Stochastic Fractional Hamiltonian Monte Carlo

Nanyang Ye[1], **Zhanxing Zhu**[*2,3]
[1] University of Cambridge, Cambridge, United Kingdom
[2] Center for Data Science, Peking University, Beijing, China
[3] Beijing Institute of Big Data Research (BIBDR)
yn272@cam.ac.uk, zhanxing.zhu@pku.edu.cn

Abstract

In this paper, we propose a novel stochastic fractional Hamiltonian Monte Carlo approach which generalizes the Hamiltonian Monte Carlo method within the framework of fractional calculus and Lévy diffusion. Due to the large "jumps" introduced by Lévy noise and momentum term, the proposed dynamics is capable of exploring the parameter space more efficiently and effectively. We have shown that the fractional Hamiltonian Monte Carlo could sample the multi-modal and high-dimensional target distribution more efficiently than the existing methods driven by Brownian diffusion. We further extend our method for optimizing deep neural networks. The experimental results show that the proposed stochastic fractional Hamiltonian Monte Carlo for training deep neural networks could converge faster than other popular optimization schemes and generalize better.

1 Introduction

Sampling and minimizing error functions over continuous and high-dimensional spaces have been a primary challenge in current machine learning research, mainly attributed to the exponentially increasing local minima and saddle points of modern models [Dauphin *et al.*, 2014; Goodfellow *et al.*, 2016].

Markov Chain Monte Carlo (MCMC) methods have attracted a lot of attentions recently due to their sucesses in high-dimensional Bayesian inference [Max and Whye, 2011; Chen *et al.*, 2016; 2014]. In addition, MCMC methods have also been incorporated in optimization algorithms for better exploring the landscapes of the loss function[Chen *et al.*, 2016; Chaudhari *et al.*, 2016]. These methods are based on constructing stochastic differential equations (SDE) equipped with Brownian motion, assuming that the particle is driven by infinite number of small forces with finite variance. However, the finite variance assumption essentially restricts the trajectory of the particle to be continuous. This means that when the time step is small, the exploration over the parameter space could be quite slow. Additionally, to facilitate the Brownian motion based SDE to sample correctly the target posterior, small learning rates and large numbers of sampling steps are typically required in practice to ensure the correct stationary distribution [Ahn *et al.*, 2012]. All these existing issues lead to slow mixing during the sampling procedure.

One alternative to tackle these issues is to relax the assumption of finite variance in Brownian motion and generalize the Brownian motion based SDE to the one driven by Lévy diffusion. With Lévy diffusion, the moving of particle in the high-dimensional space does not have to be continuous, also referred to as "jumps". The jumping property of Lévy process has been proven to be more appropriate for modeling fast-changing stochastic processes, where lots of applications involve, such as finance market modeling [Barndorff-Nielsen and Shephard, 2003] and signal processing [Wolpert and Taqqu, 2005]. [Simsekli, 2017] investigated relaxing the finite variance assumption in stochastic gradient first-order Langevin dynamics, and showed that the Lévy diffusion based fractional Langevin dynamics (FLD) could sample the parameter space more efficiently than the first-order Langevin dynamics (LD). However, when the variance of Lévy noise is large, the numerical computation of FLD is unstable especially when applied to deep neural networks. Besides, this first order extension of LD could not sample the parameter space efficiently. This motivates us to marry the Lévy diffusion with HMC to facilitate more efficient sampling of parameter space.

To facilitate more efficient and effective exploration of parameter space, we marry the Lévy diffusion based SDE and Hamiltonian Monte Carlo for sampling and optimization, "Fractional Hamiltonian Monte Carlo", abbreviated as **FHMC**. This new proposal introduces a friction term (also called momentum) to handle the numerical instability of Lévy diffusion while still maintaining the nice property of efficient exploration of Lévy jumps.

For sampling, we compare our methods with first-order LD, FLD and HMC to demonstrate its effectiveness in sampling multi-modal distributions. For training deep neural networks, our training process is divided into two phases: Bayesian sampling to explore the parameter space efficiently and get to the "fat mode" for better generalization ability; optimization for fine-tuning. With the support of extensive evidence, we demonstrated the efficiency and effectiveness of our proposed algorithm for sampling and training deep neural networks. To the best of our knowledge, this is the first attempt that

*Corresponding author.

adopts Lévy diffusion and HMC into training modern deep networks and produces remarkable improvements over the state-of-the-art techniques.

