Large-Scale Bayesian Learning with Stochastic Gradient Markov Chain Monte Carlo

Changyou Chen

Department of Electrical and Computer Engineering, Duke University
cc448@duke.edu

Xidian University
August 19, 2016

Preface

Stochastic gradient Markov chain Monte Carlo:

- A new technique for approximate Bayesian sampling.
- It is about scalable Bayesian learning for big data.
- It draws samples $\{\theta\}$'s from $p(\theta; \mathbf{D})$ where $p(\theta; \mathbf{D})$ is too expensive to be evaluated in each iteration.

Outline

1. Basics on Bayesian Modeling
2. Stochastic Gradient Markov Chain Monte Carlo
3. SG-MCMC for Stochastic Optimization

Large-Scale Bayesian Learning with Stochastic Gradient Markov Chain Monte Carlo Methods

Part One: Basics on Bayesian Modeling

Outline

1. Basics on Bayesian Modeling
   - Bayesian modeling
   - Markov chain Monte Carlo

Outline

1 Basics on Bayesian Modeling
- Bayesian modeling
- Markov chain Monte Carlo

Background

1. We are in an era of abundant data:
    - text, images, videos from the Internet; raw medical notes from doctors, *etc*
2. We need tools for modeling, searching, visualizing, and understanding large-scale data sets.
3. We want our modeling tools:
    - faithfully represent uncertainty in our model structure and parameters
    - automatically deal with noise in our data
    - exhibit robustness
4. Modeling from two aspects: Bayesian and Frequentist.

Background

1. We are in an era of abundant data:
   - text, images, videos from the Internet; raw medical notes from doctors, *etc*
2. We need tools for modeling, searching, visualizing, and understanding large-scale data sets.
3. We want our modeling tools:
   - faithfully represent uncertainty in our model structure and parameters
   - automatically deal with noise in our data
   - exhibit robustness
4. Modeling from two aspects: Bayesian and Frequentist.

Bayesian vs. Frequentist

- When generating data:

Frequentist:

1. Data are a repeatable random sample:
   - there is a frequency
2. Underlying parameters remain constant during this repeatable process.
3. Parameters are fixed.
4. Task is to learn values of the unknown parameters.

1. Data are observed from the realized samples.
2. Parameters are unknown and described probabilistically.
3. Data are fixed.
4. Task is to learn distributions of the unknown parameters.

- In Bayesian modeling, parameters are treated as random variables. The prior is just the prior belief about these parameters.

Bayesian vs. Frequentist

- When generating data:

Frequentist:

1. Data are a repeatable random sample:
    - there is a frequency
2. Underlying parameters remain constant during this repeatable process.
3. Parameters are fixed.
4. Task is to learn values of the unknown parameters.

Bayesian:

1. Data are observed from the realized samples.
2. Parameters are unknown and described probabilistically.
3. Data are fixed.
4. Task is to learn distributions of the unknown parameters.

- In Bayesian modeling, parameters are treated as random variables. The prior is just the prior belief about these parameters.

Bayesian vs. Frequentist

- When generating data:

Frequentist:

1. Data are a repeatable random sample:
   - there is a frequency
2. Underlying parameters remain constant during this repeatable process.
3. Parameters are fixed.
4. Task is to learn values of the unknown parameters.

Bayesian:

1. Data are observed from the realized samples.
2. Parameters are unknown and described probabilistically.
3. Data are fixed.
4. Task is to learn distributions of the unknown parameters.

- In Bayesian modeling, parameters are treated as random variables. The prior is just the prior belief about these parameters.

Bayes' rule

$$p(\mathcal{M}|\mathcal{D}) = \frac{p(\mathcal{D},\mathcal{M})}{p(\mathcal{D})} = \frac{p(\mathcal{M})p(\mathcal{D}|\mathcal{M})}{\int p(\mathcal{M})p(\mathcal{D}|\mathcal{M})\mathrm{d}\mathcal{M}} = \frac{p(\mathcal{M})p(\mathcal{D}|\mathcal{M})}{p(\mathcal{D})} ,$$

where $\mathcal{M}$ and $\mathcal{D}$ are events

- $p(\mathcal{M})$ and $p(\mathcal{D})$: the probabilities of observing $\mathcal{M}$ and $\mathcal{D}$
- $p(\mathcal{D}|\mathcal{M})$, a conditional probability, the probability of observing event $\mathcal{D}$ given that $\mathcal{M}$ is true
- $p(\mathcal{M}|\mathcal{D})$: the probability of observing event $\mathcal{M}$ given that $\mathcal{D}$ is true

Bayes' rule in machine learning

1. Let $\mathcal{D}$ be a given data set; $\mathcal{M}$ be a model.

$$p(\mathcal{M}|\mathcal{D}) = \frac{p(\mathcal{M})p(\mathcal{D}|\mathcal{M})}{p(\mathcal{D})}$$

$p(\mathcal{M})$	:prior probability of $\mathcal{M}$
$p(\mathcal{D}|\mathcal{M})$	:likelihood of $\mathcal{M}$ on data
$p(\mathcal{M}|\mathcal{D})$	:posterior probability
$p(\mathcal{D})$	:marginal likelihood

2. Model comparison: $\mathbb{M} = \{\mathcal{M}\}$.

$$p(\mathbb{M}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbb{M})p(\mathbb{M})}{p(\mathcal{D})}, \quad p(\mathcal{D}|\mathbb{M}) = \int p(\mathcal{D}|\mathcal{M}, \mathbb{M})p(\mathcal{M}|\mathbb{M})\mathrm{d}\mathcal{M}$$

3. Prediction under posterior distribution:

$$p(\mathbf{x}|\mathcal{D}, \mathbb{M}) = \int p(\mathbf{x}|\mathcal{M}, \mathcal{D}, \mathbb{M})p(\mathcal{M}|\mathcal{D}, \mathbb{M})\mathrm{d}\mathcal{M}$$

- $p(\mathbf{x}|\mathcal{M}, \mathcal{D}, \mathbb{M}) = p(\mathbf{x}|\mathcal{M})$ for most models

Bayes' rule in machine learning

1. Let $\mathcal{D}$ be a given data set; $\mathcal{M}$ be a model.

$$p(\mathcal{M}|\mathcal{D}) = \frac{p(\mathcal{M})p(\mathcal{D}|\mathcal{M})}{p(\mathcal{D})}$$

$p(\mathcal{M})$	:prior probability of $\mathcal{M}$
$p(\mathcal{D}|\mathcal{M})$	:likelihood of $\mathcal{M}$ on data
$p(\mathcal{M}|\mathcal{D})$	:posterior probability
$p(\mathcal{D})$	:marginal likelihood

2. Model comparison: $\mathbb{M} = \{\mathcal{M}\}$.

$$p(\mathbb{M}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbb{M})p(\mathbb{M})}{p(\mathcal{D})}, \quad p(\mathcal{D}|\mathbb{M}) = \int p(\mathcal{D}|\mathcal{M}, \mathbb{M})p(\mathcal{M}|\mathbb{M})\mathrm{d}\mathcal{M}$$

3. Prediction under posterior distribution:

$$p(\mathbf{x}\,|\mathcal{D}, \mathbb{M}) = \int p(\mathbf{x}\,|\mathcal{M}, \mathcal{D}, \mathbb{M})p(\mathcal{M}|\mathcal{D}, \mathbb{M})\mathrm{d}\mathcal{M}$$

- $p(\mathbf{x}\,|\mathcal{M}, \mathcal{D}, \mathbb{M}) = p(\mathbf{x}\,|\mathcal{M})$ for most models

Why be Bayesian?

Theoretically:

1. Infinite Exchangeability:

$$\forall n, \forall \pi \text{ (permutation) }, p(\mathbf{x}_1, \cdots, \mathbf{x}_n) = p(\mathbf{x}_{\pi(1)}, \cdots, \mathbf{x}_{\pi(n)})$$

2. Infinite exchangeability means:
   - The way data items are ordered or indexed does not matter
   - Model is unaffected by existence of additional unobserved data items, *e.g.*, test items
     - to predict m additional test items, we need

$$p(\mathbf{x}_1, \cdots, \mathbf{x}_n, \mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+m})$$

     - if not infinitely exchangeable, predictive probabilities will be different for different ordering of training data
3. Exchangeability is a common assumption for most models.

Why be Bayesian?

Theoretically:

1. Infinite Exchangeability:

$$\forall n, \forall \pi \text{ (permutation) }, p(\mathbf{x}_1, \cdots, \mathbf{x}_n) = p(\mathbf{x}_{\pi(1)}, \cdots, \mathbf{x}_{\pi(n)})$$

2. Infinite exchangeability means:
   - The way data items are ordered or indexed does not matter
   - Model is unaffected by existence of additional unobserved data items, *e.g.*, test items
     - ★ to predict m additional test items, we need

$$p(\mathbf{x}_1, \cdots, \mathbf{x}_n, \mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+m})$$

     - ★ if not infinitely exchangeable, predictive probabilities will be different for different ordering of training data

3. Exchangeability is a common assumption for most models.

Why be Bayesian?

Theoretically:

1. Infinite Exchangeability:

$$\forall n, \forall \pi \text{ (permutation) }, p(\mathbf{x}_1, \cdots, \mathbf{x}_n) = p(\mathbf{x}_{\pi(1)}, \cdots, \mathbf{x}_{\pi(n)})$$

2. Infinite exchangeability means:
   - The way data items are ordered or indexed does not matter
   - Model is unaffected by existence of additional unobserved data items, *e.g.*, test items
     - to predict m additional test items, we need

$$p(\mathbf{x}_1, \cdots, \mathbf{x}_n, \mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+m})$$

     - if not infinitely exchangeable, predictive probabilities will be different for different ordering of training data

3. Exchangeability is a common assumption for most models.

Why be Bayesian?

Theoretically:

1. Infinite Exchangeability:

$$\forall n, \forall \pi \text{ (permutation) }, p(\mathbf{x}_1, \cdots, \mathbf{x}_n) = p(\mathbf{x}_{\pi(1)}, \cdots, \mathbf{x}_{\pi(n)})$$

2. De Finetti's Theorem (1955): if $(\mathbf{x}_1, \mathbf{x}_2, \cdots)$ are infinitely exchangeable, then $\forall n$,

$$p(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \int \prod_{i=1}^{n} p(\mathbf{x}_i | \mathcal{M}) \mathrm{d}P(\mathcal{M})$$

   for some random variable $\mathcal{M}$ with probability measure $P(\mathcal{M})$
   - $\mathcal{M}$ is the model in Bayes' rule, with prior measure P

Why be Bayesian?

Practically:

1. Model parameter uncertainty in prediction:

$$p(\mathbf{x}\,|\mathcal{D}) = \int p(\mathbf{x}\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\mathcal{M}$$

   - an effective way to deal with overfiting

2. In frequentist, the data are generated from a fixed model $\mathcal{M}^*$, the prediction is:

$$p(\mathbf{x}\,|\mathcal{D}) = \int p(\mathbf{x}\,|\mathcal{M})\delta(\mathcal{M} = \mathcal{M}^*)\mathrm{d}\mathcal{M} = p(\mathbf{x}\,|\mathcal{M}^*)$$

   where $\mathcal{M}^*$ is usually obtained using optimization

   - easily get overfiting when optimizing $\mathcal{M}^*$

Why be Bayesian?

Practically:

1. Model parameter uncertainty in prediction:

$$p(\mathbf{x}\,|\mathcal{D}) = \int p(\mathbf{x}\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\mathcal{M}$$

   - an effective way to deal with overfiting

2. In frequentist, the data are generated from a fixed model $\mathcal{M}^*$, the prediction is:

$$p(\mathbf{x}\,|\mathcal{D}) = \int p(\mathbf{x}\,|\mathcal{M})\delta(\mathcal{M} = \mathcal{M}^*)\mathrm{d}\mathcal{M} = p(\mathbf{x}\,|\mathcal{M}^*)$$

   where $\mathcal{M}^*$ is usually obtained using optimization

   - easily get overfiting when optimizing $\mathcal{M}^*$

Challenges for being Bayesian

1. Computing integrals could be computationally intractable.
2. Prediction:

$$p(\mathbf{x}\,|\mathcal{D}) = \int p(\mathbf{x}\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\mathcal{M}$$

3. The presence of latent variables results in additional dimensions that need to be marginalized out.

$$p(\mathbf{x}\,|\mathcal{D}) = \int\int p(\mathbf{x},\theta\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\,\boldsymbol{\theta}\,\mathrm{d}\mathcal{M}$$

Challenges for being Bayesian

1. Computing integrals could be computationally intractable.
2. Prediction:

$$p(\mathbf{x}\,|\mathcal{D}) = \int p(\mathbf{x}\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\mathcal{M}$$

3. The presence of latent variables results in additional dimensions that need to be marginalized out.

$$p(\mathbf{x}\,|\mathcal{D}) = \int\int p(\mathbf{x}, \boldsymbol{\theta}\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\,\boldsymbol{\theta}\,\mathrm{d}\mathcal{M}$$

Challenges for being Bayesian

1. Computing integrals could be computationally intractable.
2. Prediction:

$$p(\mathbf{x}\,|\mathcal{D}) = \int p(\mathbf{x}\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\mathcal{M}$$

3. The presence of latent variables results in additional dimensions that need to be marginalized out.

$$p(\mathbf{x}\,|\mathcal{D}) = \int\int p(\mathbf{x},\boldsymbol{\theta}\,|\mathcal{M})p(\mathcal{M}|\mathcal{D})\mathrm{d}\,\boldsymbol{\theta}\,\mathrm{d}\mathcal{M}$$

Approximation methods for marginalization[1]

1. Laplace approximation
2. Bayesian Information Criterion (BIC)
3. Variational inference
4. Expectation Propagation (EP)
5. Markov chain Monte Carlo methods (MCMC)
6. . . .

[1] from Zoubin Ghahramani's talk

Approximation methods for marginalization[1]

1. Laplace approximation
2. Bayesian Information Criterion (BIC)
3. Variational inference
4. Expectation Propagation (EP)
5. Markov chain Monte Carlo methods (MCMC)
6. ...

[1] from Zoubin Ghahramani's talk

Outline

1. Basics on Bayesian Modeling
   - Bayesian modeling
   - Markov chain Monte Carlo

Monte Carlo methods

- Monte Carlo method is about drawing a set of samples:

$$\theta_l \sim p(\theta), \quad l = 1, 2, \cdots, L$$

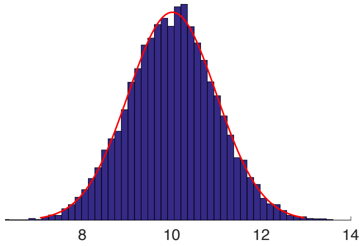
- Approximate the target distribution $p(\theta)$ as count frequency:

$$p(\theta) \approx \frac{1}{L} \sum_{l=1}^{L} \delta(\theta, \theta_l)$$

- An intractable integration is approximated as:

$$\int f(\theta) p(\theta) \approx \frac{1}{L} \sum_{l=1}^{L} f(\theta_l)$$

- In Bayesian modeling, $p(\theta)$ is usually a posterior distribution, the integral is a predicted quantity.



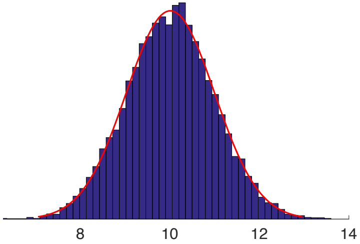
Monte Carlo methods

- Monte Carlo method is about drawing a set of samples:

$$\boldsymbol{\theta}_l \sim p(\boldsymbol{\theta}), \quad l = 1, 2, \cdots, L$$

- Approximate the target distribution $p(\boldsymbol{\theta})$ as count frequency:

$$p(\boldsymbol{\theta}) \approx \frac{1}{L} \sum_{l=1}^{L} \delta(\boldsymbol{\theta}, \boldsymbol{\theta}_l)$$



- An intractable integration is approximated as:

$$\int f(\theta) p(\theta) \approx \frac{1}{L} \sum_{l=1}^{L} f(\theta_l)$$

- In Bayesian modeling, $p(\theta)$ is usually a posterior distribution, the integral is a predicted quantity.

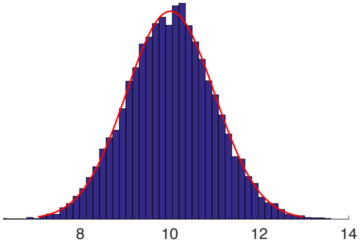
Monte Carlo methods

- Monte Carlo method is about drawing a set of samples:

$$\boldsymbol{\theta}_l \sim p(\boldsymbol{\theta}), \quad l = 1, 2, \cdots, L$$

- Approximate the target distribution $p(\boldsymbol{\theta})$ as count frequency:

$$p(\boldsymbol{\theta}) \approx \frac{1}{L} \sum_{l=1}^{L} \delta(\boldsymbol{\theta}, \boldsymbol{\theta}_l)$$



- An intractable integration is approximated as:

$$\int f(\boldsymbol{\theta}) p(\boldsymbol{\theta}) \approx \frac{1}{L} \sum_{l=1}^{L} f(\boldsymbol{\theta}_l)$$

- In Bayesian modeling, $p(\theta)$ is usually a posterior distribution, the integral is a predicted quantity.

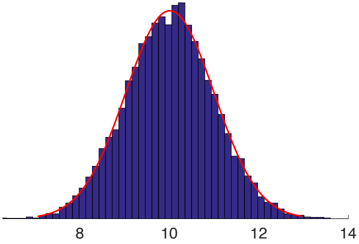
Monte Carlo methods

- Monte Carlo method is about drawing a set of samples:

$$\boldsymbol{\theta}_l \sim p(\boldsymbol{\theta}), \quad l = 1, 2, \cdots, L$$

- Approximate the target distribution $p(\boldsymbol{\theta})$ as count frequency:

$$p(\boldsymbol{\theta}) \approx \frac{1}{L} \sum_{l=1}^{L} \delta(\boldsymbol{\theta}, \boldsymbol{\theta}_l)$$



- An intractable integration is approximated as:

$$\int f(\boldsymbol{\theta}) p(\boldsymbol{\theta}) \approx \frac{1}{L} \sum_{l=1}^{L} f(\boldsymbol{\theta}_l)$$

- In Bayesian modeling, $p(\boldsymbol{\theta})$ is usually a posterior distribution, the integral is a predicted quantity.

How does the approximation work?

1. An intractable integration is approximated as:

$$\int f(\boldsymbol{\theta})p(\boldsymbol{\theta}) \approx \frac{1}{L}\sum_{l=1}^{L} f(\boldsymbol{\theta}_l) \triangleq \tilde{f}$$

2. If $\{\theta_l\}$'s are independent:

$$\mathbb{E}\tilde{f} = \mathbb{E}f, \quad \text{Var}(\tilde{f}) = \frac{1}{L}\text{Var}(f)$$

   - the variance decreases linearly w.r.t. the number of samples, and independent of the dimension of θ

3. However, obtaining independent samples is hard:
   - usually resort to drawing dependent samples with Markov chain Monte Carlo (MCMC)

How does the approximation work?

1. An intractable integration is approximated as:

$$\int f(\boldsymbol{\theta})p(\boldsymbol{\theta}) \approx \frac{1}{L}\sum_{l=1}^{L} f(\boldsymbol{\theta}_l) \triangleq \tilde{f}$$

2. If $\{\boldsymbol{\theta}_l\}$'s are independent:

$$\mathbb{E}\tilde{f} = \mathbb{E}f, \quad \mathrm{Var}(\tilde{f}) = \frac{1}{L}\mathrm{Var}(f)$$

   - the variance decreases linearly w.r.t. the number of samples, and independent of the dimension of $\boldsymbol{\theta}$

3. However, obtaining independent samples is hard:
   - usually resort to drawing dependent samples with Markov chain Monte Carlo (MCMC)

How does the approximation work?

