying far off from the actual energy state.

The lack of the Metropolis acceptance steps in SGHMC causes the stray samples to skew the variance in the approximated target set. Typically in optimization tasks one is only interested in the mode/mean of the approximate posterior. For example, the confidence intervals from approximate posteriors might not resemble those of the actual target set. With larger batch sizes, e.g. $B = N/2$, we get close approximations of the posterior. The data sizes for the gradient evaluations are cut in half, which results in improved computational efficiency while still approximately preserving the energy equilibrium.

We are able to achieve relatively tight differential privacy bounds without sacrificing almost any classification accuracy of the SGHMC algorithm. The increased stochastic noise of the custom error model represented by the additional friction term could actually help the classification task with difficult data sets by lowering the chance of overfitting to the training data. As many different parameters affect the privacy bounds, we optimize the algorithm for better results even inside set privacy restrictions.

On the other hand, the multitude of hyper-parameters makes the optimization of the algorithm also more difficult, and for algorithms such as No-U-Turn-Sampler (NUTS) [18] it is harder to balance for the efficiency and privacy of the algorithm. There is no acceptance step implemented in the stochastic algorithms: it would require

evaluating the probabilities over the whole data set and we would lose the computational gains. Instead we rely on the decreasing step size to control the deviance from the target posterior. Thus, nothing restricts the simulated posterior distributions from deviating from the actual target posterior. Therefore a careful adjusting of the different parameters is needed for achieving a good approximation of the exact target posterior. In the experiments we illustrate the effect of choosing appropriate hyper-parameters. In the thesis we omit the considerations for the privacy cost of the hyper-parameter itself.

To conclude, we cannot consider SGHMC as a reliable algorithm alone, because the attributes of the target posterior would have to be known (for example by sampling it beforehand using the standard HMC algorithm) for adjusting the model to it. We have confirmed Michael Betancourt's findings [7] that without using the exact gradients the posterior accuracy will suffer. Although adding the friction term to SGHMC algorithm is able to alleviate some amount of the injected bias, it is not a simple fix and have to be configured precisely to the model's needs. Thus we should consider SGHMC merely an imitation of the target posterior and take into consideration the possible deviation from the actual posterior distribution.

We also notice that the new privacy loss distribution technique to compute the tight privacy profiles is able to achieve tighter bounds for $\epsilon$ and $\delta$ than the moments accountant or the Rényi accountant methods. We are able to optimize the algorithms to higher precision w.r.t. achieved privacy, as a better privacy bound leads to less excessive noise injected the model. SGHMC with the added friction term proves difficult with regards to the differential privacy, as the parameters that allow us effectively approximate the target posterior also affect the achieved privacy profile.

SGHMC's strength lies in the optimization tasks: it is shown that SGHMC algorithm converges faster [12] and the results are on par with other, e.g. DP-SGD [21] algorithms.

# Bibliography


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

[2] S. Ahn, A. Korattikara, and M. Welling. Bayesian posterior sampling via stochastic gradient Fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

[3] B. J. Alder and T. E. Wainwright. Studies in molecular dynamics. I. General method. *The Journal of Chemical Physics*, 31(2):459–466, 1959.

[4] Apple Inc. Differential privacy overview. https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.

[5] V. I. Arnol'd. *Mathematical methods of classical mechanics*, volume 60. Springer Science & Business Media, 2013.

[6] H. Baker. Alternants and continuous groups. *Proceedings of the London Mathematical Society*, 2(1):24–47, 1905.

[7] M. Betancourt. The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling. In *International Conference on Machine Learning*, pages 533–540, 2015.

[8] M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[9] M. Betancourt, S. Byrne, S. Livingstone, M. Girolami, et al. The geometric foundations of Hamiltonian Monte Carlo. *Bernoulli*, 23(4A):2257–2298, 2017.

[10] S. Brooks. Markov chain Monte Carlo method and its application. *Journal of the royal statistical society: series D (the Statistician)*, 47(1):69–100, 1998.

[11] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of markov chain Monte Carlo*. CRC press, 2011.

[12] T. Chen, E. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International conference on machine learning*, pages 1683–1691, 2014.

[13] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

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

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

[16] M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[17] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[18] M. D. Hoffman and A. Gelman. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[19] A. Koskela, J. Jälkö, and A. Honkela. Computing exact guarantees for differential privacy. *arXiv preprint arXiv:1906.03049*, 2019.