2 Preliminaries

In the scenario of Bayesian learning, obtaining the samples of a high-dimensional distribution is a necessary procedure for many tasks. We denote $U(\boldsymbol{\theta})$ the potential energy function, i.e., the negative log posterior,

$$U(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \log p(\mathbf{x}_i|\boldsymbol{\theta}) - \log p_0(\boldsymbol{\theta}),$$

where $\mathbf{x}_i$ represents the i-th observed data point, $p_0(\boldsymbol{\theta})$ is the prior distribution for the model parameters and $p(\mathbf{x}_i|\boldsymbol{\theta})$ is the likelihood term for each observation. In optimization scenario, the counterpart of the complete negative log likelihood is the loss function and $-\log p_0(\boldsymbol{\theta})$ is typically referred to as a regularization term.

Classic dynamics offers such a way to sample the distribution. The Hamiltonian in classic dynamics is $H(\mathbf{z}) = H(\boldsymbol{\theta}, \mathbf{r}) = U(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{r}^T\mathbf{r}$, the sum of the potential energy $U(\boldsymbol{\theta})$ and kinetic energy $\frac{1}{2}\mathbf{r}^T\mathbf{r}$, where $\mathbf{r} \in \mathbb{R}^d$ is the momentum term, and $\mathbf{z} = (\boldsymbol{\theta}, \mathbf{r})$.

Standard (second-order) Langevin dynamics driven by Gaussian noise [1] can be described by following stochastic differential equations (SDEs),

$$\mathrm{d}\boldsymbol{\theta} = \mathbf{r}\mathrm{d}t, \quad \mathrm{d}\mathbf{r} = -\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\mathrm{d}t - \gamma\mathbf{r}\mathrm{d}t + \sqrt{2\gamma}\mathrm{d}\mathbf{W} \quad (1)$$

where $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ is the gradient of the potential energy w.r.t. the configuration states $\boldsymbol{\theta}$, γ denotes the friction coefficient, and $\mathrm{d}\mathbf{W}$ is the standard Wiener process.

If we simulate the dynamics in Eqs (1), a well-known stationary distribution can be achieved [Rapaport, 2004], $p(\mathbf{z}) = \exp\left(-H(\mathbf{z})\right)/Z$, where $Z = \int\int \exp\left(-\alpha H(\boldsymbol{\theta}, \mathbf{r})\right) \mathrm{d}\boldsymbol{\theta}\mathrm{d}\mathbf{r}$ is the normalization constant for the probability density. The desired probability distribution associated with the parameters $\boldsymbol{\theta}$ can be obtained by marginalizing the joint distribution, $p(\boldsymbol{\theta}) = \int p(\mathbf{z})\mathrm{d}\mathbf{r} \propto \exp\left(-U(\boldsymbol{\theta})\right)$.

However, the Gaussian proposal distribution induced by the dynamics (1) is light-tailed, which might not provide sufficiently large "jumps" to explore full parameter space efficiently. In this work, we consider a heavy-tailed proposal distribution driven the Lévy stable noise, and investigate its benefits for fast mixing in MCMC.

Lévy stable distributions do not have closed-form probability density function (PDF). Instead, it could be represented by its characteristic function as: $E[exp(iwZ)] = exp(-|\sigma\omega|^{\alpha})$. In order for the mean of the process to exist, the range of α is restricted to be $(1, 2]$. The Gaussian distribution is a special case for Lévy stable distributions when α is 2 and the Lévy stable distribution becomes more heavy tailed when α is closer to 1.

[1] Standard Langevin dynamics is different from that used in SGLD [Max and Whye, 2011], which is the first-order Langevin dynamics, i.e., Brownian dynamics.

3 Fractional Lévy Dynamics for MCMC

We propose a general form of Lévy dynamics as follows:

$$\mathrm{d}\mathbf{z} = (\mathbf{D} + \mathbf{Q})\,\mathbf{b}(\mathbf{z}, \alpha)\mathrm{d}t + \mathbf{D}^{1/\alpha}\mathrm{d}\mathbf{L}^{\alpha}, \quad (2)$$

where $\mathrm{d}\mathbf{L}^{\alpha}$ represents the Lévy stable process, and the drift force has the following form,

$$\mathbf{b}(\mathbf{z}, \alpha) = \frac{\mathcal{D}^{\alpha-2}\{f_p(\mathbf{z})\}}{\phi(\mathbf{z})} \quad (3)$$

$$f_p(\mathbf{z}) = -\phi(\mathbf{z})\frac{\partial H(\mathbf{z})}{\partial \mathbf{z}}, \quad (4)$$

where the unnormalized target probability density function $\phi(\mathbf{z}) = \exp\left(-H(\mathbf{z})\right)$. The matrix $\mathbf{Q}$ is a skew-symmetric curl matrix representing the deterministic traversing effects in the proposed Lévy dynamics. In contrast, the diffusion matrix $\mathbf{D}$ determines the strength of the Lévy process-driven diffusion. Matrices $\mathbf{D}$ and $\mathbf{Q}$ will be specified later to attain fast convergence to the target distribution.