1. An intractable integration is approximated as:

$$\int f(\boldsymbol{\theta})p(\boldsymbol{\theta}) \approx \frac{1}{L}\sum_{l=1}^{L} f(\boldsymbol{\theta}_l) \triangleq \tilde{f}$$

2. If $\{\boldsymbol{\theta}_l\}$'s are independent:

$$\mathbb{E}\tilde{f} = \mathbb{E}f, \quad \mathsf{Var}(\tilde{f}) = \frac{1}{L}\mathsf{Var}(f)$$

   - the variance decreases linearly w.r.t. the number of samples, and independent of the dimension of $\boldsymbol{\theta}$

3. However, obtaining independent samples is hard:
   - usually resort to drawing dependent samples with Markov chain Monte Carlo (MCMC)

MCMC example: a Gaussian model

1. Assume the following generative process (with $\alpha = 5, \beta = 1$):

$$\begin{aligned} x_i \mid \mu, \tau &\sim N(\mu, 1/\tau), \quad i = 1, \cdots, n = 1000 \\ \mu \mid \tau, \{x_i\} &\sim N(\mu_0, 1/\tau), \\ \tau &\sim \text{Gamma}(\alpha, \beta) \end{aligned}$$

2. Posterior distribution:
$p(\mu, \tau|\{x_i\}) \propto \left[\prod_{i=1}^{n} N(x_i; \mu, 1/\tau)\right] N(\mu; \mu_0, 1/\tau)\text{Gamma}(\tau; \alpha, \beta)$
3. Marginal posterior distributions for μ and τ are available:

$$p(\mu|\{x_i\}) \propto \left(2\beta + (\mu - \mu_0)^2 + \sum_i (x_i - \mu)^2\right)^{-\alpha-(n+1)/2}$$

$$p(\tau|\{x_i\}) = \text{Gamma}\left(\alpha + \frac{n}{2}, \beta + \frac{1}{2}\sum_i (x_i - \bar{x})^2 + \frac{n}{2(n+1)}(\bar{x} - \mu_0)^2\right)$$

- $p(\mu|\{x_i\})$ is a non-standardized Student's t-distribution with mean $(\sum_i x_i + \mu_0)/(n+1)$

MCMC example: a Gaussian model

1. Assume the following generative process (with $\alpha = 5, \beta = 1$):

$$\begin{aligned} x_i|\mu,\tau &\sim N(\mu, 1/\tau), \quad i = 1, \cdots, n = 1000 \\ \mu|\tau,\{x_i\} &\sim N(\mu_0, 1/\tau), \\ \tau &\sim \text{Gamma}(\alpha, \beta) \end{aligned}$$

2. Posterior distribution:
$p(\mu,\tau|\{x_i\}) \propto \left[\prod_{i=1}^n N(x_i;\mu,1/\tau)\right] N(\mu;\mu_0,1/\tau)\text{Gamma}(\tau;\alpha,\beta)$
3. Marginal posterior distributions for μ and τ are available:

$$p(\mu|\{x_i\}) \propto \left(2\beta + (\mu-\mu_0)^2 + \sum_i (x_i-\mu)^2\right)^{-\alpha-(n+1)/2}$$

$$p(\tau|\{x_i\}) = \text{Gamma}\left(\alpha + \frac{n}{2}, \beta + \frac{1}{2}\sum_i (x_i - \bar{x})^2 + \frac{n}{2(n+1)}(\bar{x}-\mu_0)^2\right)$$

   - $p(\mu|\{x_i\})$ is a non-standardized Student's t-distribution with mean $(\sum_i x_i + \mu_0)/(n+1)$

MCMC example: a Gaussian model

1. Assume the following generative process (with $\alpha = 5, \beta = 1$):

$$\begin{aligned} x_i|\mu,\tau &\sim N(\mu, 1/\tau), \quad i = 1, \cdots, n = 1000 \\ \mu|\tau,\{x_i\} &\sim N(\mu_0, 1/\tau), \\ \tau &\sim \text{Gamma}(\alpha, \beta) \end{aligned}$$

2. Posterior distribution:
$p(\mu,\tau|\{x_i\}) \propto \left[\prod_{i=1}^{n} N(x_i;\mu,1/\tau)\right] N(\mu;\mu_0,1/\tau)\text{Gamma}(\tau;\alpha,\beta)$
3. Marginal posterior distributions for μ and τ are available:

$$p(\mu|\{x_i\}) \propto \left(2\beta + (\mu-\mu_0)^2 + \sum_i (x_i-\mu)^2\right)^{-\alpha-(n+1)/2}$$

$$p(\tau|\{x_i\}) = \text{Gamma}\left(\alpha + \frac{n}{2}, \beta + \frac{1}{2}\sum_i (x_i - \bar{x})^2 + \frac{n}{2(n+1)}(\bar{x}-\mu_0)^2\right)$$

- $p(\mu|\{x_i\})$ is a non-standardized Student's t-distribution with mean $(\sum_i x_i + \mu_0)/(n+1)$

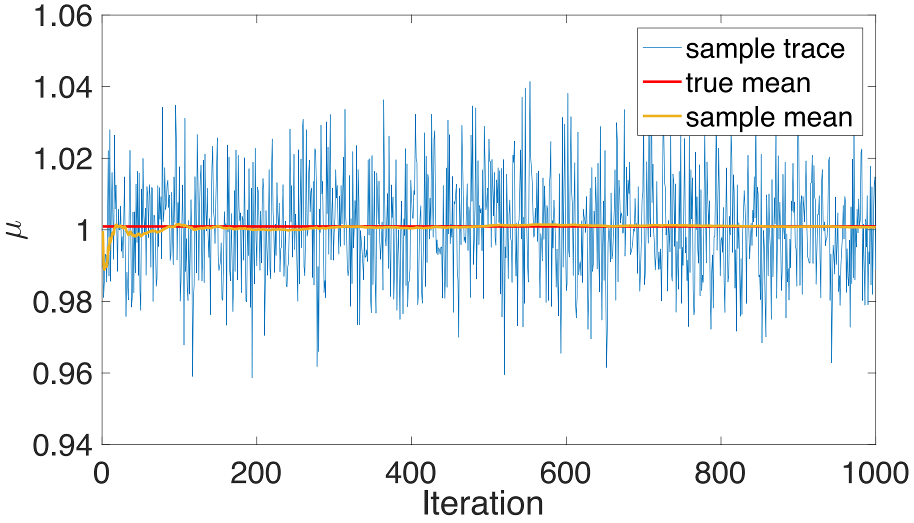
Gibbs sampling μ and τ

1. Conditional distributions:

$$\mu|\tau, \{x_i\} \sim N\left(\frac{n}{n+1}\bar{x} + \frac{1}{n+1}\mu_0, \frac{1}{(n+1)\tau}\right)$$

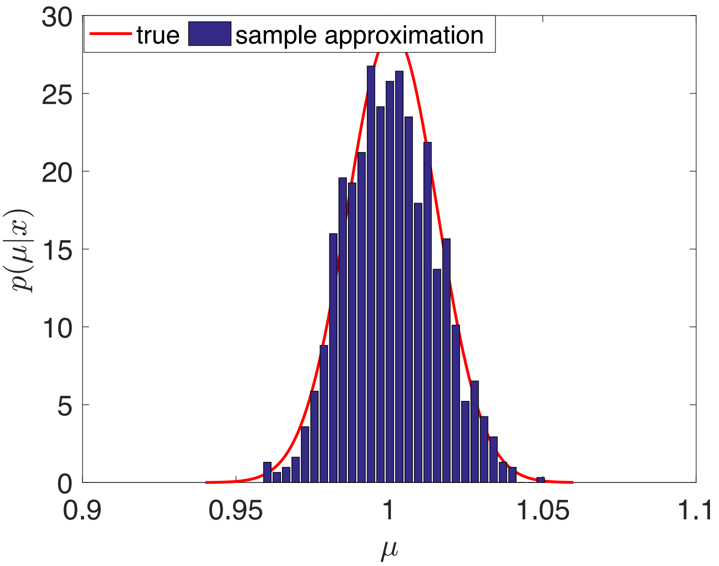
$$\tau|\mu, \{x_i\} \sim \text{Gamma}\left(\alpha + \frac{n+1}{2}, \beta + \frac{\sum_i (x_i - \mu)^2 + (\mu - \mu_0)^2}{2}\right)$$

Trace plot for μ

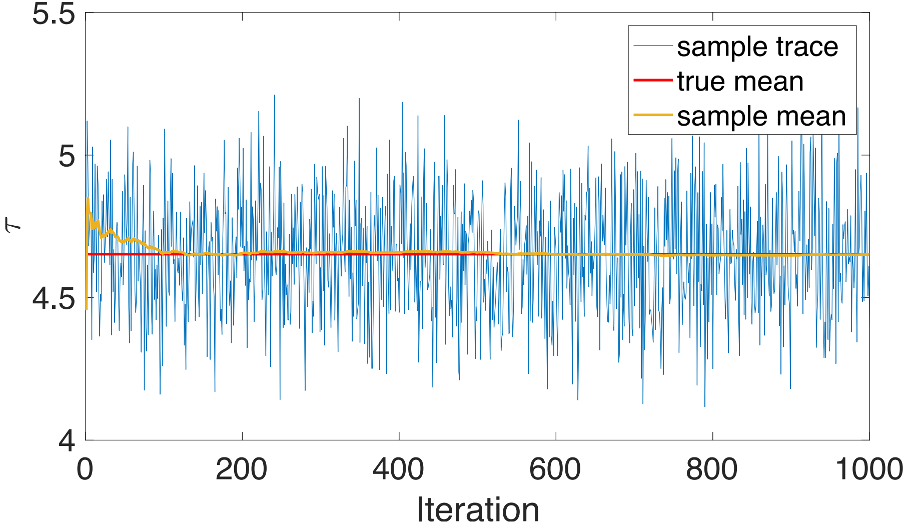


Sample approximation for μ

- True posterior is a non-standardized Student's *t*-distribution.

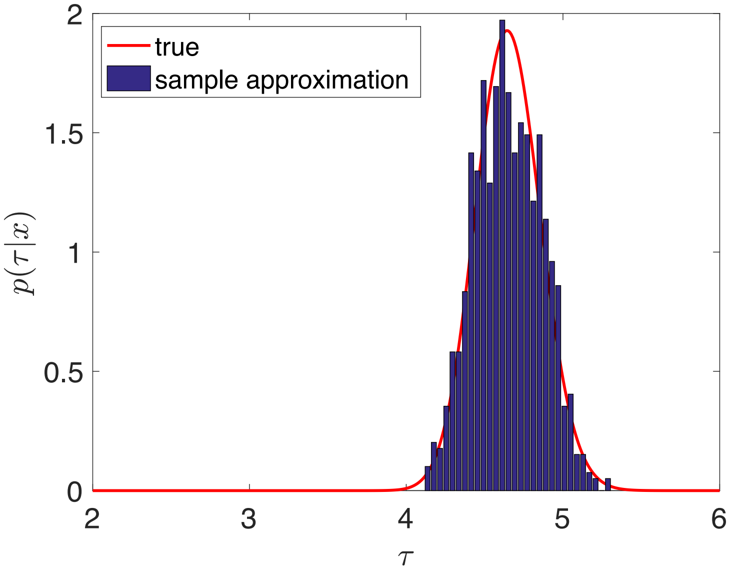


Trace plot for τ



Sample approximation for τ

- True posterior is a Gamma distribution.



Markov chain Monte Carlo methods

1. We are interested in drawing samples from some desired distribution $p^*(\theta) = \frac{1}{Z}\tilde{p}^*(\theta)$.
2. Define a Markov chain:

$$\theta_0 \to \theta_1 \to \theta_2 \to \theta_3 \to \theta_4 \to \theta_5 \to \cdots$$

where $\theta_0 \sim p_0(\theta)$, $\theta_1 \sim p_1(\theta)$, $\cdots$, satisfying

$$p_t(\theta') = \int p_{t-1}(\theta) T(\theta \to \theta') \mathrm{d}\,\theta \ ,$$

where $T(\theta \to \theta')$ is the Markov chain transition probability from θ to θ'.

3. We say $p^*(\theta)$ is an invariant (stationary) distribution of the Markov chain iff:

$$p^*(\theta') = \int p^*(\theta) T(\theta \to \theta') \mathrm{d}\,\theta$$

Markov chain Monte Carlo methods

1. We are interested in drawing samples from some desired distribution $p^*(\boldsymbol{\theta}) = \frac{1}{Z}\tilde{p}^*(\boldsymbol{\theta})$.
2. Define a Markov chain:

$$\boldsymbol{\theta}_0 \to \boldsymbol{\theta}_1 \to \boldsymbol{\theta}_2 \to \boldsymbol{\theta}_3 \to \boldsymbol{\theta}_4 \to \boldsymbol{\theta}_5 \to \cdots$$

   where $\boldsymbol{\theta}_0 \sim p_0(\boldsymbol{\theta})$, $\boldsymbol{\theta}_1 \sim p_1(\boldsymbol{\theta})$, $\cdots$, satisfying

$$p_t(\boldsymbol{\theta}') = \int p_{t-1}(\boldsymbol{\theta}) T(\boldsymbol{\theta} \to \boldsymbol{\theta}') \mathrm{d}\,\boldsymbol{\theta} \ ,$$

   where $T(\boldsymbol{\theta} \to \boldsymbol{\theta}')$ is the Markov chain transition probability from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}'$.
3. We say $p^*(\theta)$ is an invariant (stationary) distribution of the Markov chain iff:

$$p^*(\theta') = \int p^*(\theta) T(\theta \to \theta') \mathrm{d}\,\theta$$

Markov chain Monte Carlo methods

1. We are interested in drawing samples from some desired distribution $p^*(\theta) = \frac{1}{Z}\tilde{p}^*(\theta)$.
2. Define a Markov chain:

$$\theta_0 \to \theta_1 \to \theta_2 \to \theta_3 \to \theta_4 \to \theta_5 \to \cdots$$

   where $\theta_0 \sim p_0(\theta)$, $\theta_1 \sim p_1(\theta)$, $\cdots$, satisfying

$$p_t(\theta') = \int p_{t-1}(\theta) T(\theta \to \theta') \mathrm{d}\,\theta \ ,$$

   where $T(\theta \to \theta')$ is the Markov chain transition probability from θ to θ'.
3. We say $p^*(\theta)$ is an invariant (stationary) distribution of the Markov chain iff:

$$p^*(\theta') = \int p^*(\theta) T(\theta \to \theta') \mathrm{d}\,\theta$$

Markov chain Monte Carlo methods

$$\boldsymbol{\theta}_0 \to \boldsymbol{\theta}_1 \to \boldsymbol{\theta}_2 \to \boldsymbol{\theta}_3 \to \boldsymbol{\theta}_4 \to \boldsymbol{\theta}_5 \to \cdots$$

where $p_t(\boldsymbol{\theta}') = \int p_{t-1}(\boldsymbol{\theta}) T(\boldsymbol{\theta} \to \boldsymbol{\theta}') \mathrm{d}\,\boldsymbol{\theta}$.

1. An invariant (stationary) distribution satisfies:

$$p^*(\boldsymbol{\theta}') = \int p^*(\boldsymbol{\theta}) T(\boldsymbol{\theta} \to \boldsymbol{\theta}') \mathrm{d}\,\boldsymbol{\theta}$$

2. If the Markov chain is ergodic[2], we have:

$$\lim_{t \to \infty} p_t(\theta) = p^*(\theta)$$

3. The task is to design appropriate transition kernel $T(\theta \to \theta')$, so that its invariant distribution coincides $p^*(\theta)$.

[2]It could go from every state to every state.

Markov chain Monte Carlo methods

$$\theta_0 \to \theta_1 \to \theta_2 \to \theta_3 \to \theta_4 \to \theta_5 \to \cdots$$

where $p_t(\theta') = \int p_{t-1}(\theta) T(\theta \to \theta') \mathrm{d}\,\theta$.

1. An invariant (stationary) distribution satisfies:

$$p^*(\theta') = \int p^*(\theta) T(\theta \to \theta') \mathrm{d}\,\theta$$

2. If the Markov chain is ergodic[2], we have:

$$\lim_{t \to \infty} p_t(\theta) = p^*(\theta)$$

3. The task is to design appropriate transition kernel $T(\theta \to \theta')$, so that its invariant distribution coincides $p^*(\theta)$.

[2] It could go from every state to every state.

Markov chain Monte Carlo methods

$$\boldsymbol{\theta}_0 \rightarrow \boldsymbol{\theta}_1 \rightarrow \boldsymbol{\theta}_2 \rightarrow \boldsymbol{\theta}_3 \rightarrow \boldsymbol{\theta}_4 \rightarrow \boldsymbol{\theta}_5 \rightarrow \cdots$$

where $p_t(\boldsymbol{\theta}') = \int p_{t-1}(\boldsymbol{\theta}) T(\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}') \mathrm{d}\,\boldsymbol{\theta}$.

1. An invariant (stationary) distribution satisfies:

$$p^*(\boldsymbol{\theta}') = \int p^*(\boldsymbol{\theta}) T(\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}') \mathrm{d}\,\boldsymbol{\theta}$$

2. If the Markov chain is ergodic[2], we have:

$$\lim_{t\rightarrow\infty} p_t(\boldsymbol{\theta}) = p^*(\boldsymbol{\theta})$$

3. The task is to design appropriate transition kernel $T(\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}')$, so that its invariant distribution coincides $p^*(\boldsymbol{\theta})$.

[2] It could go from every state to every state.

Markov chain Monte Carlo methods

1. A sufficient (but not necessary) condition to guarantee an invariant distribution is the detailed balance condition:

$$p^*(\theta')T(\theta' \to \theta) = p^*(\theta)T(\theta \to \theta')$$

Markov chain Monte Carlo methods

1. A sufficient (but not necessary) condition to guarantee an invariant distribution is the detailed balance condition:

$$p^*(\theta')T(\theta' \to \theta) = p^*(\theta)T(\theta \to \theta')$$

Proof.

Taking integration on both sides over θ:

$$\begin{aligned}
&\int p^*(\theta')T(\theta' \to \theta)\mathrm{d}\theta = \int p^*(\theta)T(\theta \to \theta')\mathrm{d}\theta \\
\Longrightarrow &p^*(\theta')\int T(\theta' \to \theta)\mathrm{d}\theta = \int p^*(\theta)T(\theta \to \theta')\mathrm{d}\theta \\
\Longrightarrow &p^*(\theta') = \int p^*(\theta)T(\theta \to \theta')\mathrm{d}\theta \,,
\end{aligned}$$

by using the fact that $\int T(\theta' \to \theta)\mathrm{d}\theta = 1$. $\square$

Metropolis-Hasting algorithm

1. Design $T(\theta \to \theta')$ as the composition of a proposal distribution $q_t(\theta' \mid \theta)$ and an accept-reject mechanism.
2. At step t, draw a sample[3] $\theta^* \sim q_t(\theta \mid \theta_{t-1})$, and accept it with probability:

$$A_t(\theta^*, \theta_{t-1}) = \min\left(1, \frac{\tilde{p}(\theta^*) q_t(\theta_{t-1} \mid \theta^*)}{\tilde{p}(\theta_{t-1}) q_t(\theta^* \mid \theta_{t-1})}\right)$$

3. The acceptance can be done by:
   - draw a random variable $u \sim \text{Uniform}(0, 1)$
   - accept the sample if $A_t(\theta^*, \theta_{t-1}) > u$
4. The corresponding transition kernel satisfies the detailed balance condition, thus has an invariant probability $p^*(\theta)$.

[3] A standard setting of $q_t(\theta \mid \theta_{t-1})$ is a normal distribution with mean θ_{t-1} and tunable variance.

Metropolis-Hasting algorithm

1. Design $T(\boldsymbol{\theta} \to \boldsymbol{\theta}')$ as the composition of a proposal distribution $q_t(\boldsymbol{\theta}' \,|\, \boldsymbol{\theta})$ and an accept-reject mechanism.
2. At step t, draw a sample[3] $\boldsymbol{\theta}^* \sim q_t(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_{t-1})$, and accept it with probability:

$$A_t(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{t-1}) = \min\left(1, \frac{\tilde{p}(\boldsymbol{\theta}^*) q_t(\boldsymbol{\theta}_{t-1} \,|\, \boldsymbol{\theta}^*)}{\tilde{p}(\boldsymbol{\theta}_{t-1}) q_t(\boldsymbol{\theta}^* \,|\, \boldsymbol{\theta}_{t-1})}\right)$$

3. The acceptance can be done by:
   - draw a random variable $u \sim \text{Uniform}(0, 1)$
   - accept the sample if $A_t(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{t-1}) > u$
4. The corresponding transition kernel satisfies the detailed balance condition, thus has an invariant probability $p^*(\boldsymbol{\theta})$.

[3] A standard setting of $q_t(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{t-1})$ is a normal distribution with mean $\boldsymbol{\theta}_{t-1}$ and tunable variance.

Metropolis-Hasting algorithm

1. Design $T(\boldsymbol{\theta} \to \boldsymbol{\theta}')$ as the composition of a proposal distribution $q_t(\boldsymbol{\theta}' \mid \boldsymbol{\theta})$ and an accept-reject mechanism.
2. At step t, draw a sample[3] $\boldsymbol{\theta}^* \sim q_t(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{t-1})$, and accept it with probability:

$$A_t(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{t-1}) = \min\left(1, \frac{\tilde{p}(\boldsymbol{\theta}^*) q_t(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}^*)}{\tilde{p}(\boldsymbol{\theta}_{t-1}) q_t(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}_{t-1})}\right)$$

3. The acceptance can be done by:
   - draw a random variable $u \sim \text{Uniform}(0, 1)$
   - accept the sample if $A_t(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{t-1}) > u$
4. The corresponding transition kernel satisfies the detailed balance condition, thus has an invariant probability $p^*(\theta)$.

[3] A standard setting of $q_t(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{t-1})$ is a normal distribution with mean $\boldsymbol{\theta}_{t-1}$ and tunable variance.