[20] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[21] B. Li, C. Chen, H. Liu, and L. Carin. On connecting stochastic gradient mcmc and differential privacy. *arXiv preprint arXiv:1712.09097*, 2017.

[22] M. Magnusson. https://github.com/MansMeg/posteriordb.

[23] A. Narayanan and V. Shmatikov. How to break anonymity of the Netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[24] R. M. Neal et al. Mcmc using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[25] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[26] G. O. Roberts, A. Gelman, W. R. Gilks, et al. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[27] A. Rogozhnikov. Hamiltonian Monte Carlo explained. http://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html.

[28] W. Rossmann. *Lie groups: an introduction through linear groups*, volume 5. OUP Oxford, 2002.

[29] O. Thakkar, G. Andrew, and H. B. McMahan. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[30] Y.-X. Wang, B. Balle, and S. Kasiviswanathan. Subsampled Rényi differential privacy and analytical moments accountant. *arXiv preprint arXiv:1808.00087*, 2018.

[31] Y.-X. Wang, S. Fienberg, and A. Smola. Privacy for free: Posterior sampling and stochastic gradient Monte Carlo. In *International Conference on Machine Learning*, pages 2493–2502, 2015.

[32] Z. Wang, S. Mohamed, and N. Freitas. Adaptive Hamiltonian and Riemann manifold Monte Carlo. In *International Conference on Machine Learning*, pages 1462–1470, 2013.

[33] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[34] Y. Zhu and Y.-X. Wang. Poisson subsampled Rényi differential privacy. In *International Conference on Machine Learning*, pages 7634–7642, 2019.

# Appendix A. Pseudo Codes

**Algorithm 1** Hamiltonian Monte Carlo

**Input:**

Fixed step-size $\eta$ and the initial starting point $q_1$.

**for** $t = 1 : T$ **do**
  Sample momentum $p_t \sim \mathcal{N}(0, M)$
  $H_0 \leftarrow (q_t, p_t)$
  Calculate $p_0(q_0) \leftarrow \frac{\eta}{2}\nabla U(q_0)$.
  **for** $i = 1 : J$ **do**
    $q_i \leftarrow q_{i-1} + \eta M^{-1} p_{i-1}$
    $p_i \leftarrow p_{i-1} - \frac{\eta}{2}\nabla U(q_i)$
  **end for**
  $p_J \leftarrow p_J - \frac{\eta}{2}\nabla U(q_J)$
  $H \leftarrow (q_J, p_J)$
  $u \sim \text{Uniform}(0, 1)$
  $\rho = e^{H - H_0}$
  **if** $u < \min(1, \rho)$ **then**
    $q_{t+1} \leftarrow q_J$
  **end if**
**end for**

**Output:**

Return all collected samples.

**Algorithm 2** Differentially Private Stochastic Gradient Hamiltonian Monte Carlo

**Input:**

Sequence step-sizes $\eta$, $\eta_t \in \eta$, the initial starting point $q_1$, Error models $\mathcal{C}$ and $\tilde{\mathcal{B}}$, and the clipping norm $L$. The parameters must satisfy $\frac{2(\mathcal{C}-\tilde{\mathcal{B}})}{\eta_t^2} \succ \frac{1}{N} I_d$.

**for** $t = 1 : T$ **do**
  Sample momentum $p_t \sim \mathcal{N}(0, M)$
  $H_0 \leftarrow (q_t, p_t)$
  **for** $i = 1 : J$ **do**
    $q_i \leftarrow q_{i-1} + \eta M^{-1} p_{i-1}$
    $\nabla \tilde{U}(q_i) \leftarrow \nabla \tilde{U}(q_i) / \max\left(1, \frac{||\nabla \tilde{U}(q_i)||_2}{L}\right)$ (Clip norm)
    $p_i \leftarrow p_{i-1} - \eta_t \left(\nabla \tilde{U}(q_i) - \mathcal{C} M^{-1} p_{i-1} + \mathcal{N}(0, 2(\mathcal{C} - \tilde{\mathcal{B}})\right)$
  **end for**
  $H \leftarrow (q_J, p_J)$
  $q_{t+1} \leftarrow q_J$
  No MH acceptance step
**end for**

**Output:**

Return all collected samples.