The stationary distribution of the general Lévy dynamics in (2) can be characterized by the following theorem.

Theorem 1. *$p(\mathbf{z}) \propto \exp\left(-H(\mathbf{z})\right)$ is a stationary distribution of the dynamics of Eq. (2) if the matrix $\mathbf{D}$ positive semidefinite and $\mathbf{Q}$ skew-symmetric.*

Proof. We consider the Fokker-Planck equation of the Lévy dynamics described by Eq. (2)

$$\partial_t \rho(\mathbf{z}, t) = -\sum_i \frac{\partial}{\partial z_i}\left[\sum_j (D_{ij} + Q_{ij}) b_j(\mathbf{z}, \alpha)\rho(\mathbf{z}, t)\right] + \mathcal{D}^{\alpha-2}\left\{\sum_{ij} \frac{\partial}{\partial z_i z_j} D_{ij}\rho(\mathbf{z}, t)\right\} \quad (5)$$

Denote the two terms A and B, respectively, and insert $\mathbf{b}(\mathbf{z}, \alpha)$ into A,

$$A = \sum_i \frac{\partial}{\partial z_i}\left[\sum_j (D_{ij} + Q_{ij}) \frac{\mathcal{D}^{\alpha-2}\{\phi(\mathbf{z})\partial H(\mathbf{z})/\partial z_j\}}{\phi(\mathbf{z})}\rho(\mathbf{z}, t)\right] \quad (6)$$

$$= \sum_i \frac{\partial}{\partial z_i}\left[\sum_j (D_{ij} + Q_{ij}) \frac{\mathcal{D}^{\alpha-2}\{p(\mathbf{z})\partial H(\mathbf{z})/\partial z_j\}}{p(\mathbf{z})}\rho(\mathbf{z}, t)\right] \quad (7)$$

$$= -\sum_i \frac{\partial}{\partial z_i}\left[\sum_j (D_{ij} + Q_{ij}) \frac{\mathcal{D}^{\alpha-2}\{\partial p(\mathbf{z})/\partial z_j\}}{p(\mathbf{z})}\rho(\mathbf{z}, t)\right] \quad (8)$$

Now we verify whether $\partial_t \rho(\mathbf{z}, t)$ will vanish given that

$\rho(\mathbf{z}, t) = p(\mathbf{z})$,

$$A = -\sum_i \frac{\partial}{\partial z_i}\left[\sum_j (D_{ij} + Q_{ij})\mathcal{D}^{\alpha-2}\left\{\frac{\partial p(\mathbf{z})}{\partial z_j}\right\}\right] \quad (9)$$

$$= -\sum_{ij} (D_{ij} + Q_{ij})\mathcal{D}^{\alpha-2}\left\{\frac{\partial p(\mathbf{z})}{\partial z_i \partial z_j}\right\} \quad (10)$$

$$= -\mathcal{D}^{\alpha-2}\left\{\sum_{ij} \frac{\partial}{\partial z_i z_j} D_{ij} p(\mathbf{z})\right\}, \quad (11)$$

where the last equality holds because $\sum_{ij} \frac{\partial}{\partial z_i \partial z_j}[Q_{ij} p(\mathbf{z})] = 0$ due to anti-symmetry of $\mathbf{Q}$. Therefore, when $\rho(\mathbf{z}, t) = p(\mathbf{z}) \propto \exp(-H(\mathbf{z}))$, the Fokker-Planck equation $\partial_t \rho(\mathbf{z}, t) = A + B = 0$. □