Metropolis-Hasting algorithm

1. Design $T(\boldsymbol{\theta} \to \boldsymbol{\theta}')$ as the composition of a proposal distribution $q_t(\boldsymbol{\theta}' \mid \boldsymbol{\theta})$ and an accept-reject mechanism.
2. At step t, draw a sample[3] $\boldsymbol{\theta}^* \sim q_t(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{t-1})$, and accept it with probability:

$$A_t(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{t-1}) = \min\left(1, \frac{\tilde{p}(\boldsymbol{\theta}^*) q_t(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}^*)}{\tilde{p}(\boldsymbol{\theta}_{t-1}) q_t(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}_{t-1})}\right)$$

3. The acceptance can be done by:
    - draw a random variable $u \sim \text{Uniform}(0, 1)$
    - accept the sample if $A_t(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{t-1}) > u$
4. The corresponding transition kernel satisfies the detailed balance condition, thus has an invariant probability $p^*(\boldsymbol{\theta})$.

[3] A standard setting of $q_t(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{t-1})$ is a normal distribution with mean $\boldsymbol{\theta}_{t-1}$ and tunable variance.

Metroplis-Hasting algorithm

1. The corresponding transition kernel:

$$T(\theta \to \theta') = q_t(\theta^* \,|\, \theta_{t-1}) A_t(\theta^*, \theta_{t-1})$$

2. Satisfying the detailed balance condition:

$$\begin{aligned}
&p(\theta_{t-1}) q_t(\theta^* \,|\, \theta_{t-1}) A_t(\theta^*, \theta_{t-1}) \\
&= \min\left(p(\theta_{t-1}) q_t(\theta^* \,|\, \theta_{t-1}), p(\theta^*) q_t(\theta_{t-1} \,|\, \theta^*)\right) \\
&= \min\left(p(\theta^*) q_t(\theta_{t-1} \,|\, \theta^*), p(\theta_{t-1}) q_t(\theta^* \,|\, \theta_{t-1})\right) \\
&= p(\theta^*) q_t(\theta_{t-1} \,|\, \theta^*) \min\left(1, \frac{p(\theta_{t-1}) q_t(\theta^* \,|\, \theta_{t-1})}{p(\theta^*) q_t(\theta_{t-1} \,|\, \theta^*)}\right) \\
&= p(\theta^*) q_t(\theta_{t-1} \,|\, \theta^*) A_t(\theta_{t-1}, \theta^*)
\end{aligned}$$

Metroplis-Hasting algorithm

1. The corresponding transition kernel:

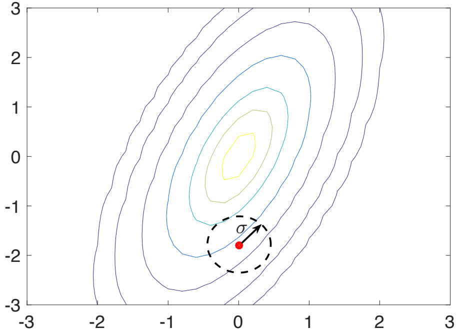
$$T(\theta \to \theta') = q_t(\theta^* \mid \theta_{t-1}) A_t(\theta^*, \theta_{t-1})$$

2. Satisfying the detailed balance condition:

$$\begin{aligned}
& p(\theta_{t-1}) q_t(\theta^* \mid \theta_{t-1}) A_t(\theta^*, \theta_{t-1}) \\
= & \min\left(p(\theta_{t-1}) q_t(\theta^* \mid \theta_{t-1}), p(\theta^*) q_t(\theta_{t-1} \mid \theta^*)\right) \\
= & \min\left(p(\theta^*) q_t(\theta_{t-1} \mid \theta^*), p(\theta_{t-1}) q_t(\theta^* \mid \theta_{t-1})\right) \\
= & p(\theta^*) q_t(\theta_{t-1} \mid \theta^*) \min\left(1, \frac{p(\theta_{t-1}) q_t(\theta^* \mid \theta_{t-1})}{p(\theta^*) q_t(\theta_{t-1} \mid \theta^*)}\right) \\
= & p(\theta^*) q_t(\theta_{t-1} \mid \theta^*) A_t(\theta_{t-1}, \theta^*)
\end{aligned}$$

Discussion on the proposal distribution

1. Standard proposal distribution is an isotropic Gaussian center at the current state with variance σ:
   - small σ leads to high acceptance rate, but moves too slow
   - large σ moves fast, but leads to high rejection rate
2. How to choose better proposals?



Gibbs sampler

1. Assume $\boldsymbol{\theta}$ is multi-dimensional[4], $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_k, \cdots, \boldsymbol{\theta}_K)$, denote $\boldsymbol{\theta}_{-k} \triangleq \{\boldsymbol{\theta}_j : j \neq k\}$.
2. Sample θ_k sequentially, with proposal distribution being the true conditional distribution:

$$q_k(\theta^* \mid \theta) = p(\theta_k^* \mid \theta_{-k})$$

3. Note $\theta_{-k}^* = \theta_{-k}$, $p(\theta) = p(\theta_k \mid \theta_{-k})p(\theta_{-k})$.
4. The MH acceptance probability is:

$$A(\theta^*, \theta) = \frac{p(\theta^*)q_k(\theta \mid \theta^*)}{p(\theta)q_k(\theta^* \mid \theta)} = \frac{p(\theta_k^* \mid \theta_{-k}^*)p(\theta_{-k}^*)p(\theta_k \mid \theta_{-k}^*)}{p(\theta_k^* \mid \theta_{-k})p(\theta_{-k})p(\theta_k \mid \theta_{-k})}$$
$$= 1$$

[4] One dimensional random variable is relatively easy to sample.

Gibbs sampler

1. Assume θ is multi-dimensional[4], $\theta = (\theta_1, \cdots, \theta_k, \cdots, \theta_K)$, denote $\theta_{-k} \triangleq \{\theta_j : j \neq k\}$.
2. Sample θ_k sequentially, with proposal distribution being the true conditional distribution:

$$q_k(\theta^* \mid \theta) = p(\theta_k^* \mid \theta_{-k})$$

3. Note $\theta_{-k}^* = \theta_{-k}$, $p(\theta) = p(\theta_k \mid \theta_{-k}) p(\theta_{-k})$.
4. The MH acceptance probability is:

$$A(\theta^*, \theta) = \frac{p(\theta^*) q_k(\theta \mid \theta^*)}{p(\theta) q_k(\theta^* \mid \theta)} = \frac{p(\theta_k^* \mid \theta_{-k}^*) p(\theta_{-k}^*) p(\theta_k \mid \theta_{-k}^*)}{p(\theta_k^* \mid \theta_{-k}) p(\theta_{-k}) p(\theta_k \mid \theta_{-k})}$$
$$= 1$$

[4] One dimensional random variable is relatively easy to sample.

Gibbs sampler

1. Assume $\boldsymbol{\theta}$ is multi-dimensional[4], $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_k, \cdots, \boldsymbol{\theta}_K)$, denote $\boldsymbol{\theta}_{-k} \triangleq \{\boldsymbol{\theta}_j : j \neq k\}$.
2. Sample $\boldsymbol{\theta}_k$ sequentially, with proposal distribution being the true conditional distribution:

$$q_k(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) = p(\boldsymbol{\theta}_k^* \mid \boldsymbol{\theta}_{-k})$$

3. Note $\boldsymbol{\theta}^*_{-k} = \boldsymbol{\theta}_{-k}$, $p(\boldsymbol{\theta}) = p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{-k}) p(\boldsymbol{\theta}_{-k})$.
4. The MH acceptance probability is:

$$A(\boldsymbol{\theta}^*, \boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta}^*) q_k(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}) q_k(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})} = \frac{p(\boldsymbol{\theta}_k^* \mid \boldsymbol{\theta}^*_{-k}) p(\boldsymbol{\theta}^*_{-k}) p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}^*_{-k})}{p(\boldsymbol{\theta}_k^* \mid \boldsymbol{\theta}_{-k}) p(\boldsymbol{\theta}_{-k}) p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{-k})}$$
$$= 1$$

[4] One dimensional random variable is relatively easy to sample.

Discussion of Gibbs sampler

1. No acceptance step, very efficient.
2. Conditional distributions are not always easy to sample.
3. Mix not well when highly variables are correlated.

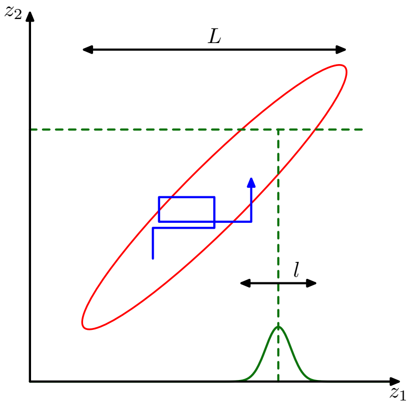
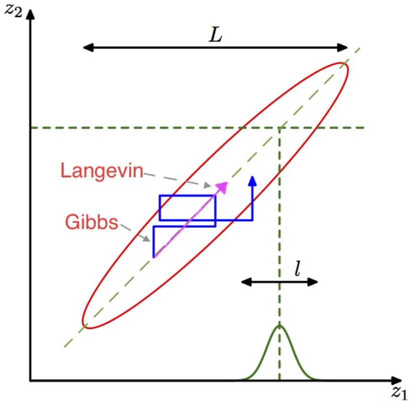


Figure: Sample path does not follow gradients. Figure from PRML, Bishop (2006)

The Metropolis-adjusted Langevin: a better proposal

1. Gibbs sampling travels the parameter space following a zipzag curve, which might be slow in high-dimensional space.
2. The Metropolis-adjusted Langevin uses a proposal that points directly to the center of the probabilistic contour.



The Metropolis-adjusted Langevin: a better proposal

1. Let $E(\boldsymbol{\theta}) \triangleq -\log \tilde{p}(\boldsymbol{\theta})$, the direction of the contour is just the gradient: $-\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$.
2. In iteration l, define the proposal as a Gaussian centering at $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{l-1} - \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{l-1}) h_l$, where h_l is a small stepsize:

$$q(\boldsymbol{\theta}_l \,|\, \boldsymbol{\theta}_{l-1}) = N\left(\boldsymbol{\theta}_l; \boldsymbol{\theta}^*, \sigma^2\right) .$$

3. Need to do an accept-reject step:
   - calculate the acceptance probability:

$$A(\theta^*, \theta_{l-1}) = \frac{\tilde{p}(\theta^*) q(\theta_{l-1} \,|\, \theta^*)}{\tilde{p}(\theta) q(\theta^* \,|\, \theta_{l-1})}$$

   - accept θ^* with probability $A(\theta^*, \theta_{l-1})$, otherwise set $\theta_l = \theta_{l-1}$

The Metropolis-adjusted Langevin: a better proposal

1. Let $E(\boldsymbol{\theta}) \triangleq -\log \tilde{p}(\boldsymbol{\theta})$, the direction of the contour is just the gradient: $-\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$.
2. In iteration l, define the proposal as a Gaussian centering at $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{l-1} - \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}_{l-1}) h_l$, where h_l is a small stepsize:

$$q(\boldsymbol{\theta}_l \,|\, \boldsymbol{\theta}_{l-1}) = N\left(\boldsymbol{\theta}_l; \boldsymbol{\theta}^*, \sigma^2\right) \ .$$

3. Need to do an accept-reject step:
   - calculate the acceptance probability:

$$A(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{l-1}) = \frac{\tilde{p}(\boldsymbol{\theta}^*) q(\boldsymbol{\theta}_{l-1} \,|\, \boldsymbol{\theta}^*)}{\tilde{p}(\boldsymbol{\theta}) q(\boldsymbol{\theta}^* \,|\, \boldsymbol{\theta}_{l-1})}$$

   - accept $\boldsymbol{\theta}^*$ with probability $A(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{l-1})$, otherwise set $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1}$

Hamiltonian Monte Carlo

1. Design a proposal that follows the gradient of the target distribution $p^*(\boldsymbol{\theta}) = \frac{1}{Z}\tilde{p}(\boldsymbol{\theta})$.
2. Construct a landscape with gravitational potential energy, $E(\boldsymbol{\theta}) = -\log \tilde{p}(\boldsymbol{\theta})$.
3. Introduce velocity $\mathbf{v}$ carrying kinetic energy $K(\mathbf{v}) = \mathbf{v}^T \mathbf{v}/2$.
4. Let $H(\boldsymbol{\theta}, \mathbf{v}) \triangleq E(\boldsymbol{\theta}) + K(\mathbf{v})$. Hamiltonian's equation used to describe the evolution of the state $(\boldsymbol{\theta}_t, \mathbf{v}_t)$ along time t[5]:

$$\frac{\mathrm{d}\,\boldsymbol{\theta}}{\mathrm{d}t} = \frac{\partial H}{\partial\,\mathbf{v}}$$
$$\frac{\mathrm{d}\,\mathbf{v}}{\mathrm{d}t} = -\frac{\partial H}{\partial\,\boldsymbol{\theta}}$$

[5] A continuous-time Markov chain.

Hamiltonian Monte Carlo

1. Design a proposal that follows the gradient of the target distribution $p^*(\boldsymbol{\theta}) = \frac{1}{Z}\tilde{p}(\boldsymbol{\theta})$.
2. Construct a landscape with gravitational potential energy, $E(\boldsymbol{\theta}) = -\log \tilde{p}(\boldsymbol{\theta})$.
3. Introduce velocity $\mathbf{v}$ carrying kinetic energy $K(\mathbf{v}) = \mathbf{v}^T \mathbf{v}/2$.
4. Let $H(\boldsymbol{\theta}, \mathbf{v}) \triangleq E(\boldsymbol{\theta}) + K(\mathbf{v})$. Hamiltonian's equation used to describe the evolution of the state $(\boldsymbol{\theta}_t, \mathbf{v}_t)$ along time t[5]:

$$\frac{\mathrm{d}\,\boldsymbol{\theta}}{\mathrm{d}t} = \frac{\partial H}{\partial\,\mathbf{v}}$$
$$\frac{\mathrm{d}\,\mathbf{v}}{\mathrm{d}t} = -\frac{\partial H}{\partial\,\boldsymbol{\theta}}$$

[5] A continuous-time Markov chain.

Hamiltonian Monte Carlo

1. Design a proposal that follows the gradient of the target distribution $p^*(\boldsymbol{\theta}) = \frac{1}{Z}\tilde{p}(\boldsymbol{\theta})$.
2. Construct a landscape with gravitational potential energy, $E(\boldsymbol{\theta}) = -\log \tilde{p}(\boldsymbol{\theta})$.
3. Introduce velocity $\mathbf{v}$ carrying kinetic energy $K(\mathbf{v}) = \mathbf{v}^T \mathbf{v} / 2$.
4. Let $H(\boldsymbol{\theta}, \mathbf{v}) \triangleq E(\boldsymbol{\theta}) + K(\mathbf{v})$. Hamiltonian's equation used to describe the evolution of the state $(\boldsymbol{\theta}_t, \mathbf{v}_t)$ along time t[5]:

$$\frac{\mathrm{d}\,\boldsymbol{\theta}}{\mathrm{d}t} = \frac{\partial H}{\partial\,\mathbf{v}}$$
$$\frac{\mathrm{d}\,\mathbf{v}}{\mathrm{d}t} = -\frac{\partial H}{\partial\,\boldsymbol{\theta}}$$

[5] A continuous-time Markov chain.

Hamiltonian Monte Carlo

1. Design a proposal that follows the gradient of the target distribution $p^*(\boldsymbol{\theta}) = \frac{1}{Z}\tilde{p}(\boldsymbol{\theta})$.
2. Construct a landscape with gravitational potential energy, $E(\boldsymbol{\theta}) = -\log \tilde{p}(\boldsymbol{\theta})$.
3. Introduce velocity $\mathbf{v}$ carrying kinetic energy $K(\mathbf{v}) = \mathbf{v}^T \mathbf{v} / 2$.
4. Let $H(\boldsymbol{\theta}, \mathbf{v}) \triangleq E(\boldsymbol{\theta}) + K(\mathbf{v})$. Hamiltonian's equation used to describe the evolution of the state $(\boldsymbol{\theta}_t, \mathbf{v}_t)$ along time t[5]:

$$\frac{\mathrm{d}\,\boldsymbol{\theta}}{\mathrm{d}t} = \frac{\partial H}{\partial\,\mathbf{v}}$$
$$\frac{\mathrm{d}\,\mathbf{v}}{\mathrm{d}t} = -\frac{\partial H}{\partial\,\boldsymbol{\theta}}$$

[5] A continuous-time Markov chain.

Hamiltonian Monte Carlo

Physics point of view:

1. A dynamic system with total energy or Hamiltonian: $H = E(\theta) + K(\mathbf{v})$.
2. Frictionless ball rolling $(\theta, \mathbf{v}) \to (\theta', \mathbf{v}')$ satisfies energy preserving, $H(\theta', \mathbf{v}') = H(\theta, \mathbf{v})$.
3. Hamiltonian's equation describes the equations of motion of the ball.
4. Ideal Hamiltonian dynamics are time reversible:
   - reverse $\mathbf{v}$ and the ball will return to its start point

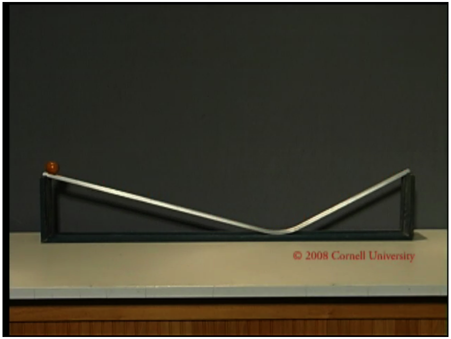


Figure: Rolling ball. Movie from Matthias Liepe

Hamiltonian Monte Carlo

Physics point of view:

1. A dynamic system with total energy or Hamiltonian: $H = E(\theta) + K(\mathbf{v})$.
2. Frictionless ball rolling $(\theta, \mathbf{v}) \rightarrow (\theta', \mathbf{v}')$ satisfies energy preserving, $H(\theta', \mathbf{v}') = H(\theta, \mathbf{v})$.
3. Hamiltonian's equation describes the equations of motion of the ball.
4. Ideal Hamiltonian dynamics are time reversible:
    - reverse $\mathbf{v}$ and the ball will return to its start point

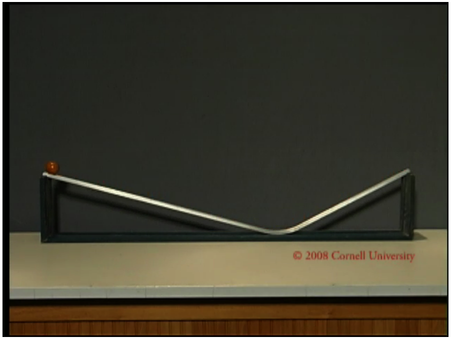


Figure: Rolling ball. Movie from Matthias Liepe

Hamiltonian Monte Carlo

Physics point of view:

1. A dynamic system with total energy or Hamiltonian: $H = E(\theta) + K(\mathbf{v})$.
2. Frictionless ball rolling $(\theta, \mathbf{v}) \to (\theta', \mathbf{v}')$ satisfies energy preserving, $H(\theta', \mathbf{v}') = H(\theta, \mathbf{v})$.
3. Hamiltonian's equation describes the equations of motion of the ball.
4. Ideal Hamiltonian dynamics are time reversible:
   - reverse $\mathbf{v}$ and the ball will return to its start point

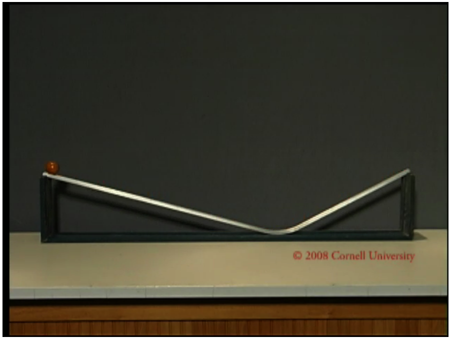


Figure: Rolling ball. Movie from Matthias Liepe

Hamiltonian Monte Carlo

Physics point of view:

1. A dynamic system with total energy or Hamiltonian: $H = E(\theta) + K(\mathbf{v})$.
2. Frictionless ball rolling $(\theta, \mathbf{v}) \to (\theta', \mathbf{v}')$ satisfies energy preserving, $H(\theta', \mathbf{v}') = H(\theta, \mathbf{v})$.
3. Hamiltonian's equation describes the equations of motion of the ball.
4. Ideal Hamiltonian dynamics are time reversible:
   - reverse **v** and the ball will return to its start point

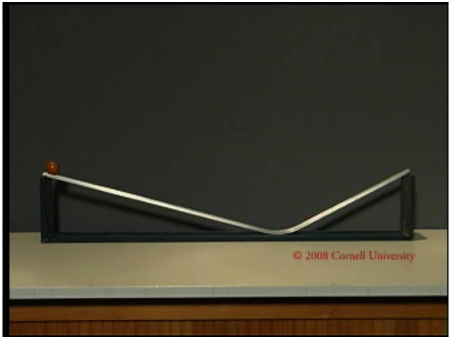


Figure: Rolling ball. Movie from Matthias Liepe

Hamiltonian Monte Carlo

Markov chain point of view:

1. Joint distribution: $p(\theta, \mathbf{v}) \propto e^{-E(\theta) - K(\mathbf{v})} = e^{-H(\theta, \mathbf{v})}$.
2. To generate a sample:
   - Gibbs sampling velocity $\mathbf{v}$ from a Gaussian
   - evolving Hamiltonian dynamics by following Hamiltonian's equation for some time, then flip sign of velocity
   - the resulting $(\theta, \mathbf{v})$ is a random sample from $p(\theta, \mathbf{v})$
3. Proposal (evolving Hamiltonian dynamics) is deterministic and reversible: $q(\theta', \mathbf{v}' \mid \theta, \mathbf{v}) = q(\theta, \mathbf{v} \mid \theta', \mathbf{v}') = 1$.
4. Conservation of energy means $p(\theta, \mathbf{v}) = p(\theta', \mathbf{v}')$.
5. As a result, acceptance rate is always 1.

Hamiltonian Monte Carlo

Markov chain point of view:

1. Joint distribution: $p(\boldsymbol{\theta}, \mathbf{v}) \propto e^{-E(\boldsymbol{\theta})-K(\mathbf{v})} = e^{-H(\boldsymbol{\theta},\mathbf{v})}$.
2. To generate a sample:
   - Gibbs sampling velocity **v** from a Gaussian
   - evolving Hamiltonian dynamics by following Hamiltonian's equation for some time, then flip sign of velocity
   - the resulting $(\boldsymbol{\theta}, \mathbf{v})$ is a random sample from $p(\boldsymbol{\theta}, \mathbf{v})$
3. Proposal (evolving Hamiltonian dynamics) is deterministic and reversible: $q(\theta', \mathbf{v}' \mid \theta, \mathbf{v}) = q(\theta, \mathbf{v} \mid \theta', \mathbf{v}') = 1$.
4. Conservation of energy means $p(\theta, \mathbf{v}) = p(\theta', \mathbf{v}')$.
5. As a result, acceptance rate is always 1.

Hamiltonian Monte Carlo

Markov chain point of view:

1. Joint distribution: $p(\boldsymbol{\theta}, \mathbf{v}) \propto e^{-E(\boldsymbol{\theta})-K(\mathbf{v})} = e^{-H(\boldsymbol{\theta},\mathbf{v})}$.
2. To generate a sample:
   - Gibbs sampling velocity $\mathbf{v}$ from a Gaussian
   - evolving Hamiltonian dynamics by following Hamiltonian's equation for some time, then flip sign of velocity
   - the resulting $(\boldsymbol{\theta}, \mathbf{v})$ is a random sample from $p(\boldsymbol{\theta}, \mathbf{v})$
3. Proposal (evolving Hamiltonian dynamics) is deterministic and reversible: $q(\boldsymbol{\theta}', \mathbf{v}' \mid \boldsymbol{\theta}, \mathbf{v}) = q(\boldsymbol{\theta}, \mathbf{v} \mid \boldsymbol{\theta}', \mathbf{v}') = 1$.
4. Conservation of energy means $p(\boldsymbol{\theta}, \mathbf{v}) = p(\boldsymbol{\theta}', \mathbf{v}')$.
5. As a result, acceptance rate is always 1.

Hamiltonian Monte Carlo

Markov chain point of view:

1. Joint distribution: $p(\boldsymbol{\theta}, \mathbf{v}) \propto e^{-E(\boldsymbol{\theta})-K(\mathbf{v})} = e^{-H(\boldsymbol{\theta},\mathbf{v})}$.
2. To generate a sample:
   - Gibbs sampling velocity **v** from a Gaussian
   - evolving Hamiltonian dynamics by following Hamiltonian's equation for some time, then flip sign of velocity
   - the resulting $(\boldsymbol{\theta}, \mathbf{v})$ is a random sample from $p(\boldsymbol{\theta}, \mathbf{v})$
3. Proposal (evolving Hamiltonian dynamics) is deterministic and reversible: $q(\boldsymbol{\theta}', \mathbf{v}' \mid \boldsymbol{\theta}, \mathbf{v}) = q(\boldsymbol{\theta}, \mathbf{v} \mid \boldsymbol{\theta}', \mathbf{v}') = 1$.
4. Conservation of energy means $p(\boldsymbol{\theta}, \mathbf{v}) = p(\boldsymbol{\theta}', \mathbf{v}')$.
5. As a result, acceptance rate is always 1.

Except we can't simulate Hamiltonian dynamics exactly , *i.e.*, $p(\boldsymbol{\theta}, \mathbf{v}) \neq p(\boldsymbol{\theta}', \mathbf{v}')$

Solving Hamiltonian dynamics

1. Solving the continuous-time differential equation with discretized-time approximation:

$$\begin{cases} \mathrm{d}\,\boldsymbol{\theta} &= \mathbf{v}\,\mathrm{d}t \\ \mathrm{d}\,\mathbf{v} &= \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta})\mathrm{d}t \end{cases} \Longrightarrow \begin{cases} \boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{v}_{l-1}\, h_l \\ \mathbf{v}_l &= \mathbf{v}_{l-1} + \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta}_l) h_l \end{cases}$$

   - proposals follow historical gradients of the distribution contour

2. Need an accept-reject test to design whether accept the proposal, because of the discretization error:
   - proposal is deterministic
   - acceptance probability: $\min\left(1, \exp\left\{H(\boldsymbol{\theta}_l, \mathbf{v}_l) - H(\boldsymbol{\theta}_{l+1}, \mathbf{v}_{l+1})\right\}\right)$

3. Almost identical to SGD with momentum:
   - $\begin{cases} \boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}_{l-1} \\ \mathbf{p}_l &= (1-m)\,\mathbf{p}_{l-1} + \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta}_l)\epsilon_l \end{cases}$
   - they will be make equivalent in the context of stochastic gradient MCMC

Solving Hamiltonian dynamics

1. Solving the continuous-time differential equation with discretized-time approximation:

$$\left\{\begin{array}{ll} \mathrm{d}\,\boldsymbol{\theta} & = \mathbf{v}\,\mathrm{d}t \\ \mathrm{d}\,\mathbf{v} & = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta})\mathrm{d}t \end{array}\right. \Longrightarrow \left\{\begin{array}{ll} \boldsymbol{\theta}_l & = \boldsymbol{\theta}_{l-1} + \mathbf{v}_{l-1}\,h_l \\ \mathbf{v}_l & = \mathbf{v}_{l-1} + \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta}_l) h_l \end{array}\right.$$

   - proposals follow historical gradients of the distribution contour

2. Need an accept-reject test to design whether accept the proposal, because of the discretization error:
   - proposal is deterministic
   - acceptance probability: $\min\left(1, \exp\left\{H(\boldsymbol{\theta}_l, \mathbf{v}_l) - H(\boldsymbol{\theta}_{l+1}, \mathbf{v}_{l+1})\right\}\right)$

3. Almost identical to SGD with momentum:
   - $\left\{\begin{array}{ll} \theta_l & = \theta_{l-1} + \mathbf{p}_{l-1} \\ \mathbf{p}_l & = (1-m)\,\mathbf{p}_{l-1} + \nabla_\theta \log \tilde{p}(\theta_l)\epsilon_l \end{array}\right.$
   - they will be make equivalent in the context of stochastic gradient MCMC

Solving Hamiltonian dynamics

1. Solving the continuous-time differential equation with discretized-time approximation:

$$\begin{cases} \mathrm{d}\,\boldsymbol{\theta} & = \mathbf{v}\,\mathrm{d}t \\ \mathrm{d}\,\mathbf{v} & = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta})\mathrm{d}t \end{cases} \Longrightarrow \begin{cases} \boldsymbol{\theta}_l & = \boldsymbol{\theta}_{l-1} + \mathbf{v}_{l-1}\, h_l \\ \mathbf{v}_l & = \mathbf{v}_{l-1} + \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta}_l) h_l \end{cases}$$

   - proposals follow historical gradients of the distribution contour

2. Need an accept-reject test to design whether accept the proposal, because of the discretization error:
   - proposal is deterministic
   - acceptance probability: $\min\left(1, \exp\left\{H(\boldsymbol{\theta}_l, \mathbf{v}_l) - H(\boldsymbol{\theta}_{l+1}, \mathbf{v}_{l+1})\right\}\right)$

3. Almost identical to SGD with momentum:
   - $\begin{cases} \boldsymbol{\theta}_l & = \boldsymbol{\theta}_{l-1} + \mathbf{p}_{l-1} \\ \mathbf{p}_l & = (1-m)\,\mathbf{p}_{l-1} + \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta}_l)\epsilon_l \end{cases}$
   - they will be make equivalent in the context of stochastic gradient MCMC

Detailed balance

1. Verify that the detailed balance for HMC holds.
   - let the initial state be $(\boldsymbol{\theta}, \mathbf{v})$, the state after Leap-frog simulation be $(\boldsymbol{\theta}', \mathbf{v}')$

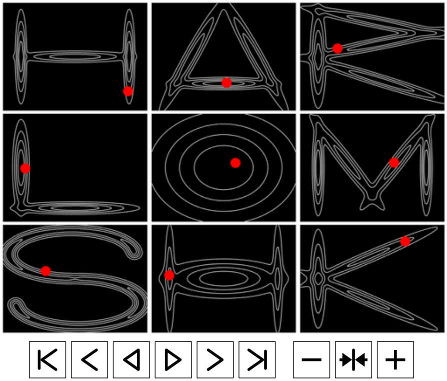
$$
\begin{aligned}
&\frac{1}{Z}\exp(-H(\boldsymbol{\theta}, \mathbf{v}))\min\left(1, exp(-H(\boldsymbol{\theta}', \mathbf{v}') + H(\boldsymbol{\theta}, \mathbf{v}))\right) \\
=&\frac{1}{Z}\min\left(\exp(-H(\boldsymbol{\theta}, \mathbf{v})), \exp(-H(\boldsymbol{\theta}', \mathbf{v}'))\right) \\
=&\frac{1}{Z}\exp(-H(\boldsymbol{\theta}', \mathbf{v}'))\min\left(1, exp(-H(\boldsymbol{\theta}, \mathbf{v}) + H(\boldsymbol{\theta}', \mathbf{v}'))\right)
\end{aligned}
$$

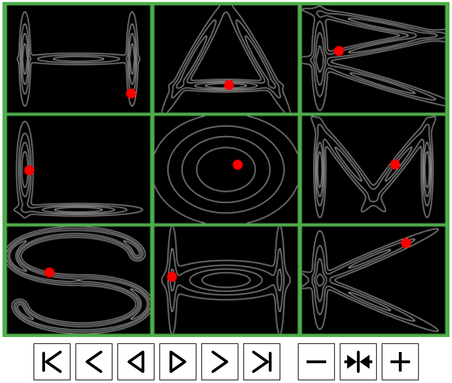
Hamiltonian Monte Carlo algorithm

```
Set $l = 0$
Random initialize a position state $\boldsymbol{\theta}_0$
for $l = 1, 2, \ldots$ do
    Sample a new initial momentum $\mathbf{v}_0 \sim e^{-K(\mathbf{v})}$ (Gaussian)
    Set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{l-1}$
    Run Leap-frog algorithm starting at $(\boldsymbol{\theta}_0, \mathbf{v}_0)$ for $L$ steps to obtain
    proposed states $(\boldsymbol{\theta}^*, \mathbf{v}^*)$
    Calculate the Metropolis acceptance probability:
    $\alpha = \min\left(1, \exp\left(H(\boldsymbol{\theta}_0, \mathbf{v}_0) - H(\boldsymbol{\theta}^*, \mathbf{v}^*)\right)\right)$
    Draw $u \sim \text{Unif}(0, 1)$
        if $u \leq \alpha$, $\boldsymbol{\theta}_l = \boldsymbol{\theta}^*$
        else $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1}$
end
```

Demo: MH vs. HMC

1. Nine mixtures of Gaussians[6].
2. Sequential of samples connected by yellow lines.





[6]Demo by T. Broderick and D. Duvenaud.

Discussion

1. All the above traditional MCMC methods are not scalable in a big-data setting[7], in each iteration:
    - the whole data need to be used to generate a proposal
    - the whole data need to be used to calculate the acceptance probability
    - scales $O(N)$, where N is the number of data samples
2. Scalable MCMC uses sub-data in each iteration,
    - to calculate the acceptance probability[8]
    - to generate proposals with acceptance probability close to 1, and ignore the acceptance step – stochastic gradient MCMC methods (SG-MCMC)

[7] when the number of data samples are large.

[8] A. Korattikara, Y. Chen, and M. Welling. "Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget". In: *ICML*. 2014; R. Bardenet, A. Doucet, and C. Holmes. "Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach". In: *ICML*. 2014.

Discussion

1. All the above traditional MCMC methods are not scalable in a big-data setting[7], in each iteration:
   - the whole data need to be used to generate a proposal
   - the whole data need to be used to calculate the acceptance probability
   - scales $O(N)$, where N is the number of data samples
2. Scalable MCMC uses sub-data in each iteration,
   - to calculate the acceptance probability[8]
   - to generate proposals with acceptance probability close to 1, and ignore the acceptance step – stochastic gradient MCMC methods (SG-MCMC)

[7] when the number of data samples are large.

[8] A. Korattikara, Y. Chen, and M. Welling. “Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget”. In: *ICML*. 2014; R. Bardenet, A. Doucet, and C. Holmes. “Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach”. In: *ICML*. 2014.

Large-Scale Bayesian Learning with Stochastic Gradient Markov Chain Monte Carlo Methods

Part Two: Stochastic Gradient Markov Chain Monte Carlo

Outline

2. Stochastic Gradient Markov Chain Monte Carlo
   - SG-MCMC algorithms
   - Theory

Outline

2. Stochastic Gradient Markov Chain Monte Carlo
   - SG-MCMC algorithms
   - Theory

Two key steps in SG-MCMC

1. Proposals typically follow stochastic gradients of log-posteriors:
   - make samples concentrate on the modes
2. Adding random Gaussian noise to proposals.
   - encourage algorithms to jump out of local modes, and to explore the parameter space

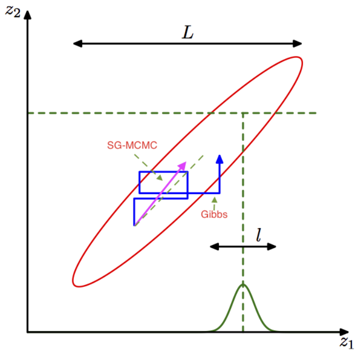


Figure: Proposals of Gibbs and SG-MCMC.

Basic setup

1. Given data $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, a generative model (likelihood) $p(\mathbf{X} | \theta) = \prod_{i=1}^{N} p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$ and prior $p(\boldsymbol{\theta})$, we want to sample from the posterior:

$$p(\boldsymbol{\theta} \,|\, \mathbf{X}) \propto p(\boldsymbol{\theta}) p(\mathbf{X} \,|\, \boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{i=1}^{N} p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$$

2. We are interested in the case when N is extremely large, so that computing $p(\mathbf{X} \,|\, \theta)$ is prohibitively expensive.
3. Define the following two quantities (unnormalized log-posterior and stochastic unnormalized log-posterior):

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

$$\tilde{U}(\boldsymbol{\theta}) \triangleq -\frac{N}{n} \sum_{i=1}^{n} \log p(\mathbf{x}_{\pi_i} \,|\, \theta) - \log p(\theta)$$

where $(\pi_1, \cdots, \pi_N)$ is a random permutation of $(1, \cdots, N)$.

Basic setup

1. Given data $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, a generative model (likelihood) $p(\mathbf{X} \,|\theta) = \prod_{i=1}^N p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$ and prior $p(\boldsymbol{\theta})$, we want to sample from the posterior:

$$p(\boldsymbol{\theta} \,|\, \mathbf{X}) \propto p(\boldsymbol{\theta}) p(\mathbf{X} \,|\, \boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{i=1}^N p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$$

2. We are interested in the case when N is extremely large, so that computing $p(\mathbf{X} \,|\, \boldsymbol{\theta})$ is prohibitively expensive.

3. Define the following two quantities (unnormalized log-posterior and stochastic unnormalized log-posterior):

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

$$\tilde{U}(\boldsymbol{\theta}) \triangleq -\frac{N}{n} \sum_{i=1}^n \log p(\mathbf{x}_{\pi_i} \,|\, \theta) - \log p(\theta)$$

where $(\pi_1, \cdots, \pi_N)$ is a random permutation of $(1, \cdots, N)$.

Basic setup

1. Given data $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, a generative model (likelihood) $p(\mathbf{X} \,|\theta) = \prod_{i=1}^N p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$ and prior $p(\boldsymbol{\theta})$, we want to sample from the posterior:

$$p(\boldsymbol{\theta} \,|\, \mathbf{X}) \propto p(\boldsymbol{\theta}) p(\mathbf{X} \,|\, \boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{i=1}^{N} p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$$

2. We are interested in the case when N is extremely large, so that computing $p(\mathbf{X} \,|\, \boldsymbol{\theta})$ is prohibitively expensive.
3. Define the following two quantities (unnormalized log-posterior and stochastic unnormalized log-posterior):

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

$$\tilde{U}(\boldsymbol{\theta}) \triangleq -\frac{N}{n} \sum_{i=1}^{n} \log p(\mathbf{x}_{\pi_i} \,|\, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})$$

where $(\pi_1, \cdots, \pi_N)$ is a random permutation of $(1, \cdots, N)$.

Basic setup

1. SG-MCMC relies on the following quantity (stochastic gradient):

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}) \triangleq -\frac{N}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{\pi_i} \,|\, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) \ ,$$

2. $\nabla_\theta \tilde{U}(\theta)$ is an unbiased estimate of $\nabla_\theta U(\theta)$:
   - SG-MCMC samples parameters based on $\nabla_\theta \tilde{U}(\theta)$
   - very cheap to compute
   - bringing the name "stochastic gradient MCMC"

Basic setup

1. SG-MCMC relies on the following quantity (stochastic gradient):

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}) \triangleq -\frac{N}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{\pi_i} \,|\, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) \ ,$$

2. $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta})$ is an unbiased estimate of $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$:
   - SG-MCMC samples parameters based on $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta})$
   - very cheap to compute
   - bringing the name "stochastic gradient MCMC"

Some facts about SG-MCMC

1. By ignoring the acceptance step:
   - the detailed balance condition typically not hold, and the algorithm is not reversible[9]
   - typically leads to biased, but controllable estimations
2. By using sub-data in each iteration:
   - yielding stochastic gradients
   - does not affect the convergence properties (*e.g.*, convergence rates), compared to using the whole data in each iteration

[9] These are not necessary conditions for a valid MCMC method.

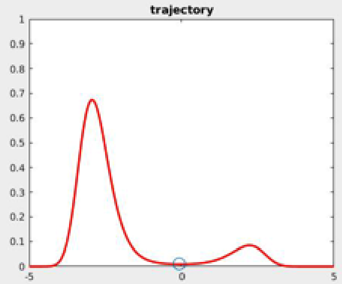
Some facts about SG-MCMC

1. By ignoring the acceptance step:
    - the detailed balance condition typically not hold, and the algorithm is not reversible[9]
    - typically leads to biased, but controllable estimations
2. By using sub-data in each iteration:
    - yielding stochastic gradients
    - does not affect the convergence properties (*e.g.*, convergence rates), compared to using the whole data in each iteration

[9]These are not necessary conditions for a valid MCMC method.

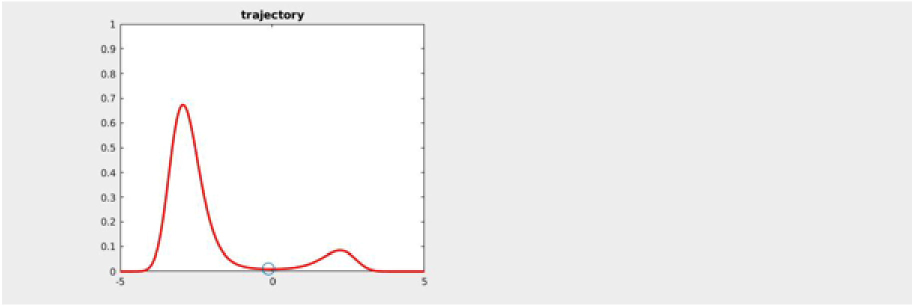
Demo: the two key steps

1. Proposals follow stochastic gradients of log-posteriors:
   - stuck in a local mode



Demo: the two key steps

1. After adding random Gaussian noise:
   - it works !!



Outline

Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
   - Stochastic Gradient Langevin Dynamics (SGLD)
   - Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)
   - Stochastic Gradient Thermostats (SGNHT)
   - Stochastic Gradient MCMC with Riemannian Geometry
     - stochastic gradient Riemannian Langevin dynamics (SGRLD)
     - preconditioned stochastic gradient Langevin dynamics (PSGLD)
2. Theory

Outline

Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
   - Stochastic Gradient Langevin Dynamics (SGLD)
   - Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)
   - Stochastic Gradient Thermostats (SGNHT)
   - Stochastic Gradient MCMC with Riemannian Geometry
     - stochastic gradient Riemannian Langevin dynamics (SGRLD)
     - preconditioned stochastic gradient Langevin dynamics (PSGLD)
2. Theory

First attempt

1. A 1st-order method: directly update on the model parameter $\boldsymbol{\theta}$.
2. Use a proposal that follows the stochastic gradient of the log-posterior:

$$\boldsymbol{\theta}_{l+1} = \boldsymbol{\theta}_l - h_{l+1} \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l)$$

   - h_l's are the stepsizes, could be fixed ($\forall l, h_l = h$) or deceasing ($\forall l, h_l > h_{l+1}$)

3. Ignore the acceptance step.
4. Resulting in Stochastic Gradient Descend (SGD).

First attempt

1. A 1st-order method: directly update on the model parameter $\boldsymbol{\theta}$.
2. Use a proposal that follows the stochastic gradient of the log-posterior:

$$\boldsymbol{\theta}_{l+1} = \boldsymbol{\theta}_l - h_{l+1} \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l)$$

   - h_l's are the stepsizes, could be fixed ($\forall l, h_l = h$) or deceasing ($\forall l, h_l > h_{l+1}$)
3. Ignore the acceptance step.
4. Resulting in Stochastic Gradient Descend (SGD).