3.1 Fractional Hamiltonian Monte Carlo

We now specify a simple and rather effective configuration of matrix $\mathbf{D}$ and $\mathbf{Q}$ to facilitate efficient sampling and optimization, $\mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & \gamma\mathbf{I} \end{bmatrix}$, $\mathbf{G} = \begin{bmatrix} 0 & -\mathbf{I} \\ \mathbf{I} & 0 \end{bmatrix}$, where γ is the friction coefficient with the role in standard Langevin dynamics (1). With this configuration, the general Lévy dynamics can be instantiated as following,

$$\mathrm{d}\boldsymbol{\theta} = \frac{\mathcal{D}^{\alpha-2}\{\phi(\mathbf{z})\mathbf{r}\}}{\phi(\mathbf{z})}\mathrm{d}t \quad (12)$$

$$\mathrm{d}\mathbf{r} = -\frac{\mathcal{D}^{\alpha-2}\{\phi(\mathbf{z})\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\}}{\phi(\mathbf{z})}\mathrm{d}t - \gamma\frac{\mathcal{D}^{\alpha-2}\{\phi(\mathbf{z})\mathbf{r}\}}{\phi(\mathbf{z})}\mathrm{d}t + \gamma^{1/\alpha}\mathrm{d}\mathbf{L}^{\alpha} \quad (13)$$

We name the dynamics described by Eq. (12) and (12) as Fractional Hamiltonian Monte Carlo (**FHMC**) due to its similarity with Hamiltonian Monte Carlo.

However, there exists two issues when we implementing FHMC for practical use,

- Though Theorem 1 guarantees the desired stationary distribution of FHMC, the Riesz derivatives cannot be computed exactly in general. We have to resort to some numerical approximation;
- When we consider the "Big Data" settings with large N, evaluating the full gradient term $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ is computationally expensive during the iterative sampling procedures. An efficient strategy is required for handling large-scale dataset.

To facilitate its practical use, we propose to apply approximation schemes to FHMC, as elaborated in the following section.

4 Two Approximations to FHMC

Riesz approximation

We use the similar methods proposed by [Çelik and Duman, 2012; Simsekli, 2017], the α-th order fractional Riesz derivative of a function $g(\mathbf{z})$ can be approximated by the following equation,

$$\mathcal{D}^{\alpha} g(\mathbf{z}) \approx c_{\alpha} g(\mathbf{z}) \quad (14)$$

where $c_{\alpha} = \Gamma(\alpha + 1)/\Gamma(\frac{\alpha}{2} + 1)^2$.

With this approximation, we have the following form of FHMC,

$$\begin{aligned} \mathrm{d}\boldsymbol{\theta} &= c_{\alpha}\mathbf{r}\mathrm{d}t, \\ \mathrm{d}\mathbf{r} &= -c_{\alpha}\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\mathrm{d}t - \gamma\mathbf{r}\mathrm{d}t + (\gamma)^{1/\alpha}\mathrm{d}\mathbf{L}_{\alpha} \end{aligned} \quad (15)$$

Stochastic Gradient Approximation

To allow its applicability to large-scale datasets, we use stochastic approximation to reduces the computational burden dramatically, where a much smaller subset of the data, $\{\mathbf{x}_{k_1}, \ldots, \mathbf{x}_{k_m}\}$, is selected randomly to approximate the full one,

$$\tilde{U}(\boldsymbol{\theta}) = -\frac{N}{m}\sum_{j=1}^{m} \log p(\mathbf{x}_{k_j}|\boldsymbol{\theta}) - \log p_0(\boldsymbol{\theta}). \quad (16)$$

And the resulting stochastic gradient $\nabla\tilde{U}(\boldsymbol{\theta})$ is an unbiased estimation of the true gradient. This sampling method is referred as Stochastic Gradient FHMC (**SGFHMC**).

We summarize the procedure of FHMC for sampling in Algorithm 1.

Algorithm 1 (Stochastic Gradient) Fractional Hamiltonian Monte Carlo

1: **Input:** γ, α, η, and number of sampling steps L.
2: Initialize $\boldsymbol{\theta}_0$, $\mathbf{r}_0$.
3: **for** $t = 1, 2, \ldots, L$ **do**
4: If N is large, randomly sample a minibatch of the dataset with size m to obtain $\tilde{U}(\boldsymbol{\theta}^{(t)})$ to approximate $\nabla U(\boldsymbol{\theta}_t)$, otherwise evaluate the full gradient $\nabla U(\boldsymbol{\theta}_t)$;
5: Sample $\boldsymbol{\epsilon}_t \sim \mathbf{L}_{\alpha}$;
6: Update $\boldsymbol{\theta}$:
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + c_{\alpha}\eta\mathbf{r}_{t-1},$$
7: Update $\mathbf{r}$:
$$\mathbf{r}_t = (1 - \eta\gamma)\mathbf{r}_{t-1} - c_{\alpha}\eta\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta}_{t-1}) + (\eta\gamma)^{1/\alpha}\boldsymbol{\epsilon}_t$$
8: **end for**