Random noise to the rescue

1. Need to make the algorithm explore the parameter space:
   - adding random Gaussian noise to the update[10]

$$\begin{aligned} \boldsymbol{\theta}_{l+1} &= \boldsymbol{\theta}_l - h_{l+1} \nabla_{\boldsymbol{\theta}} \tilde{U}(\theta_l) + \sqrt{2h_{l+1}} \zeta_{l+1} \\ \zeta_{l+1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned}$$

2. The magnitude of the Gaussian needs to be $\sqrt{2h_{l+1}}$ in order to guarantee a correct sampler:
   - reasons to be explained later
3. This is called stochastic gradient Langevin dynamics (SGLD).

[10] In the following, we will directly use $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to represent a normal random variable with zero-mean and covariance matrix $\mathbf{I}$.

SGLD in algorithm

```
Input: Parameters $\{h_l\}$
Output: Approximate samples $\{\boldsymbol{\theta}_l\}$

Initialize $\boldsymbol{\theta}_0 \in \mathbb{R}^n$
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_{l-1}) h_l + \sqrt{2h_l}\mathcal{N}(\mathbf{0}, \mathbf{I})$
end
Return $\{\boldsymbol{\theta}_l\}$
```

Algorithm 1: Stochastic Gradient Langevin Dynamics

Example[11]

1. A simple Gaussian mixture:

$$\theta_1 \sim \mathcal{N}(0, 10), \quad \boldsymbol{\theta}_2 \sim \mathcal{N}(0, 1)$$
$$x_i \sim \frac{1}{2}\mathcal{N}(\boldsymbol{\theta}_1, 2) + \frac{1}{2}\mathcal{N}(\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2, 2), \quad i = 1, \cdots, 100$$

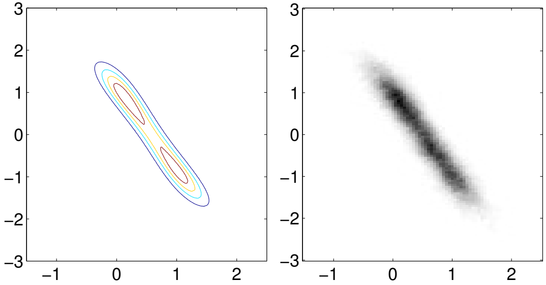


Figure: Left: true posterior; Right: sample-based estimation.

[11] M. Welling and Y. W. Teh. "Bayesian learning via stochastic gradient Langevin dynamics". In: *ICML*. 2011.

Outline

Stochastic Gradient Markov Chain Monte Carlo

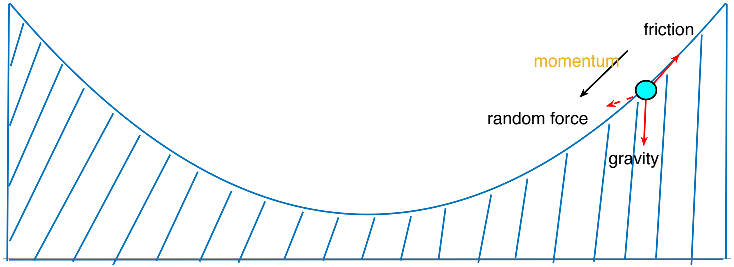
1. SG-MCMC algorithms
   - Stochastic Gradient Langevin Dynamics (SGLD)
   - Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)
   - Stochastic Gradient Thermostats (SGNHT)
   - Stochastic Gradient MCMC with Riemannian Geometry
     - stochastic gradient Riemannian Langevin dynamics (SGRLD)
     - preconditioned stochastic gradient Langevin dynamics (PSGLD)
2. Theory

SGHMC

1. SGLD is slow when parameter space exhibits uneven curvatures.
2. Use the momentum idea to improve SGLD:
   - a generalization of the HMC, in that the ball is rolling on a friction surface
   - the ball follows the momentum instead of gradients, which is a summarization of historical gradients, thus could jump out local modes easier and move faster
   - needs a balance between these extra forces

SGHMC

1. SGLD is slow when parameter space exhibits uneven curvatures.
2. Use the momentum idea to improve SGLD:
   - a generalization of the HMC, in that the ball is rolling on a friction surface
   - the ball follows the momentum instead of gradients, which is a summarization of historical gradients, thus could jump out local modes easier and move faster
   - needs a balance between these extra forces

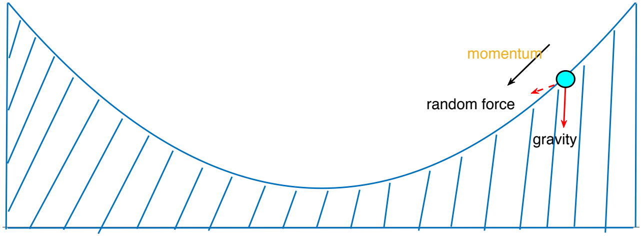


A naive approach to generalize HMC without friction

1. Simply using injected Gaussian noise (random wind) in SGD with momentum.

$$\theta_l = \theta_{l-1} + \mathbf{p}\, h_l$$
$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_{\theta} \tilde{U}(\theta_l) h_l + \underbrace{\sqrt{2h_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})}_{\text{random wind}}$$

2. Would not work:
   - random wind tends to uniformize the location distribution[12]
   - the probability of see the ball at any location is equal



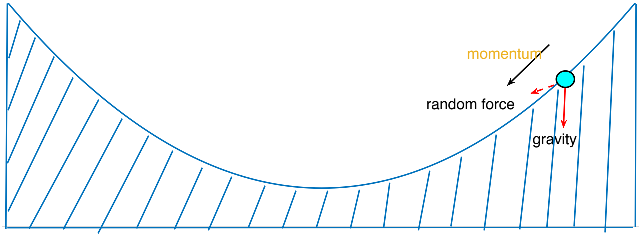
[12] T. Chen, E. B. Fox, and C. Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo". In: *ICML*. 2014.

A naive approach to generalize HMC without friction

1. Simply using injected Gaussian noise (random wind) in SGD with momentum.

$$\begin{aligned} \boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\ \mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l + \underbrace{\sqrt{2h_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})}_{\text{random wind}} \end{aligned}$$

2. Would not work:
   - random wind tends to uniformize the location distribution[12]
   - the probability of see the ball at any location is equal



[12] T. Chen, E. B. Fox, and C. Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo". In: *ICML*. 2014.

Adding a friction term

1. Without a friction term, the random Gaussian noise would drive the ball too far away from their stationary distribution.
2. After adding a friction term:

$$\theta_l = \theta_{l-1} + \mathbf{p}\, h_l$$
$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_\theta \tilde{U}(\theta_l) h_l - A\mathbf{p}\, h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I}) \; ,$$

   where $A > 0$ is a constant[13], controlling the magnitude of the friction.
3. The fraction term penalize the momentum:
   - the more momentum, the more fraction it has, thus slowing down the ball

[13] In the original SGHMC paper, A is decomposed into a known variance of injected noise and an unknown variance of stochastic gradients.

Adding a friction term

1. Without a friction term, the random Gaussian noise would drive the ball too far away from their stationary distribution.
2. After adding a friction term:

$$\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l$$
$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - A\, \mathbf{p}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \ ,$$

   where $A > 0$ is a constant[13], controlling the magnitude of the friction.
3. The fraction term penalize the momentum:
   - the more momentum, the more fraction it has, thus slowing down the ball

[13] In the original SGHMC paper, A is decomposed into a known variance of injected noise and an unknown variance of stochastic gradients.

Adding a friction term

1. Without a friction term, the random Gaussian noise would drive the ball too far away from their stationary distribution.
2. After adding a friction term:

$$\begin{aligned}\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\ \mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - A\, \mathbf{p}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})\ ,\end{aligned}$$

   where $A > 0$ is a constant[13], controlling the magnitude of the friction.
3. The fraction term penalize the momentum:
   - the more momentum, the more fraction it has, thus slowing down the ball

[13] In the original SGHMC paper, A is decomposed into a known variance of injected noise and an unknown variance of stochastic gradients.

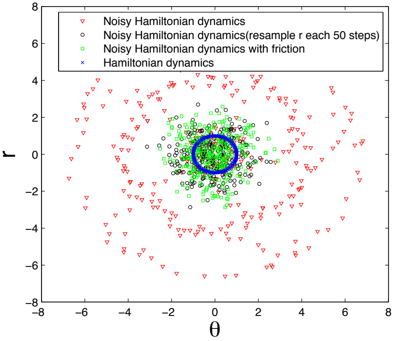
SGHMC in algorithm

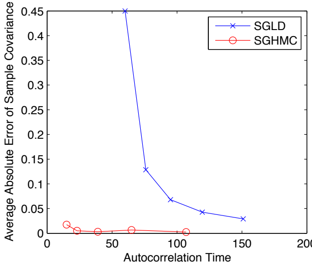
```
Input: Parameters $A, \{h_l\}$
Output: Approximate samples $\{\boldsymbol{\theta}_l\}$

Initialize $\boldsymbol{\theta}_0 \in \mathbb{R}^n$
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l$
    $\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_l) h_l - A\, \mathbf{p}_{l-1}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
end
Return $\{\boldsymbol{\theta}_l\}$
```

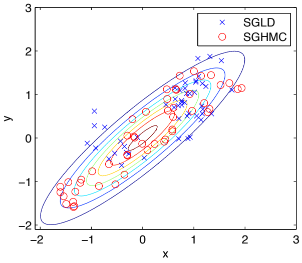
Algorithm 2: Stochastic Gradient Hamiltonian Monte Carlo

Example[14]

1. Sample from a 1D Gaussian distribution:
   - $U(\theta) = \frac{1}{2}\,\theta^2$
2. Sample from a 2D Gaussian distribution:
   - $U(\theta) = \frac{1}{2}\,\theta^T\,\Sigma^{-1}\,\theta$







[14] T. Chen, E. B. Fox, and C. Guestrin. “Stochastic Gradient Hamiltonian Monte Carlo”. In: *ICML*. 2014.

Outline

Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
   - Stochastic Gradient Langevin Dynamics (SGLD)
   - Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)
   - Stochastic Gradient Thermostats (SGNHT)
   - Stochastic Gradient MCMC with Riemannian Geometry
     - stochastic gradient Riemannian Langevin dynamics (SGRLD)
     - preconditioned stochastic gradient Langevin dynamics (PSGLD)
2. Theory

Stochastic gradient Nośe-Hoover thermostats

1. Revisit SGHMC:

$$\begin{aligned}\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\ \mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - A\,\mathbf{p}\, h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})\ ,\end{aligned}$$

2. In the existence of stochastic gradient noise, *e.g.*, $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) = \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_l) + \mathcal{N}(\mathbf{0}, B\mathbf{I})$, the update of p:

$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_l) h_l - A\,\mathbf{p}\, h_l + \sqrt{2(A+B)h_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

3. The friction coefficient should be set to $A + B$ instead of A, to correctly sample from true posteriors[15]:
   - B is usually unknown, needs a good estimation
   - could it be learned from the algorithm?

[15] According to the Fokker-Planck equation in stochastic differential equation theory.

Stochastic gradient Nośe-Hoover thermostats

1. Revisit SGHMC:

$$\theta_l = \theta_{l-1} + \mathbf{p}\, h_l$$
$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\theta_l) h_l - A\mathbf{p}\, h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I}) \;,$$

2. In the existence of stochastic gradient noise, *e.g.*, $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) = \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_l) + \mathcal{N}(\mathbf{0}, B\mathbf{I})$, the update of p:

$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} U(\theta_l) h_l - A\mathbf{p}\, h_l + \sqrt{2(A+B)h_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

3. The friction coefficient should be set to $A + B$ instead of A, to correctly sample from true posteriors[15]:
   - B is usually unknown, needs a good estimation
   - could it be learned from the algorithm?

[15] According to the Fokker-Planck equation in stochastic differential equation theory.

Stochastic gradient Nośe-Hoover thermostats

1. Revisit SGHMC:

$$\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l$$
$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - A\mathbf{p}\, h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I}) \; ,$$

2. In the existence of stochastic gradient noise, *e.g.*, $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) = \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_l) + \mathcal{N}(\mathbf{0}, B\mathbf{I})$, the update of p:

$$\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_l) h_l - A\mathbf{p}\, h_l + \sqrt{2(A+B)h_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

3. The friction coefficient should be set to $A + B$ instead of A, to correctly sample from true posteriors[15]:
   - B is usually unknown, needs a good estimation
   - could it be learned from the algorithm?

[15] According to the Fokker-Planck equation in stochastic differential equation theory.

Stochastic gradient Nośe-Hoover thermostats

1. How to adaptively learn the noise coefficient B?
2. Use the Nośe-Hoover thermostat:
   - a physical system (*e.g.*, rolling ball) embedded in a heat bath for energy exchange
   - when the system temperature is high, the heat bath absorbs heat/energy by increasing the friction, thus slows down the movement
   - when the system temperature is low, the heat bath releases heat/energy by decreasing the friction, thus speeds up the movement
   - the energy absorbing/releasing keeps the system steady (sampling from the true posterior distribution)

Stochastic gradient Nośe-Hoover thermostats

1. How to adaptively learn the noise coefficient B?
2. Use the Nośe-Hoover thermostat:
   - a physical system (*e.g.*, rolling ball) embedded in a heat bath for energy exchange
   - when the system temperature is high, the heat bath absorbs heat/energy by increasing the friction, thus slows down the movement
   - when the system temperature is low, the heat bath releases heat/energy by decreasing the friction, thus speeds up the movement
   - the energy absorbing/releasing keeps the system steady (sampling from the true posterior distribution)

Stochastic gradient Nośe-Hoover thermostats

1. How to adaptively learn the noise coefficient B?
2. Use the Nośe-Hoover thermostat:
   - a physical system (*e.g.*, rolling ball) embedded in a heat bath for energy exchange
   - when the system temperature is high, the heat bath absorbs heat/energy by increasing the friction, thus slows down the movement
   - when the system temperature is low, the heat bath releases heat/energy by decreasing the friction, thus speeds up the movement
   - the energy absorbing/releasing keeps the system steady (sampling from the true posterior distribution)

Stochastic gradient Nośe-Hoover thermostats

1. How to adaptively learn the noise coefficient B?
2. Use the Nośe-Hoover thermostat:
    - a physical system (*e.g.*, rolling ball) embedded in a heat bath for energy exchange
    - when the system temperature is high, the heat bath absorbs heat/energy by increasing the friction, thus slows down the movement
    - when the system temperature is low, the heat bath releases heat/energy by decreasing the friction, thus speeds up the movement
    - the energy absorbing/releasing keeps the system steady (sampling from the true posterior distribution)

Stochastic gradient Nośe-Hoover thermostats

1. How to adaptively learn the noise coefficient B?
2. Use the Nośe-Hoover thermostat:
   - a physical system (*e.g.*, rolling ball) embedded in a heat bath for energy exchange
   - when the system temperature is high, the heat bath absorbs heat/energy by increasing the friction, thus slows down the movement
   - when the system temperature is low, the heat bath releases heat/energy by decreasing the friction, thus speeds up the movement
   - the energy absorbing/releasing keeps the system steady (sampling from the true posterior distribution)

A little bit of statistical physics

1. Statistical physics describes the probability of states $(\boldsymbol{\theta}, \mathbf{p})$ of a system in thermal equilibrium with a heat bath at temperature T.
2. The probability follows the canonical distribution:

$$\rho(\theta, \mathbf{p}) \propto \exp\left(-H(\theta, \mathbf{p}) / (k_B T)\right) \triangleq \exp\left(-\frac{E(\theta, \mathbf{p}) + K(\mathbf{p})}{k_B T}\right) ,$$

where k_B is the Boltzmann constant, $E(\theta, \mathbf{p})$ the potential energy, $K(\mathbf{p})$ the kinetic energy.
3. Thermal equilibrium condition:

$$k_B T/2 = \mathbb{E}\left[K(\mathbf{p})\right]/D \rightarrow k_B T = \mathbb{E}\left[\mathbf{p}^T \mathbf{p}\right]/D$$

A little bit of statistical physics

1. Statistical physics describes the probability of states $(\boldsymbol{\theta}, \mathbf{p})$ of a system in thermal equilibrium with a heat bath at temperature T.
2. The probability follows the canonical distribution:

$$\rho(\boldsymbol{\theta}, \mathbf{p}) \propto \exp\left(-H(\boldsymbol{\theta}, \mathbf{p})/\left(k_B T\right)\right) \triangleq \exp\left(-\frac{E(\boldsymbol{\theta}, \mathbf{p}) + K(\mathbf{p})}{k_B T}\right) ,$$

   where k_B is the Boltzmann constant, $E(\boldsymbol{\theta}, \mathbf{p})$ the potential energy, $K(\mathbf{p})$ the kinetic energy.
3. Thermal equilibrium condition:

$$k_B T/2 = \mathbb{E}\left[K(\mathbf{p})\right]/D \rightarrow k_B T = \mathbb{E}\left[\mathbf{p}^T \mathbf{p}\right]/D$$

A little bit of statistical physics

1. Statistical physics describes the probability of states $(\boldsymbol{\theta}, \mathbf{p})$ of a system in thermal equilibrium with a heat bath at temperature T.
2. The probability follows the canonical distribution:

$$\rho(\boldsymbol{\theta}, \mathbf{p}) \propto \exp\left(-H(\boldsymbol{\theta}, \mathbf{p}) / \left(k_B T\right)\right) \triangleq \exp\left(-\frac{E(\boldsymbol{\theta}, \mathbf{p}) + K(\mathbf{p})}{k_B T}\right) ,$$

   where k_B is the Boltzmann constant, $E(\boldsymbol{\theta}, \mathbf{p})$ the potential energy, $K(\mathbf{p})$ the kinetic energy.
3. Thermal equilibrium condition:

$$k_B T/2 = \mathbb{E}\left[K(\mathbf{p})\right]/D \to k_B T = \mathbb{E}\left[\mathbf{p}^T \mathbf{p}\right]/D$$

SGNHT

1. In Bayesian setting, the equilibrium distribution $\rho(\boldsymbol{\theta}, \mathbf{p}) \propto \exp\left(-H(\boldsymbol{\theta}, \mathbf{p})\right)$, thus $k_B T = 1$

$$\mathbb{E}\left[\mathbf{p}^T \mathbf{p}\right] / D = k_B T = 1$$

2. In SGHMC with stochastic gradients $\nabla_{\theta} \tilde{U}(\theta, \mathbf{p})$:
   - the dynamic may drift away from thermal equilibrium if stochastic gradients exibit too much noise
   - need to adaptively control the friction
   - idea is to replace the friction coefficient A in SGHMC with a thermostat variable ξ, which is adaptively estimated using thermal equilibrium condition

$$\begin{aligned}
\theta_l &= \theta_{l-1} + \mathbf{p}\, h_l \\
\mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\theta} \tilde{U}(\theta_l) h_l - \xi_{l-1}\, \mathbf{p}\, h_l + \sqrt{2 A h_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
\xi_l &= \xi_{l-1} + \left(\mathbf{p}_l^T \mathbf{p}_l / D - 1\right) h_l
\end{aligned}$$

SGNHT

1. In Bayesian setting, the equilibrium distribution $\rho(\boldsymbol{\theta}, \mathbf{p}) \propto \exp(-H(\boldsymbol{\theta}, \mathbf{p}))$, thus $k_B T = 1$

$$\mathbb{E}\left[\mathbf{p}^T \mathbf{p}\right] / D = k_B T = 1$$

2. In SGHMC with stochastic gradients $\nabla_{\boldsymbol{\theta}} \tilde{U}(\theta, \mathbf{p})$:
   - the dynamic may drift away from thermal equilibrium if stochastic gradients exibit too much noise
   - need to adaptively control the friction
   - idea is to replace the friction coefficient A in SGHMC with a thermostat variable ξ, which is adaptively estimated using thermal equilibrium condition

$$\begin{aligned}
\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\
\mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - \xi_{l-1}\, \mathbf{p}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
\xi_l &= \xi_{l-1} + \left(\mathbf{p}_l^T \mathbf{p}_l / D - 1\right) h_l
\end{aligned}$$

SGNHT

$$\begin{aligned}
\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\
\mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - \xi_{l-1}\, \mathbf{p}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
\xi_l &= \xi_{l-1} + \left(\mathbf{p}_l^T\, \mathbf{p}_l / D - 1 \right) h_l
\end{aligned}$$

1. If the kinetic energy is higher than $1/2$ (high temperature), ξ gets bigger, friction gets bigger, momentum **p** gets lower, vice versa.
2. The equilibrium is reached when $\mathbb{E}\left[\mathbf{p}_l^T\, \mathbf{p}_l\right] / D = 1$:
   - exactly the thermal equilibrium condition
3. Samples generated from the true posterior distribution.