5 FHMC for Optimizing Deep Neural Networks

Minimizing non-convex error functions $U(\boldsymbol{\theta})$ over continuous and high-dimensional spaces has been a challenging issue in machine learning. Particularly, optimizing modern deep neural networks exhibits severe obstacles, mainly due to the large number of critical points, including various saddle points and minima [Goodfellow *et al.*, 2016; Dauphin *et al.*, 2014]. Moreover, the objective functions of deep networks contain multiple, nearly equivalent global minima. The key difference between these minima is whether they are "*flat*" or "*sharp*", i.e., lying in "wide" or "stiff valleys". A recent study by [Keskar *et al.*, 2016] showed that flat minima of the energy landscape tend to generalize better due to their robustness to data perturbations, noise in the activations as well as perturbations of the parameters. However, most of existing optimization methods are incapable to efficiently

explore the flat minima, often trapping into sharp ones too early.

We deal with this problem from a Bayesian perspective: the flat minima corresponds to "fat" modes of the induced probability distribution over $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}) \propto \exp(-U(\boldsymbol{\theta}))$. Clearly, the fat modes have much more probability mass than "thin" ones since they are nearly as "tall" as each other, particularly in high-dimensional cases. Based on this simple observation, we propose to implement a Bayesian sampling procedure before the optimization phase. Bayesian learning is capable of exploring the energy landscape more thoroughly. Due to the large probability mass, the sampler tends to capture the areas near the desired "flat" minima. This provides a good starting region for optimization phase to fine-tune the parameters.

When sampling the distribution $p(\boldsymbol{\theta})$, the multi-modality issue demands the samplers to transit between isolated modes efficiently. Fortunately, SGFHMC with Lévy noise can produce large "jumps" in the parameter space, facilitating efficient traversing between different modes. Therefore, we plug SGFHMC sampling as the first phase for training deep neural networks. And the entire procedure is detailed in Algorithm 2.

Algorithm 2 SGFHMC for training neural networks

Input: γ, α, η, number of steps for sampling L_s.
Initialize $\boldsymbol{\theta}^{(0)}$, $\mathbf{r}^{(0)}$, .
for $t = 1, 2, \ldots$ **do**
  Randomly sample a minibatch of the dataset with size m to obtain $\tilde{U}(\boldsymbol{\theta}^{(t)})$;
  if $t < L_s$ **then**
    Sample $\boldsymbol{\epsilon}_t \sim \mathbf{L}_\alpha$;
    $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + c_\alpha \eta \mathbf{r}_{t-1}, \quad \mathbf{r}_t = (1 - \eta\gamma)\mathbf{r}_{t-1} - c_\alpha \eta \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_t) + (\eta\gamma)^{1/\alpha} \boldsymbol{\epsilon}_t$
  else
    $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + c_\alpha \eta \mathbf{r}_{t-1}, \quad \mathbf{r}_t = (1 - \eta\gamma)\mathbf{r}_{t-1} - c_\alpha \eta \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_t)\eta$.
  end if
end for

5.1 Connection to Other Methods

There is a direct relationship between the SGFHMC to other methods. In the sampling phase, when α equals 2, SGFHMC becomes SGHMC since $c_\alpha = 1$. In the optimization phase, SGFHMC is essentially SGD with momentum. To facilitate the use of SGFHMC, we use a re-parameterization scheme by replacing γ with $(1 - \text{momentum})/\eta$. Thus we could tune the momentum from $[0, 1)$ just like tuning SGD with momentum.

6 Experiments

To evaluate the proposed method on both sampling and optimization, we conduct experiments on synthetic examples and mnist classification task. For sampling, we compare our method with FLD, HMC and LD. For training deep neural networks, we compare our method with popular optimization methods-SGD, Adam, RMSprop. The same parameter initialization is used for all methods. In the following experiments, for the synthetic example, we only use the sampling steps, for training neural networks, we fix the number of sampling steps to be 8000 and the temperature to be 10^{-6}.

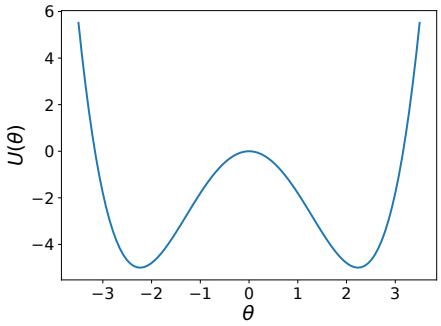

Figure 1: Double well potential function.

6.1 Synthetic Example

Double well potential function is widely used for evaluating inference methods. The target distribution function used is: $U(\theta) = -2\theta^2 + 0.2 * \theta^4$ as shown in Figure 1, similar to the one used in [Chen *et al.*, 2014]. Note that the sampling iteration is set to be 5000 instead of 80000×50 in the original example [Chen *et al.*, 2014] to test the methods' ability to efficiently sample the parameter space in a limited amount of time. Besides, the momentum resampling is neglected which corresponds to the practical algorithm in [Chen *et al.*, 2014]. We set the initialization point of θ at 2 which is close to a local minima and our goal is to estimate the expectation of the distribution function.

In the experiment, we found that when the α is small, the large jump probability of α-stable noise could cause problem in numerical computation for FLD, which makes us hard to choose larger learning rate for FLD. But when small learning rate is chosen for FLD, large α is needed for fast mixing. The setting for each method is the following: FHMC(learning rate is 0.05, momentum is 0.9, α is 1.6), FLD(learning rate is 0.01, α is 1.6), SGHMC(learning rate is 0.1, momentum is 0.1), SGLD(learning rate is 0.05). The corresponding estimation bias achieved by these methods are shown in Table 1.

Method	Bias
FHMC	**0.0360**
FLD	0.6768
HMC	1.9913
LD	0.2661

Table 1: Estimation bias of double well potential

As FHMC and FLD are more general cases for HMC and LD in the framework of α stable noise driven process, we further show the sampling distribution of these two method to know how the extension works in real practice. In Figure 2, the sampling distribution for FLD is not accurate because there are too many samples concentrated close to the initialization point $\theta = -2$. We could conclude that with the introduction of momentum, the sampling of parameter space becomes more

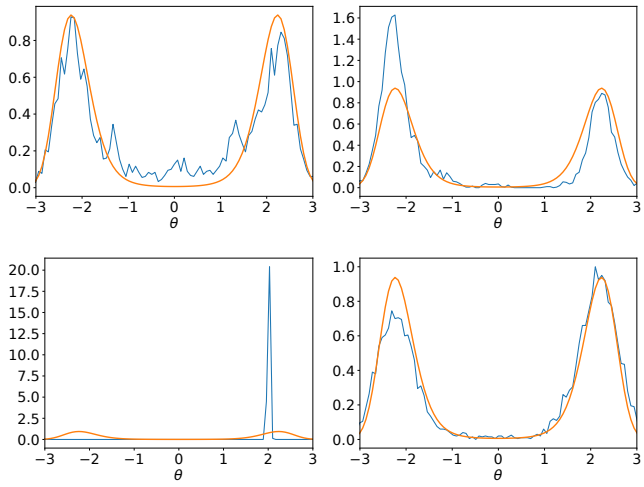

Figure 2: (TopLeft) FHMC result; (TopRight) FLD result; (DownLeft) HMC result; (DownRight) LD result

gentle thus enable the adoption of small α for more efficient sampling. In this task, the mixing speed is crucial for successfully sampling across the parameter space, the HMC could get easily stuck in local minima because of the extended momentum could bias it into the nearby local minima [Tripuraneni *et al.*, 2017].

Furthermore, to compare the mixing speed of FHMC and HMC, we plot the trajectory of FHMC and HMC in this task on the first 3000 time steps as shown in Figure 3.

From Figure 3, we could conclude that the proposed FHMC could jump out of the local minima much more efficiently than other baseline methods. This property could be used to efficiently sampling high-dimensional loss functions such as neural networks. We will demonstrate our proposed methods efficiency in training deep neural networks, such as variational autoencoders as below.

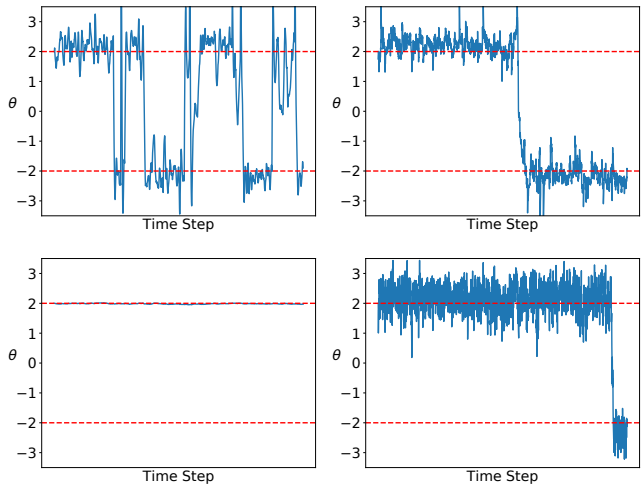

Figure 3: (TopLeft) FHMC result; (TopRight) FLD result; (DownLeft) HMC result; (DownRight) LD result