SGNHT

$$
\begin{aligned}
\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\
\mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - \xi_{l-1}\, \mathbf{p}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
\xi_l &= \xi_{l-1} + \left(\mathbf{p}_l^T \mathbf{p}_l / D - 1 \right) h_l
\end{aligned}
$$

1. If the kinetic energy is higher than $1/2$ (high temperature), ξ gets bigger, friction gets bigger, momentum **p** gets lower, vice versa.
2. The equilibrium is reached when $\mathbb{E}\left[\mathbf{p}_l^T \mathbf{p}_l\right] / D = 1$:
   - exactly the thermal equilibrium condition
3. Samples generated from the true posterior distribution.

SGNHT

$$
\begin{aligned}
\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\
\mathbf{p}_l &= \mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) h_l - \xi_{l-1}\, \mathbf{p}\, h_l + \sqrt{2 A h_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
\xi_l &= \xi_{l-1} + \left(\mathbf{p}_l^T\, \mathbf{p}_l \,/ D - 1\right) h_l
\end{aligned}
$$

1. If the kinetic energy is higher than $1/2$ (high temperature), ξ gets bigger, friction gets bigger, momentum **p** gets lower, vice versa.
2. The equilibrium is reached when $\mathbb{E}\left[\mathbf{p}_l^T\, \mathbf{p}_l\right] / D = 1$:
   - exactly the thermal equilibrium condition
3. Samples generated from the true posterior distribution.

SGNHT in algorithm

```
**Input**: Parameters $A, \{h_l\}$
**Output**: Approximate samples $\{\boldsymbol{\theta}_l\}$

Initialize $\boldsymbol{\theta}_0 \in \mathbb{R}^n$
**for** $l = 1, 2, \ldots$ **do**
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l$
    $\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_l) h_l - \xi_{l-1}\, \mathbf{p}_{l-1}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\xi_l = \xi_{l-1} + \left(\mathbf{p}^T \mathbf{p} / D - 1\right) h_l$
**end**
Return $\{\boldsymbol{\theta}_l\}$
```

Algorithm 3: Stochastic Gradient Nośe-Hoover Thermostat

Comparison: SGHMC vs. SGNHT[16]

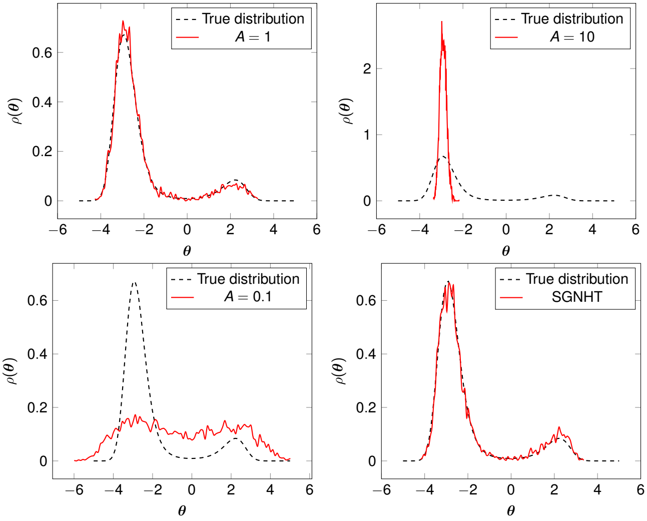


Figure: SGHMC with $A = 1$, $A = 10$, $A = 0.1$, and the SGNHT.

[16] N. Ding et al. “Bayesian Sampling Using Stochastic Gradient Thermostats”. In: *NIPS*. 2014.

Relation wih stochastic optimization

1. SG-MCMC is essentially stochastic optimzation with appropriate injected noise:
   - large noise tends to make samples uniform, small noise tends to stuck algorithms on local modes
2. SGLD vs. SGD.
3. SGHMC vs. SGD with momentum.
4. No traditional stochastic optimization counterpart for SGNHT yet[17].

[17] Some new algorithm such as Santa could be considered as the counterpart, discussed later.

Relation wih stochastic optimization

1. SG-MCMC is essentially stochastic optimzation with appropriate injected noise:
   - large noise tends to make samples uniform, small noise tends to stuck algorithms on local modes
2. SGLD vs. SGD.
3. SGHMC vs. SGD with momentum.
4. No traditional stochastic optimization counterpart for SGNHT yet[17].

[17] Some new algorithm such as Santa could be considered as the counterpart, discussed later.

SGLD vs. SGD

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1}) \triangleq -\frac{N}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{\pi_i} \mid \boldsymbol{\theta}_{l-1}) - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_{l-1}) ,$$

```
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_{l-1}) h_l + \sqrt{2h_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
end
```

SGHMC vs. SGD-M

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) \triangleq -\frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{\pi_i} \,|\, \boldsymbol{\theta}_l) - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_l) \ ,$$

```
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the
    $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l$
    $\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_l) h_l -$
    $A\, \mathbf{p}_{l-1}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
end
```

SGHMC vs. SGD-M

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) \triangleq -\frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{\pi_i} \,|\, \boldsymbol{\theta}_l) - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_l) ,$$

```
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the
    $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l$
    $\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_l) h_l -$
    $A\,\mathbf{p}_{l-1}\, h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$
end
```

- Reparametrization: $\epsilon = h^2$, $m = Ah$, $\mathbf{v} = \mathbf{p}\, h$

SGHMC vs. SGD-M

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_l) \triangleq -\frac{N}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{\pi_i} \mid \boldsymbol{\theta}_l) - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_l) \ ,$$

```
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l$
    $\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_l) h_l - A\, \mathbf{p}_{l-1}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
end
```

```
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{l-1})$ from the $l$-th minibatch
    $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} + \mathbf{v}_{l-1}$
    $\mathbf{v}_l = (1 - m)\, \mathbf{v}_{l-1} - \nabla \tilde{U}(\boldsymbol{\theta}_l) \epsilon_l + \sqrt{2m\epsilon_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
end
```

- Reparametrization: $\epsilon = h^2$, $m = Ah$, $\mathbf{v} = \mathbf{p}\, h$

SGHMC vs. SGD-M

$$\nabla_{\theta} \tilde{U}(\theta_l) \triangleq -\frac{N}{n} \sum_{i=1}^{n} \nabla_{\theta} \log p(\mathbf{x}_{\pi_i} \mid \theta_l) - \nabla_{\theta} \log p(\theta_l) ,$$

```
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\theta} \tilde{U}(\theta_{l-1})$ from the $l$-th minibatch
    $\theta_l = \theta_{l-1} + \mathbf{p}\, h_l$
    $\mathbf{p}_l = \mathbf{p}_{l-1} - \nabla \tilde{U}(\theta_l) h_l - A\, \mathbf{p}_{l-1}\, h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
end
```

```
for $l = 1, 2, \ldots$ do
    Evaluate $\nabla_{\theta} \tilde{U}(\theta_{l-1})$ from the $l$-th minibatch
    $\theta_l = \theta_{l-1} + \mathbf{v}_{l-1}$
    $\mathbf{v}_l = (1 - m)\, \mathbf{v}_{l-1} - \nabla \tilde{U}(\theta_l) \epsilon_l + \sqrt{2m\epsilon_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
end
```

- Reparametrization: $\epsilon = h^2$, $m = Ah$, $\mathbf{v} = \mathbf{p}\, h$
- ϵ: learning rate; m: momentum weight

Outline

Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
   - Stochastic Gradient Langevin Dynamics (SGLD)
   - Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)
   - Stochastic Gradient Thermostats (SGNHT)
   - Stochastic Gradient MCMC with Riemannian Geometry
     - stochastic gradient Riemannian Langevin dynamics (SGRLD)
     - preconditioned stochastic gradient Langevin dynamics (PSGLD)
2. Theory

Outline

Stochastic Gradient Markov Chain Monte Carlo

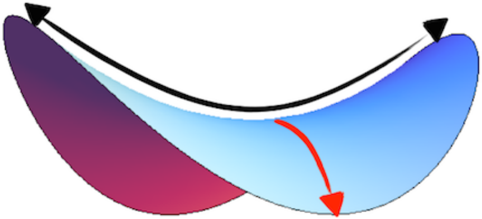
1. SG-MCMC algorithms
   - Stochastic Gradient Langevin Dynamics (SGLD)
   - Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)
   - Stochastic Gradient Thermostats (SGNHT)
   - Stochastic Gradient MCMC with Riemannian Geometry
     - stochastic gradient Riemannian Langevin dynamics (SGRLD)
     - preconditioned stochastic gradient Langevin dynamics (PSGLD)
2. Theory

Manifold information geometry

1. Higher-order gradient information have proven helpful in training high-dimensional, complex optimization problems, *e.g.*, deep learning:
   - quasi-Newton methods
   - rescale parameters so that the loss function has similar curvature along all directions: Adagrad, Adadelta, Adamand RMSprop algorithms
   - approximation to using Riemannian information geometry
2. Geometry information is encoded with a Riemannian metric $G(\theta)$:
   - reflects the curvature property, *e.g.*, inner product of two vectors $\mathbf{v}$ and $\mathbf{w}$ on a tangent space is $\mathbf{v}^T G(\theta) \mathbf{w}^T$

Manifold information geometry

1. Higher-order gradient information have proven helpful in training high-dimensional, complex optimization problems, *e.g.*, deep learning:
   - quasi-Newton methods
   - rescale parameters so that the loss function has similar curvature along all directions: Adagrad, Adadelta, Adamand RMSprop algorithms
   - approximation to using Riemannian information geometry
2. Geometry information is encoded with a Riemannian metric $G(\theta)$:
   - reflects the curvature property, *e.g.*, inner product of two vectors $\mathbf{v}$ and $\mathbf{w}$ on a tangent space is $\mathbf{v}^T G(\theta) \mathbf{w}^T$



Stochastic gradient Riemannian Langevin dynamics

1. Adding Riemannian information geometry into SGLD:

$$\theta_{l+1} = \theta_l - h_{l+1}\left(G(\theta_l)\nabla_\theta \tilde{U}(\theta_l) + \Gamma(\theta_l)\right) + \sqrt{2h_{l+1}G(\theta_l)}\zeta_{l+1}$$

   - $G(\theta)$: Riemannian metric, sometimes refer to as preconditioner
   - $\Gamma_i(\theta) \triangleq \sum_j \frac{\partial G_{ij}(\theta)}{\partial \theta_j}$: change of manifold curvature
   - In SGLD, $G(\theta) = \mathbf{I}$, $\Gamma(\theta) = \mathbf{0}$
2. SGRLD for LDA[18] is a good example of SGRLD.
3. Imposing Riemannian geometry into other SG-MCMC algorithms follows similarly.
4. Challenge: $G(\theta)$ is usually intractable:
   - need a computational efficient way to approximate $G(\theta)$

[18] S. Patterson and Y. W. Teh. "Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex". In: *NIPS*. 2013.

Stochastic gradient Riemannian Langevin dynamics

1. Adding Riemannian information geometry into SGLD:

$$\boldsymbol{\theta}_{l+1} = \boldsymbol{\theta}_l - h_{l+1}\left(G(\boldsymbol{\theta}_l)\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta}_l) + \Gamma(\boldsymbol{\theta}_l)\right) + \sqrt{2h_{l+1}G(\boldsymbol{\theta}_l)}\zeta_{l+1}$$

   - $G(\boldsymbol{\theta})$: Riemannian metric, sometimes refer to as preconditioner
   - $\Gamma_i(\boldsymbol{\theta}) \triangleq \sum_j \frac{\partial G_{ij}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j}$: change of manifold curvature
   - In SGLD, $G(\boldsymbol{\theta}) = \mathbf{I}$, $\Gamma(\boldsymbol{\theta}) = \mathbf{0}$
2. SGRLD for LDA[18] is a good example of SGRLD.
3. Imposing Riemannian geometry into other SG-MCMC algorithms follows similarly.
4. Challenge: $G(\theta)$ is usually intractable:
   - need a computational efficient way to approximate $G(\theta)$

[18] S. Patterson and Y. W. Teh. "Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex". In: *NIPS*. 2013.

Stochastic gradient Riemannian Langevin dynamics

1. Adding Riemannian information geometry into SGLD:

$$\boldsymbol{\theta}_{l+1} = \boldsymbol{\theta}_l - h_{l+1}\left(G(\boldsymbol{\theta}_l)\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta}_l) + \Gamma(\boldsymbol{\theta}_l)\right) + \sqrt{2h_{l+1}G(\boldsymbol{\theta}_l)}\zeta_{l+1}$$

   - $G(\boldsymbol{\theta})$: Riemannian metric, sometimes refer to as preconditioner
   - $\Gamma_i(\boldsymbol{\theta}) \triangleq \sum_j \frac{\partial G_{ij}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j}$: change of manifold curvature
   - In SGLD, $G(\boldsymbol{\theta}) = \mathbf{I}$, $\Gamma(\boldsymbol{\theta}) = \mathbf{0}$
2. SGRLD for LDA[18] is a good example of SGRLD.
3. Imposing Riemannian geometry into other SG-MCMC algorithms follows similarly.
4. Challenge: $G(\boldsymbol{\theta})$ is usually intractable:
   - need a computational efficient way to approximate $G(\boldsymbol{\theta})$

[18] S. Patterson and Y. W. Teh. “Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex”. In: *NIPS*. 2013.

Outline

Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
   - Stochastic Gradient Langevin Dynamics (SGLD)
   - Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)
   - Stochastic Gradient Thermostats (SGNHT)
   - Stochastic Gradient MCMC with Riemannian Geometry
     - stochastic gradient Riemannian Langevin dynamics (SGRLD)
     - preconditioned stochastic gradient Langevin dynamics (PSGLD)
2. Theory

Preconditioned stochastic gradient Langevin dynamics

1. RMSprop as the Preconditioner (Riemannian metric).
2. $\bar{g}(\theta_l) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p(\mathbf{d}_{\pi_i} \mid \theta_l)$: sample mean of gradient.
3. Preconditioner construction:

$$V(\theta_{l+1}) = \alpha V(\theta_l) + (1-\alpha)\bar{g}(\theta_l) \odot \bar{g}(\theta_l)$$
$$G(\theta_{l+1}) = \text{diag}\left(1 \oslash \left(\lambda + \sqrt{V(\theta_{l+1})}\right)\right)$$

4. Intuitive interpretations:
   - the preconditioner equalizes the gradient so that a constant stepsize is adequate for all dimensions
   - the stepsizes are adaptive, in that flat directions have larger stepsizes while curved directions have smaller stepsizes

Preconditioned stochastic gradient Langevin dynamics

1. RMSprop as the Preconditioner (Riemannian metric).
2. $\bar{g}(\boldsymbol{\theta}_l) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{d}_{\pi_i} \mid \boldsymbol{\theta}_l)$: sample mean of gradient.
3. Preconditioner construction:

$$V(\boldsymbol{\theta}_{l+1}) = \alpha V(\boldsymbol{\theta}_l) + (1-\alpha)\bar{g}(\boldsymbol{\theta}_l) \odot \bar{g}(\boldsymbol{\theta}_l)$$
$$G(\boldsymbol{\theta}_{l+1}) = \mathrm{diag}\left(1 \oslash \left(\lambda + \sqrt{V(\boldsymbol{\theta}_{l+1})}\right)\right)$$

4. Intuitive interpretations:
   - the preconditioner equalizes the gradient so that a constant stepsize is adequate for all dimensions
   - the stepsizes are adaptive, in that flat directions have larger stepsizes while curved directions have smaller stepsizes

Preconditioned stochastic gradient Langevin dynamics

1. RMSprop as the Preconditioner (Riemannian metric).
2. $\bar{g}(\boldsymbol{\theta}_l) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{d}_{\pi_i} \,|\, \boldsymbol{\theta}_l)$: sample mean of gradient.
3. Preconditioner construction:

$$V(\boldsymbol{\theta}_{l+1}) = \alpha V(\boldsymbol{\theta}_l) + (1-\alpha)\bar{g}(\boldsymbol{\theta}_l) \odot \bar{g}(\boldsymbol{\theta}_l)$$
$$G(\boldsymbol{\theta}_{l+1}) = \mathrm{diag}\left(1 \oslash \left(\lambda + \sqrt{V(\boldsymbol{\theta}_{l+1})}\right)\right)$$

4. Intuitive interpretations:
   - the preconditioner equalizes the gradient so that a constant stepsize is adequate for all dimensions
   - the stepsizes are adaptive, in that flat directions have larger stepsizes while curved directions have smaller stepsizes

Preconditioned stochastic gradient Langevin dynamics

1. RMSprop as the Preconditioner (Riemannian metric).
2. $\bar{g}(\boldsymbol{\theta}_l) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{d}_{\pi_i} \,|\, \boldsymbol{\theta}_l)$: sample mean of gradient.
3. Preconditioner construction:

$$V(\boldsymbol{\theta}_{l+1}) = \alpha V(\boldsymbol{\theta}_l) + (1-\alpha)\bar{g}(\boldsymbol{\theta}_l) \odot \bar{g}(\boldsymbol{\theta}_l)$$
$$G(\boldsymbol{\theta}_{l+1}) = \mathrm{diag}\left(1 \oslash \left(\lambda + \sqrt{V(\boldsymbol{\theta}_{l+1})}\right)\right)$$

4. Intuitive interpretations:
   - the preconditioner equalizes the gradient so that a constant stepsize is adequate for all dimensions
   - the stepsizes are adaptive, in that flat directions have larger stepsizes while curved directions have smaller stepsizes

Outline

2 Stochastic Gradient Markov Chain Monte Carlo
   1 SG-MCMC algorithms
   2 Theory

Outline

Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
2. Theory
   - Itô diffusion
   - Convergence theory

Outline

Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
2. Theory
   - Itô diffusion
   - Convergence theory

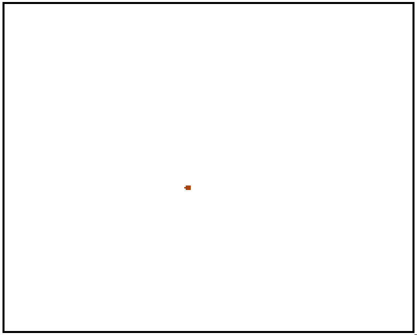
Itô diffusion

1. Itô diffusion is a continuous-time stochastic process, governed by stochastic differential equations of the form:

$$\mathrm{d}\,\mathbf{x}_t = F(\mathbf{x}_t)\mathrm{d}t + \sigma(\mathbf{x}_t)\mathrm{d}\,\mathbf{w}_t$$

- t: time index
- $\mathbf{x}_t$: model states, typically includes $\boldsymbol{\theta}$
- $\mathbf{w}_t$: standard Brownian motion, *e.g.*, $\forall t, \Delta h > 0$, $\Delta\,\mathbf{w}_t \triangleq \mathbf{w}_{t+\Delta h} - \mathbf{w}_t$ are zero-mean Gaussian random variables with standard deviation Δh
- $F(\mathbf{x}_t)$: drift coefficient
- $\sigma(\mathbf{x}_t)$: diffusion coefficient

Brownian motion (Jean Giraud code



Itô diffusion

1. Itô diffusion typically endows an invariant measure, *i.e.*, the probability distribution of $\mathbf{x}_t$, $\forall t$ (time invariant).
2. Ornstein-Uhlenbeck (OU) process:

$$\mathrm{d}x_t = \underbrace{\beta\left(\mu - x_t\right)}_{F(x_t)}\mathrm{d}t + \underbrace{\alpha}_{\sigma(x_t)}\mathrm{d}w_t, \quad \beta, \alpha > 0$$

- invariant measure: $\rho(x) = \sqrt{\frac{\beta}{\pi\alpha^2}} e^{-\beta(x-\mu)^2/\alpha^2}$

Itô diffusion

1. Itô diffusion typically endows an invariant measure, *i.e.*, the probability distribution of $\mathbf{x}_t$, $\forall t$ (time invariant).
2. Ornstein-Uhlenbeck (OU) process:

$$\mathrm{d}x_t = \underbrace{\beta\left(\mu - x_t\right)}_{F(x_t)}\mathrm{d}t + \underbrace{\alpha}_{\sigma(x_t)}\mathrm{d}w_t, \quad \beta, \alpha > 0$$

- invariant measure: $\rho(x) = \sqrt{\frac{\beta}{\pi\alpha^2}} e^{-\beta(x-\mu)^2/\alpha^2}$

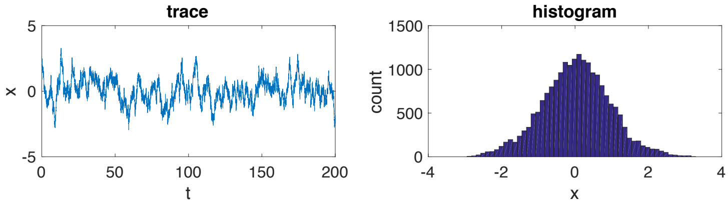


Figure: OU process with $\mu = 0, \beta = 0.5, \alpha = 1$.