6.2 Variational Autoencoders

The variational autoencoder (VAE) is an efficient algorithm for training direct graphical models with continuous latent variables [Kingma and Welling, 2013]. We used the training set of MNIST dataset consisting of 60000 training images for this task. We use multi-layered perceptrons(MLPs) with Gaussian distributed outputs for the encoder to encode the input images to latent spaces. Then we use MLPs with Bernoulli distributed outputs to decode the encoded features in the latent spaces. The number of hidden neurons in the MLP is 500 and the number of latent vectors is 20. The batch size is set to be 128. Our implementation is adapted from [2]. The total likelihood loss that includes the negative log likelihood and KL-divergence is measured. We compare our proposed method with popular optimization algorithms including SGD, Adam and RMSprop. The best parameter setting for each method are: SGFHMC(learning rate is 0.03, momentum is 0.9, α is 1.6), SGD(learning rate is 0.003, momentum is 0.2), Adam(learning rate is 0.0001), RMSprop (learning rate is 0.0001). We run the model with different methods and the result is shown in Figure 4. From Figure 4, after efficiently exploring parameter space in the sampling phase at around 17 epochs with SGFHMC, the proposed algorithm then converges very fast and achieves the best result.

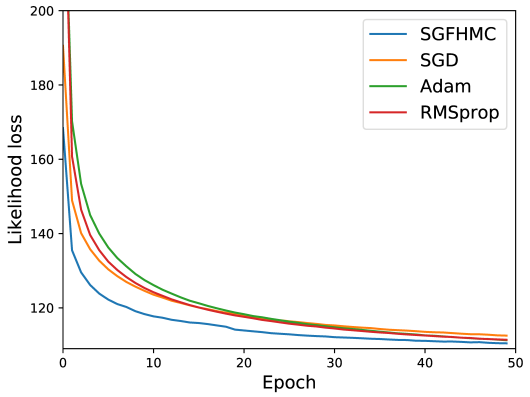

Figure 4: Learning curves of variational autoencoders

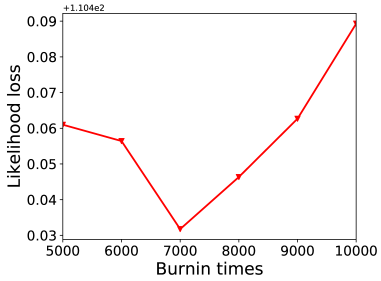

Figure 5: Likelihood loss versus burnin times

[2]https://github.com/hwalsuklee/tensorflow-mnist-VAE

Figure 6: (Left) Original test digits; (Middle) SGFHMC result; (Right) RMSprop result

To further analyze the convergence speed, we plot the generated digits output from models trained by different methods at the same epoch as shown in Figure 6. From Figure 6, we can conclude that the results obtained by SGFHMC are generally clearer and contain less errors than the results obtained by RMSprop. For example, the number in row 3, column 3 which should be '6' is decoded as '0' by RMSprop trained model, while for SGFHMC, the result is similar to the original input.

To analyze the proposed method's robustness against the chosen of burnin times, we plot the results with different burnin times for SGFHMC as shown in Figure 5. From Figure 5, we could conclude that the proposed method is very robust against different burnin times. This is because the proposed SGFHMC could mix fast and the target "fat mode" could be sampled at high probabilities.

6.3 Recurrent Neural Networks for Language Modeling

We test our method on the task of character prediction using LSTM networks. The objective is to minimize the per-character perplexity, $\frac{1}{N}\sum_{i=1}^{N}\exp\left(\sum_{t=1}^{T_i}-\log p(\mathbf{x}_t^i|\mathbf{x}_1^i,...,\mathbf{x}_{t-1}^i;\boldsymbol{\theta})\right)$, where $\boldsymbol{\theta}$ is a set of parameters for the model, $\mathbf{x}_t^n$ is the observed data and T_i is the length of i-th sentence. The hidden units are set as LSTM units. We run the models with different methods on the PTB dataset with a setting of 2-layer LSTM, 200 LSTM units. We adopt the implementation from Tensorflow official document. Noted that we do not use the learning rate decay for fine-tuning to focus on tuning optimizers only. During training, the dataset is split into training, validation and test dataset. We run the training for 13 epochs and choose parameters and the early-stopping epoch based on the results on validation dataset. The best parameter setting for each method are: SGFHMC(learning rate is 0.6, momentum is 0.5, α is 1.6), SGD(learning rate is 0.1, momentum is 0.1), Adam(learning rate is 0.0003), RMSprop (learning rate is 0.0005). The best training and test perplexities are reached by our method SGFHMC. The learning curves are shown in Figure 7. From Figure 7, we could conclude that SGFHMC converges fastest among all methods. Then, we choose the best model for each method based on the validation learning curve to generate results on test dataset. The results are shown in Table 2. From Table 2, we could conclude that with Lévy-driven SDEs, SGFHMC could have better generalization abilities with fast convergence. Note that although SGD and Adam could achieve similar loss in the training dataset, the Adam's test loss is worse than SGD. This is consistent with the empirical observations that most of the state-of-the-art results are obtained by fine-tuned SGD. For our method, the use of SGFHMC in the sampling phase and SGD in the fine-tuning phase could help training converge faster and generalize better at the same time.

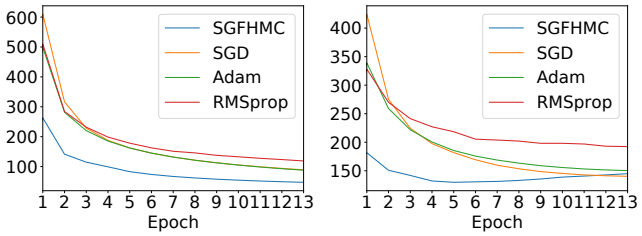

Figure 7: CharRNN learning curves on PTB dataset: (Left) Training perplexity; (Right) Validation perplexity

Method	Test-perplexity
SGFHMC	**125**
SGD	135
Adam	138
RMSprop	187

Table 2: Test perplexity (test loss)

7 Conclusion

We proposed (SG)FHMC-an effective method for sampling and optimization in high-dimensional parameter space based on Lévy diffusion and Hamiltonian Monte Carlo. Extensive empirical evidence indicates the superiority of the proposed methods over the existing methods. Future directions include more accurate and computation cost effective way of approximating the fractional derivatives, and incorporating the adaptive momentum to further enhance the performance.

Acknowledgments

Dr. Zhanxing Zhu is supported by Beijing Municipal Natural Science Foundation, project 4184090.

References

[Ahn *et al.*, 2012] S. Ahn, A. Korattikara, and M. Welling. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring. *arXiv preprint arXiv:1206.6380*, 2012.

[Barndorff-Nielsen and Shephard, 2003] Ole Barndorff-Nielsen and Neil Shephard. Impact of jumps on returns and realised variances: econometric analysis of time-deformed levy processes. Economics Papers 2003-W12, Economics Group, Nuffield College, University of Oxford, 2003.

[Çelik and Duman, 2012] Cem Çelik and Melda Duman. Crank–nicolson method for the fractional diffusion equation with the riesz fractional derivative. *Journal of Computational Physics*, 231(4):1743–1750, 2012.

[Chaudhari *et al.*, 2016] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *CoRR*, abs/1611.01838, 2016.

[Chen *et al.*, 2014] T. Chen, E. B. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1683–1691, 2014.

[Chen *et al.*, 2016] C. Chen, D. Carlson, Z. Gan, C. Li, and L. Carin. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In *AISTATS*, 2016.

[Dauphin *et al.*, 2014] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Keskar *et al.*, 2016] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[Kingma and Welling, 2013] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[Max and Whye, 2011] Welling Max and Teh Yee Whye. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[Rapaport, 2004] D. C. Rapaport. *The art of molecular dynamics simulation*. Cambridge university press, 2004.

[Simsekli, 2017] Umut Simsekli. Fractional Langevin Monte carlo: Exploring Levy driven stochastic differential equations for Markov chain Monte Carlo. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3200–3209, 2017.

[Tripuraneni *et al.*, 2017] Nilesh Tripuraneni, Mark Rowland, Zoubin Ghahramani, and Richard Turner. Magnetic Hamiltonian Monte Carlo. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3453–3461, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[Wolpert and Taqqu, 2005] Robert L. Wolpert and Murad S. Taqqu. Fractional ornstein-uhlenbeck lévy processes and the telecom process: Upstairs and downstairs. *Signal Process.*, 85(8):1523–1545, August 2005.