Fokker-Planck equation

1. Also known as the Kolmogorov forward equation.
2. It describes the time-evolving probability density function $p(\mathbf{x}, t)$ on the random variable $\mathbf{x}$, driven by the Itô diffusion: $\mathrm{d}\,\mathbf{x}_t = F(\mathbf{x}_t)\mathrm{d}t + \sigma(\mathbf{x}_t)\mathrm{d}\,\mathbf{w}_t$.
3. Let $D_{ij}(\mathbf{x}_t) \triangleq \sum_k \sigma_{ik}(\mathbf{x}_t)\sigma_{jk}(\mathbf{x}_t)$, then $p(\mathbf{x}, t)$ satisfies the Fokker-Planck equation:

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\sum_i \frac{\partial}{\partial \mathbf{x}_i}\left[F_i(\mathbf{x}_t)p(\mathbf{x}, t)\right] + \frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x}, t)\right] .$$

4. In stationary region, $p(\mathbf{x}, t)$ is independent of t, thus $\frac{\partial p(\mathbf{x},t)}{\partial t} = 0$, the Fokker-Planck equation becomes:

$$\sum_i \frac{\partial}{\partial \mathbf{x}_i}\left[F_i(\mathbf{x}_t)p(\mathbf{x})\right] = \frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x})\right] .$$

Fokker-Planck equation

1. Also known as the Kolmogorov forward equation.
2. It describes the time-evolving probability density function $p(\mathbf{x}, t)$ on the random variable $\mathbf{x}$, driven by the Itô diffusion: $\mathrm{d}\,\mathbf{x}_t = F(\mathbf{x}_t)\mathrm{d}t + \sigma(\mathbf{x}_t)\mathrm{d}\,\mathbf{w}_t$.
3. Let $D_{ij}(\mathbf{x}_t) \triangleq \sum_k \sigma_{ik}(\mathbf{x}_t)\sigma_{jk}(\mathbf{x}_t)$, then $p(\mathbf{x}, t)$ satisfies the Fokker-Planck equation:

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\sum_i \frac{\partial}{\partial \mathbf{x}_i}\left[F_i(\mathbf{x}_t)p(\mathbf{x}, t)\right] + \frac{1}{2}\sum_{i,j} \frac{\partial^2}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x}, t)\right] .$$

4. In stationary region, $p(\mathbf{x}, t)$ is independent of t, thus $\frac{\partial p(\mathbf{x},t)}{\partial t} = 0$, the Fokker-Planck equation becomes:

$$\sum_i \frac{\partial}{\partial \mathbf{x}_i}\left[F_i(\mathbf{x}_t)p(\mathbf{x})\right] = \frac{1}{2}\sum_{i,j} \frac{\partial^2}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x})\right] .$$

Fokker-Planck equation

1. Also known as the Kolmogorov forward equation.
2. It describes the time-evolving probability density function $p(\mathbf{x}, t)$ on the random variable $\mathbf{x}$, driven by the Itô diffusion: $\mathrm{d}\,\mathbf{x}_t = F(\mathbf{x}_t)\mathrm{d}t + \sigma(\mathbf{x}_t)\mathrm{d}\,\mathbf{w}_t$.
3. Let $D_{ij}(\mathbf{x}_t) \triangleq \sum_k \sigma_{ik}(\mathbf{x}_t)\sigma_{jk}(\mathbf{x}_t)$, then $p(\mathbf{x}, t)$ satisfies the Fokker-Planck equation:

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\sum_i \frac{\partial}{\partial \mathbf{x}_i}[F_i(\mathbf{x}_t)p(\mathbf{x}, t)] + \frac{1}{2}\sum_{i,j} \frac{\partial^2}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x}, t)\right] .$$

4. In stationary region, $p(\mathbf{x}, t)$ is independent of t, thus $\frac{\partial p(\mathbf{x},t)}{\partial t} = 0$, the Fokker-Planck equation becomes:

$$\sum_i \frac{\partial}{\partial \mathbf{x}_i}[F_i(\mathbf{x}_t)p(\mathbf{x})] = \frac{1}{2}\sum_{i,j} \frac{\partial^2}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x})\right] .$$

Fokker-Planck equation

1. Also known as the Kolmogorov forward equation.
2. It describes the time-evolving probability density function $p(\mathbf{x}, t)$ on the random variable $\mathbf{x}$, driven by the Itô diffusion: $\mathrm{d}\,\mathbf{x}_t = F(\mathbf{x}_t)\mathrm{d}t + \sigma(\mathbf{x}_t)\mathrm{d}\,\mathbf{w}_t$.
3. Let $D_{ij}(\mathbf{x}_t) \triangleq \sum_k \sigma_{ik}(\mathbf{x}_t)\sigma_{jk}(\mathbf{x}_t)$, then $p(\mathbf{x}, t)$ satisfies the Fokker-Planck equation:

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\sum_i \frac{\partial}{\partial\,\mathbf{x}_i}\left[F_i(\mathbf{x}_t)p(\mathbf{x}, t)\right] + \frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial\,\mathbf{x}_i\,\partial\,\mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x}, t)\right] .$$

4. In stationary region, $p(\mathbf{x}, t)$ is independent of t, thus $\frac{\partial p(\mathbf{x}, t)}{\partial t} = 0$, the Fokker-Planck equation becomes:

$$\sum_i \frac{\partial}{\partial\,\mathbf{x}_i}\left[F_i(\mathbf{x}_t)p(\mathbf{x})\right] = \frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial\,\mathbf{x}_i\,\partial\,\mathbf{x}_j}\left[D_{ij}(\mathbf{x}_t)p(\mathbf{x})\right] .$$

Fokker-Planck equation

1. The Fokker-Planck equation is useful in verifying the stationary distribution for some specify Itô diffusions.
2. We can use it to verify that the stationary distribution of the following Itô diffusion is $p(\mathbf{x}) \propto e^{-U(\mathbf{x})}$:

$$\mathrm{d}\,\mathbf{x}_t = -\nabla_{\mathbf{x}} U(\mathbf{x}_t) + \frac{1}{2}\mathrm{d}\,\mathbf{w}_t$$

Diffusion form for SGLD

$$\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} - \nabla_{\boldsymbol{\theta}} \tilde{U}_l(\boldsymbol{\theta}_{l-1}) h_l + \sqrt{2h_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

1. SGLD is based on 1st-order Langevin dynamics, with $\mathbf{x} = \boldsymbol{\theta}$:

$$\mathrm{d}\,\boldsymbol{\theta}_t = \underbrace{-\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_t)}_{F(\mathbf{x}_t)} + \underbrace{\frac{1}{2}\mathbf{I}}_{\sigma(\mathbf{x}_t)}\,\mathrm{d}\,\mathbf{w}_t$$

   - invariant measure: $\rho(\boldsymbol{\theta}) \propto e^{-U(\boldsymbol{\theta})}$

Diffusion form for SGHMC

$$\begin{aligned}\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\ \mathbf{p}_l &= (1 - Ah_l)\,\mathbf{p}_{l-1} - \nabla \tilde{U}_l(\boldsymbol{\theta}_l) h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}$$

1. SGHMC is based on 2nd-order Langevin dynamics, with $\mathbf{x} = \{\boldsymbol{\theta}, \mathbf{p}\}$:

$$\mathrm{d}\begin{pmatrix} \boldsymbol{\theta}_t \\ \mathbf{p}_t \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{p}_t \\ -A\,\mathbf{p}_t - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) \end{pmatrix}}_{F(\mathbf{x}_t)} \mathrm{d}t + \underbrace{\sqrt{2A}\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}}_{\sigma(\mathbf{x}_t)} \mathrm{d}\,\mathbf{w}_t$$

- invariant measure: $\rho(\boldsymbol{\theta}, \mathbf{p}) \propto \exp\left\{-U(\boldsymbol{\theta}) - \frac{\mathbf{p}^T\mathbf{p}}{2}\right\}$

Diffusion form for SGNHT

$$
\begin{aligned}
\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}\, h_l \\
\mathbf{p}_l &= (1 - \xi_{l-1} h_l)\, \mathbf{p}_{l-1} - \nabla \tilde{U}_l(\boldsymbol{\theta}_l) h_l + \sqrt{2Ah_l}\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
\xi_l &= \xi_{l-1} + \left(\mathbf{p}_l^T\, \mathbf{p}_l / D - 1 \right) h_l
\end{aligned}
$$

1. SGNHT is based on the Nosé-Hoover thermostat, with $\mathbf{x} = \{\boldsymbol{\theta}, \mathbf{p}, \xi\}$:

$$
\mathrm{d} \begin{pmatrix} \boldsymbol{\theta}_t \\ \mathbf{p}_t \\ \xi_t \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{p}_t \\ -\xi_t\, \mathbf{p}_t - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_t) \\ \mathbf{p}_t^T\, \mathbf{p}_t / D - 1 \end{pmatrix}}_{F(\mathbf{x}_t)} \mathrm{d}t + \underbrace{\sqrt{2A} \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}}_{\sigma(\mathbf{x}_t)} \mathrm{d}\,\mathbf{w}_t
$$

   - invariant measure: $\rho(\boldsymbol{\theta}, \mathbf{p}, \xi) \propto \exp\left\{ -U(\boldsymbol{\theta}) - \frac{\mathbf{p}^T \mathbf{p}}{2} - \frac{D}{2} (\xi - D)^2 \right\}$

A complete recipe to construct appropriate Itô diffusions

1. Ma *et al.*[19] gave a complete recipe to construct $F(\mathbf{x})$ and $\sigma(\mathbf{x})$:

$$F(\mathbf{x}) = -\left(D(\mathbf{x}) + Q(\mathbf{x})\right)\nabla_{\mathbf{x}} H(\mathbf{x}) + \Gamma(\mathbf{x})$$
$$\sigma(\mathbf{x}) = \sqrt{2D(\mathbf{x})}\,,$$

   - $Q(\mathbf{x})$: a skew-symmetric curl matrix, *e.g.*, $-\mathbf{M} = \mathbf{M}^T$
   - $D(\mathbf{x})$: a positive semidefinite diffusion matrix

2. Any diffusion with the above form endows a marginal invariant measure: $\rho(\theta) \propto e^{-U(\theta)}$.
3. In SGHMC, $D(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A \cdot \mathbf{I} \end{pmatrix}$, $Q(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}$.

[19] Y. A. Ma, T. Chen, and E. B. Fox. "A Complete Recipe for Stochastic Gradient MCMC". . In: *NIPS*. 2015.

A complete recipe to construct appropriate Itô diffusions

1. Ma *et al.*[19] gave a complete recipe to construct $F(\mathbf{x})$ and $\sigma(\mathbf{x})$:

$$F(\mathbf{x}) = -\left(D(\mathbf{x}) + Q(\mathbf{x})\right) \nabla_{\mathbf{x}} H(\mathbf{x}) + \Gamma(\mathbf{x})$$
$$\sigma(\mathbf{x}) = \sqrt{2D(\mathbf{x})} \, ,$$

   - $Q(\mathbf{x})$: a skew-symmetric curl matrix, *e.g.*, $-\mathbf{M} = \mathbf{M}^T$
   - $D(\mathbf{x})$: a positive semidefinite diffusion matrix

2. Any diffusion with the above form endows a marginal invariant measure: $\rho(\boldsymbol{\theta}) \propto e^{-U(\boldsymbol{\theta})}$.

3. In SGHMC, $D(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A \cdot \mathbf{I} \end{pmatrix}$, $Q(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}$.

[19] Y. A. Ma, T. Chen, and E. B. Fox. "A Complete Recipe for Stochastic Gradient MCMC". . In: *NIPS*. 2015.

A complete recipe to construct appropriate Itô diffusions

1. Ma *et al.*[19] gave a complete recipe to construct $F(\mathbf{x})$ and $\sigma(\mathbf{x})$:

$$F(\mathbf{x}) = -\left(D(\mathbf{x}) + Q(\mathbf{x})\right) \nabla_{\mathbf{x}} H(\mathbf{x}) + \Gamma(\mathbf{x})$$
$$\sigma(\mathbf{x}) = \sqrt{2D(\mathbf{x})}\ ,$$

   - $Q(\mathbf{x})$: a skew-symmetric curl matrix, *e.g.*, $-\mathbf{M} = \mathbf{M}^T$
   - $D(\mathbf{x})$: a positive semidefinite diffusion matrix

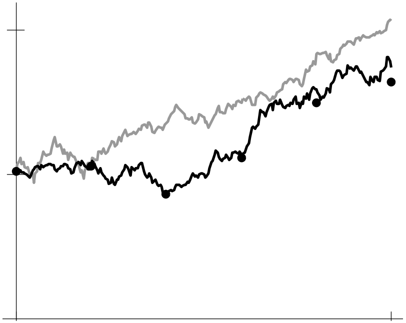
2. Any diffusion with the above form endows a marginal invariant measure: $\rho(\boldsymbol{\theta}) \propto e^{-U(\boldsymbol{\theta})}$.

3. In SGHMC, $D(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A \cdot \mathbf{I} \end{pmatrix}$, $Q(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}$.

[19] Y. A. Ma, T. Chen, and E. B. Fox. "A Complete Recipe for Stochastic Gradient MCMC". . In: *NIPS*. 2015.

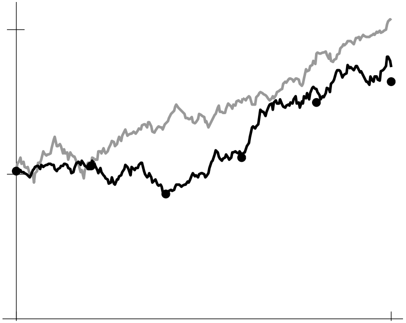
From diffusions to algorithms: numerical integrator

1. The diffusions defined previously are continuous-time Markov processes.
2. SG-MCMC algorithms approximate solutions of these Markov processes via numerical integrators/methods.
3. Characterize how accurate the algorithms approximate the continuous-time processes in terms of orders:
   - *e.g.*, a 1st-order numerical integrator approximates the true process, with an error bounded by $O(h)$, when evolving the process for time h



From diffusions to algorithms: numerical integrator

1. The diffusions defined previously are continuous-time Markov processes.
2. SG-MCMC algorithms approximate solutions of these Markov processes via numerical integrators/methods.
3. Characterize how accurate the algorithms approximate the continuous-time processes in terms of orders:
   - *e.g.*, a 1st-order numerical integrator approximates the true process, with an error bounded by $O(h)$, when evolving the process for time h



Example: SGHMC

$$\mathrm{d}\begin{pmatrix}\theta_t\\ \mathbf{p}_t\end{pmatrix} = \begin{pmatrix}\mathbf{p}_t\\ -A\mathbf{p}_t - \nabla_\theta U(\theta)\end{pmatrix}\mathrm{d}t + \sqrt{2A}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{pmatrix}\mathrm{d}\mathbf{w}_t$$

1. Use a 1st-order Euler integrator to solve the SDE:
   - divide the time into L small intervals, each with a duration h
   - in each interval, solve $(\theta_l, \mathbf{p}_l)$ sequentially, while fixing the others

$$\begin{aligned}\theta_l &= \theta_{l-1} + \mathbf{p}_{l-1} h_l\\ \mathbf{p}_l &= (1 - Ah_l)\mathbf{p}_{l-1} - \nabla_\theta U(\theta_l)h_l + \sqrt{2Ah_l}\mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}$$

2. Induce an error of $O(h_l)$ compared to exactly solving the SDE.
3. Also induce a global bias of $O(h)$ if $h_l = h, \forall l$ (introduced next).

Example: SGHMC

$$\mathrm{d}\begin{pmatrix}\boldsymbol{\theta}_t \\ \mathbf{p}_t\end{pmatrix} = \begin{pmatrix}\mathbf{p}_t \\ -A\mathbf{p}_t - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\end{pmatrix}\mathrm{d}t + \sqrt{2A}\begin{pmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}\end{pmatrix}\mathrm{d}\mathbf{w}_t$$

1. Use a 1st-order Euler integrator to solve the SDE:
   - divide the time into L small intervals, each with a duration h
   - in each interval, solve $(\boldsymbol{\theta}_l, \mathbf{p}_l)$ sequentially, while fixing the others

$$\begin{aligned}\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}_{l-1} h_l \\ \mathbf{p}_l &= (1 - Ah_l)\mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_l) h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}$$

2. Induce an error of $O(h_l)$ compared to exactly solving the SDE.
3. Also induce a global bias of $O(h)$ if $h_l = h, \forall l$ (introduced next).

Example: SGHMC

$$\mathrm{d}\begin{pmatrix}\boldsymbol{\theta}_t \\ \mathbf{p}_t\end{pmatrix} = \begin{pmatrix}\mathbf{p}_t \\ -A\,\mathbf{p}_t - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\end{pmatrix}\mathrm{d}t + \sqrt{2A}\begin{pmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}\end{pmatrix}\mathrm{d}\,\mathbf{w}_t$$

1. Use a 1st-order Euler integrator to solve the SDE:
   - divide the time into L small intervals, each with a duration h
   - in each interval, solve $(\boldsymbol{\theta}_l, \mathbf{p}_l)$ sequentially, while fixing the others

$$\begin{aligned}\boldsymbol{\theta}_l &= \boldsymbol{\theta}_{l-1} + \mathbf{p}_{l-1} h_l \\ \mathbf{p}_l &= (1 - Ah_l)\mathbf{p}_{l-1} - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}_l) h_l + \sqrt{2Ah_l}\,\mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}$$

2. Induce an error of $O(h_l)$ compared to exactly solving the SDE.
3. Also induce a global bias of $O(h)$ if $h_l = h, \forall l$ (introduced next).

High-order numerical integrators

1. Start an Itô diffusion from $\mathbf{x}_0$, let $\mathbf{x}_h$ be the random variable after evolving the diffusion for time h, $\tilde{\mathbf{x}}_h$ be the value obtained from a numerical method.
2. If $\mathbb{E}\left|f(\mathbf{x}_h) - f(\tilde{\mathbf{x}}_h)\right| = O(h^K)$, then the numerical integrator is said to be order K.
3. The Euler method is a 1st-order numerical integrator.
4. The symmetric splitting integrator[20] is a 2nd-order numerical integrator:
    - the idea is to split the infeasible SDE into several sub-SDEs, such that each of the sub-SDE can be solved exactly

[20] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

High-order numerical integrators

1. Start an Itô diffusion from $\mathbf{x}_0$, let $\mathbf{x}_h$ be the random variable after evolving the diffusion for time h, $\tilde{\mathbf{x}}_h$ be the value obtained from a numerical method.
2. If $\mathbb{E}\left|f(\mathbf{x}_h) - f(\tilde{\mathbf{x}}_h)\right| = O(h^K)$, then the numerical integrator is said to be order K.
3. The Euler method is a 1st-order numerical integrator.
4. The symmetric splitting integrator[20] is a 2nd-order numerical integrator:
   - the idea is to split the infeasible SDE into several sub-SDEs, such that each of the sub-SDE can be solved exactly

[20] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

High-order numerical integrators

1. Start an Itô diffusion from $\mathbf{x}_0$, let $\mathbf{x}_h$ be the random variable after evolving the diffusion for time h, $\tilde{\mathbf{x}}_h$ be the value obtained from a numerical method.
2. If $\mathbb{E}\left|f(\mathbf{x}_h) - f(\tilde{\mathbf{x}}_h)\right| = O(h^K)$, then the numerical integrator is said to be order K.
3. The Euler method is a 1st-order numerical integrator.
4. The symmetric splitting integrator[20] is a 2nd-order numerical integrator:
   - the idea is to split the infeasible SDE into several sub-SDEs, such that each of the sub-SDE can be solved exactly

[20] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

SGHMC using symmetric splitting integrators

$$\mathrm{d}\begin{pmatrix}\boldsymbol{\theta}\\ \mathbf{p}\end{pmatrix} = \begin{pmatrix}\mathbf{p}\\ -A\mathbf{p} - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\end{pmatrix}\mathrm{d}t + \sqrt{2A}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{pmatrix}\mathrm{d}\mathbf{w}$$

1. Split the above SDE into the following sub-SDEs:

$$A : \begin{cases}\mathrm{d}\boldsymbol{\theta} & = \mathbf{p}\,\mathrm{d}t\\ \mathrm{d}\mathbf{p} & = 0\end{cases}, B : \begin{cases}\mathrm{d}\boldsymbol{\theta} & = 0\\ \mathrm{d}\mathbf{p} & = -D\mathbf{p}\,\mathrm{d}t\end{cases}$$

$$O : \begin{cases}\mathrm{d}\boldsymbol{\theta} & = 0\\ \mathrm{d}\mathbf{p} & = -\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta})\mathrm{d}t + \sqrt{2D}\mathrm{d}\mathbf{w}\end{cases}$$

2. Solve the sub-SDEs in a symmetric way, *e.g.*, *ABOBA*, resulting in the following updates:

3. Induce $O(h^2)$ error, more accurate than the Euler integrator.

SGHMC using symmetric splitting integrators

$$\mathrm{d}\begin{pmatrix}\boldsymbol{\theta}\\ \mathbf{p}\end{pmatrix}=\begin{pmatrix}\mathbf{p}\\ -A\mathbf{p}-\nabla_{\boldsymbol{\theta}}U(\boldsymbol{\theta})\end{pmatrix}\mathrm{d}t+\sqrt{2A}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{pmatrix}\mathrm{d}\mathbf{w}$$

1. Split the above SDE into the following sub-SDEs:

$$A:\left\{\begin{array}{ll}\mathrm{d}\boldsymbol{\theta} & =\mathbf{p}\,\mathrm{d}t\\ \mathrm{d}\mathbf{p} & =0\end{array}\right.,B:\left\{\begin{array}{ll}\mathrm{d}\boldsymbol{\theta} & =0\\ \mathrm{d}\mathbf{p} & =-D\mathbf{p}\,\mathrm{d}t\end{array}\right.$$

$$O:\left\{\begin{array}{ll}\mathrm{d}\boldsymbol{\theta} & =0\\ \mathrm{d}\mathbf{p} & =-\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta})\mathrm{d}t+\sqrt{2D}\mathrm{d}\mathbf{w}\end{array}\right.$$

2. Solve the sub-SDEs in a symmetric way, *e.g.*, *ABOBA*, resulting in the following updates:

$$\boldsymbol{\theta}_l^{(1)}\overset{A}{=}\boldsymbol{\theta}_{l-1}+\mathbf{p}_{l-1}\,h/2\Rightarrow\mathbf{p}_l^{(1)}\overset{B}{=}e^{-Dh/2}\mathbf{p}_{l-1}\Rightarrow\mathbf{p}_l^{(2)}\overset{O}{=}\mathbf{p}_l^{(1)}-\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta}_l^{(1)})h$$
$$+\sqrt{2Dh}\mathcal{N}(\mathbf{0},\mathbf{I})\Rightarrow\ \mathbf{p}_l\overset{B}{=}e^{-Dh/2}\mathbf{p}_l^{(2)}\ \Rightarrow\ \boldsymbol{\theta}_l\overset{A}{=}\boldsymbol{\theta}_l^{(1)}+\mathbf{p}_l\,h/2$$

3. Induce $O(h^2)$ error, more accurate than the Euler integrator.

SGHMC using symmetric splitting integrators

$$\mathrm{d}\begin{pmatrix}\boldsymbol{\theta}\\ \mathbf{p}\end{pmatrix} = \begin{pmatrix}\mathbf{p}\\ -A\mathbf{p} - \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\end{pmatrix}\mathrm{d}t + \sqrt{2A}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{pmatrix}\mathrm{d}\mathbf{w}$$

1. Split the above SDE into the following sub-SDEs:

$$A : \begin{cases}\mathrm{d}\boldsymbol{\theta} &= \mathbf{p}\,\mathrm{d}t\\ \mathrm{d}\mathbf{p} &= 0\end{cases}, B : \begin{cases}\mathrm{d}\boldsymbol{\theta} &= 0\\ \mathrm{d}\mathbf{p} &= -D\mathbf{p}\,\mathrm{d}t\end{cases}$$

$$O : \begin{cases}\mathrm{d}\boldsymbol{\theta} &= 0\\ \mathrm{d}\mathbf{p} &= -\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta})\mathrm{d}t + \sqrt{2D}\mathrm{d}\mathbf{w}\end{cases}$$

2. Solve the sub-SDEs in a symmetric way, *e.g.*, *ABOBA*, resulting in the following updates:

$$\boldsymbol{\theta}_l^{(1)} \overset{A}{=} \boldsymbol{\theta}_{l-1} + \mathbf{p}_{l-1}\,h/2 \Rightarrow \mathbf{p}_l^{(1)} \overset{B}{=} e^{-Dh/2}\mathbf{p}_{l-1} \Rightarrow \mathbf{p}_l^{(2)} \overset{O}{=} \mathbf{p}_l^{(1)} - \nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta}_l^{(1)})h$$
$$+ \sqrt{2Dh}\mathcal{N}(\mathbf{0}, \mathbf{I}) \Rightarrow \mathbf{p}_l \overset{B}{=} e^{-Dh/2}\mathbf{p}_l^{(2)} \Rightarrow \boldsymbol{\theta}_l \overset{A}{=} \boldsymbol{\theta}_l^{(1)} + \mathbf{p}_l\,h/2$$

3. Induce $O(h^2)$ error, more accurate than the Euler integrator.

Outline: Stochastic Gradient Markov Chain Monte Carlo

1. SG-MCMC algorithms
2. Theory
   - Itô diffusion
   - Convergence theory

Setup

1. $\rho(\mathbf{x})$: stationary distribution of an Itô diffusion.
2. $\{\mathbf{x}_1, \cdots, \mathbf{x}_L\}$: samples from the corresponding SG-MCMC algorithm.
3. $\phi(\mathbf{x})$: a test function.
4. $\bar{\phi} \triangleq \int \phi(\mathbf{x})\rho(\mathbf{x})\mathrm{d}\,\mathbf{x}$: posterior average.
5. $\hat{\phi}_L \triangleq \frac{1}{L}\sum_{l=1}^{L}\phi(\mathbf{x}_l)$: sample average (fixed step size).
6. $\tilde{\phi}_L \triangleq \frac{1}{\sum_{l=1}^{L} h_l}\sum_{l=1}^{L} h_l\phi(\mathbf{x}_l)$: sample average (decreasing step sizes).
7. In weak convergence analysis, we study how $\hat{\phi}_L$ approximates $\bar{\phi}$, in terms of:
   - bias: $\left|\mathbb{E}\hat{\phi}_L - \bar{\phi}\right|$, or $\left|\mathbb{E}\tilde{\phi}_L - \bar{\phi}\right|$
   - mean square error (MSE): $\mathbb{E}\left(\hat{\phi}_L - \bar{\phi}\right)^2$, or $\mathbb{E}\left(\tilde{\phi}_L - \bar{\phi}\right)^2$

Setup

1. $\rho(\mathbf{x})$: stationary distribution of an Itô diffusion.
2. $\{\mathbf{x}_1, \cdots, \mathbf{x}_L\}$: samples from the corresponding SG-MCMC algorithm.
3. $\phi(\mathbf{x})$: a test function.
4. $\bar{\phi} \triangleq \int \phi(\mathbf{x})\rho(\mathbf{x})\mathrm{d}\,\mathbf{x}$: posterior average.
5. $\hat{\phi}_L \triangleq \frac{1}{L}\sum_{l=1}^{L} \phi(\mathbf{x}_l)$: sample average (fixed step size).
6. $\tilde{\phi}_L \triangleq \frac{1}{\sum_{l=1}^{L} h_l}\sum_{l=1}^{L} h_l\phi(\mathbf{x}_l)$: sample average (decreasing step sizes).
7. In weak convergence analysis, we study how $\hat{\phi}_L$ approximates $\bar{\phi}$, in terms of:
    - bias: $\left|\mathbb{E}\hat{\phi}_L - \bar{\phi}\right|$, or $\left|\mathbb{E}\tilde{\phi}_L - \bar{\phi}\right|$
    - mean square error (MSE): $\mathbb{E}\left(\hat{\phi}_L - \bar{\phi}\right)^2$, or $\mathbb{E}\left(\tilde{\phi}_L - \bar{\phi}\right)^2$

Typical assumptions

- The convergence theory relies on some assumptions on the continuous-time Itô diffusions and the numerical methods.

1. **Ellipticity/hypoellipticity**: the noise from Brownian motion could spread out over the whole space (diffusion coefficient).
2. **Smoothness and boundedness**: the drift coefficient $F(\mathbf{x})$ is smooth and bounded by some function.
3. **Ergodicity**: numerical methods are able to explore the whole parameter space.
4. **Nice properties (smooth, bounded) of ψ**: ψ is the solution functional of $\frac{1}{L}\sum_{l=1}^{L}\mathcal{L}\psi(\mathbf{x}_l) = \hat{\phi}_L - \bar{\phi}$, with $\mathcal{L}$ the infinite generator of the corresponding Itô diffusion.

Typical assumptions

- The convergence theory relies on some assumptions on the continuous-time Itô diffusions and the numerical methods.

Informally:

1. **Ellipticity/hypoellipticity**: the noise from Brownian motion could spread out over the whole space (diffusion coefficient).
2. **Smoothness and boundedness**: the drift coefficient $F(\mathbf{x})$ is smooth and bounded by some function.
3. **Ergodicity**: numerical methods are able to explore the whole parameter space.
4. **Nice properties (smooth, bounded) of ψ**: ψ is the solution functional of $\frac{1}{L}\sum_{l=1}^{L}\mathcal{L}\psi(\mathbf{x}_l) = \hat{\phi}_L - \bar{\phi}$, with $\mathcal{L}$ the infinite generator of the corresponding Itô diffusion.

Typical assumptions

- The convergence theory relies on some assumptions on the continuous-time Itô diffusions and the numerical methods.

Informally:

1. **Ellipticity/hypoellipticity**: the noise from Brownian motion could spread out over the whole space (diffusion coefficient).
2. **Smoothness and boundedness**: the drift coefficient $F(\mathbf{x})$ is smooth and bounded by some function.
3. **Ergodicity**: numerical methods are able to explore the whole parameter space.
4. **Nice properties (smooth, bounded) of ψ**: ψ is the solution functional of $\frac{1}{L}\sum_{l=1}^{L}\mathcal{L}\psi(\mathbf{x}_l) = \hat{\phi}_L - \bar{\phi}$, with $\mathcal{L}$ the infinite generator of the corresponding Itô diffusion.

Typical assumptions

- The convergence theory relies on some assumptions on the continuous-time Itô diffusions and the numerical methods.

Informally:

1. **Ellipticity/hypoellipticity**: the noise from Brownian motion could spread out over the whole space (diffusion coefficient).
2. **Smoothness and boundedness**: the drift coefficient $F(\mathbf{x})$ is smooth and bounded by some function.
3. **Ergodicity**: numerical methods are able to explore the whole parameter space.
4. **Nice properties (smooth, bounded) of ψ**: ψ is the solution functional of $\frac{1}{L}\sum_{l=1}^{L}\mathcal{L}\psi(\mathbf{x}_l) = \hat{\phi}_L - \bar{\phi}$, with $\mathcal{L}$ the infinite generator of the corresponding Itô diffusion.

Typical assumptions

- The convergence theory relies on some assumptions on the continuous-time Itô diffusions and the numerical methods.

Informally:

1. **Ellipticity/hypoellipticity**: the noise from Brownian motion could spread out over the whole space (diffusion coefficient).
2. **Smoothness and boundedness**: the drift coefficient $F(\mathbf{x})$ is smooth and bounded by some function.
3. **Ergodicity**: numerical methods are able to explore the whole parameter space.
4. **Nice properties (smooth, bounded) of ψ**: ψ is the solution functional of $\frac{1}{L}\sum_{l=1}^{L}\mathcal{L}\psi(\mathbf{x}_l) = \hat{\phi}_L - \bar{\phi}$, with $\mathcal{L}$ the infinite generator of the corresponding Itô diffusion.

Revisit orders of numerical integrators

1. SG-MCMC algorithms are discretized approximation of continuous-time Itô diffusions.
2. The accuracy of the samples generated from SG-MCMC algorithms is described by their orders of numerical methods.

For example:

1. Use an SG-MCMC algorithm to generate $\mathbf{x}_l$ from $\mathbf{x}_{l-1}$ with stepsize h.
2. Evolve the corresponding Itô diffusion exactly for time period h, starting from $\mathbf{x}_{l-1}$, and ending up with $\tilde{\mathbf{x}}_l$.
3. Calculate the difference: $D_f(\mathbf{x}_l, \tilde{\mathbf{x}}_l) \triangleq \mathbb{E}\left|f(\mathbf{x}_l) - f(\tilde{\mathbf{x}}_l)\right|$, where f is a test function.
4. If $D_f(\mathbf{x}_l, \tilde{\mathbf{x}}_l) = O(h^K)$, then the numerical integrator is called an Kth-order integrator.

Revisit orders of numerical integrators

1. SG-MCMC algorithms are discretized approximation of continuous-time Itô diffusions.
2. The accuracy of the samples generated from SG-MCMC algorithms is described by their orders of numerical methods.
3. The popular Euler method is a 1st-order integrator.
4. The symmetric splitting integrator[a] is a 2nd-order integrator.
5. We will present results with general Kth-order integrators.

[a] C. Chen, N. Ding, and L. Carin. “On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators”. In: *NIPS*. 2015.

Convergence bounds of bias and MSE[21]

Theorem (Fixed step size)

Under standard assumptions, the bias and MSE of a fixed-step-size SG-MCMC with a Kth-order integrator at time $T = hL$ are bounded as:

$$\textit{Bias: } \left|\mathbb{E}\hat{\phi}_L - \bar{\phi}\right| \leq C_1 \left(\frac{1}{Lh} + h^K\right)$$

$$\textit{MSE: } \mathbb{E}\left(\hat{\phi}_L - \bar{\phi}\right)^2 \leq C_2 \left(\frac{1}{Lh} + h^{2K}\right)$$

[21] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

Convergence bounds of bias and MSE[21]

Theorem (Decreasing step sizes)

Under standard assumptions, the bias and MSE of a decreasing-step-size SG-MCMC with a Kth-order integrator at time $S_L \triangleq \sum_{l=1}^{L} h_l$ are bounded as:

$$\textit{Bias: } \left|\mathbb{E}\tilde{\phi}_L - \bar{\phi}\right| \leq C_1 \left(\frac{1}{S_L} + \frac{\sum_{l=1}^{L} h_l^{K+1}}{S_L}\right)$$

$$\textit{MSE: } \mathbb{E}\left(\tilde{\phi}_L - \bar{\phi}\right)^2 \leq C_2 \left(\frac{1}{S_L} + \frac{(\sum_{l=1}^{L} h_l^{K+1})^2}{S_L^2} + \frac{\sum_{l=1}^{L} h_l^2}{S_L^2}\right)$$

- To ensure the bias and MSE asymptotically approach zero, we need:

$$S_L \to \infty, \quad \frac{\sum_{l=1}^{L} h_l^{K+1}}{S_L} \to 0, \quad \frac{\sum_{l=1}^{L} h_l^2}{S_L^2} \to 0$$

[21] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

Optimal convergence rates

1. When optimizing the bounds over step size, we get the optimal convergence rates.

Optimal convergence rates

1. When optimizing the bounds over step size, we get the optimal convergence rates.

Fixed step size:

$$\text{Bias: } \left|\mathbb{E}\hat{\phi}_L - \bar{\phi}\right| \leq C_1 \left(\frac{1}{Lh} + h^K\right) \Rightarrow C_1 L^{-K/(K+1)}$$

$$\text{MSE: } \mathbb{E}\left(\hat{\phi}_L - \bar{\phi}\right)^2 \leq C_2 \left(\frac{1}{Lh} + h^{2K}\right) \Rightarrow C_2 L^{-2K/(2K+1)}$$

2. Slower than stochastic optimization:
   - bias typically decreases as L^{-1}
3. Also slower than standard MCMC:
   - square root of MSE typically decreases as $L^{-1/2}$
   - however, standard MCMC is typically computationally infeasible for even a single iteration

Optimal convergence rates

1. When optimizing the bounds over step size, we get the optimal convergence rates.

Fixed step size:

$$\text{Bias: } \left|\mathbb{E}\hat{\phi}_L - \bar{\phi}\right| \leq C_1 \left(\frac{1}{Lh} + h^K\right) \Rightarrow C_1 L^{-K/(K+1)}$$

$$\text{MSE: } \mathbb{E}\left(\hat{\phi}_L - \bar{\phi}\right)^2 \leq C_2 \left(\frac{1}{Lh} + h^{2K}\right) \Rightarrow C_2 L^{-2K/(2K+1)}$$

2. Slower than stochastic optimization:
   - bias typically decreases as L^{-1}
3. Also slower than standard MCMC:
   - square root of MSE typically decreases as $L^{-1/2}$
   - however, standard MCMC is typically computationally infeasible for even a single iteration

Optimal convergence rates

1. When optimizing the bounds over step size, we get the optimal convergence rates.

Fixed step size:

$$\text{Bias: } \left|\mathbb{E}\hat{\phi}_L - \bar{\phi}\right| \leq C_1 \left(\frac{1}{Lh} + h^K\right) \Rightarrow C_1 L^{-K/(K+1)}$$

$$\text{MSE: } \mathbb{E}\left(\hat{\phi}_L - \bar{\phi}\right)^2 \leq C_2 \left(\frac{1}{Lh} + h^{2K}\right) \Rightarrow C_2 L^{-2K/(2K+1)}$$

2. Slower than stochastic optimization:
   - bias typically decreases as L^{-1}
3. Also slower than standard MCMC:
   - square root of MSE typically decreases as $L^{-1/2}$
   - however, standard MCMC is typically computationally infeasible for even a single iteration

Optimal convergence rates

1. When optimizing the bounds over step size, we get the optimal convergence rates.

Decreasing step sizes: consider $h_l \propto l^{-\alpha}$

$$\text{Bias: } \left|\mathbb{E}\tilde{\phi}_L - \bar{\phi}\right| \leq C_1 \left(\frac{1}{S_L} + \frac{\sum_{l=1}^{L} h_l^{K+1}}{S_L}\right)$$

$$\Longrightarrow C_1 L^{-K/(K+1)}, \text{ with } \alpha = 1/(K+1)$$

$$\text{MSE: } \mathbb{E}\left(\tilde{\phi}_L - \bar{\phi}\right)^2 \leq C_2 \left(\frac{1}{S_L} + \frac{(\sum_{l=1}^{L} h_l^{K+1})^2}{S_L^2} + \frac{\sum_{l=1}^{L} h_l^2}{S_L^2}\right)$$

$$\Longrightarrow C_2 L^{-2K/(2K+1)}, \text{ with } \alpha = 1/(2K+1)$$

2. Behave similarly to the fixed-step-size case

Optimal convergence rates

1. When optimizing the bounds over step size, we get the optimal convergence rates.

Decreasing step sizes: consider $h_l \propto l^{-\alpha}$

$$\text{Bias: } \left|\mathbb{E}\tilde{\phi}_L - \bar{\phi}\right| \leq C_1 \left(\frac{1}{S_L} + \frac{\sum_{l=1}^L h_l^{K+1}}{S_L}\right)$$

$$\Longrightarrow C_1 L^{-K/(K+1)}, \text{ with } \alpha = 1/(K+1)$$

$$\text{MSE: } \mathbb{E}\left(\tilde{\phi}_L - \bar{\phi}\right)^2 \leq C_2 \left(\frac{1}{S_L} + \frac{(\sum_{l=1}^L h_l^{K+1})^2}{S_L^2} + \frac{\sum_{l=1}^L h_l^2}{S_L^2}\right)$$

$$\Longrightarrow C_2 L^{-2K/(2K+1)}, \text{ with } \alpha = 1/(2K+1)$$

2. Behave similarly to the fixed-step-size case

Synthetic experiments[22]

1. A standard Gaussian model:

$$x_i \sim \mathcal{N}(\theta, 1), \quad \theta \sim \mathcal{N}(0, 1), \quad i = 1, \cdots, 1000$$

2. Test function: $\phi(\theta) = \theta^2$.

[22] C. Chen, N. Ding, and L. Carin. “On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators”. In: *NIPS*. 2015.

Synthetic experiments[22]

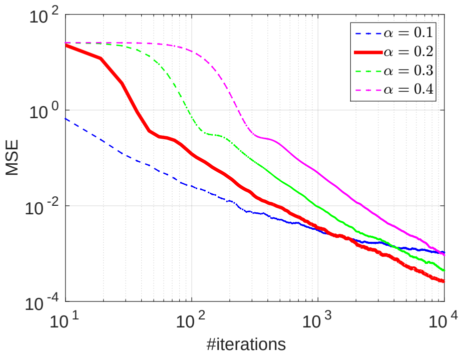
1. A standard Gaussian model:

$$x_i \sim \mathcal{N}(\theta, 1), \quad \theta \sim \mathcal{N}(0, 1), \quad i = 1, \cdots, 1000$$

2. Test function: $\phi(\theta) = \theta^2$.

Fixed step size:

- Use a 2nd-order symmetric splitting integrator.
- Optimal step size: $h \propto L^{-\alpha}$ with $\alpha = 0.2$ for the MSE.



[22] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

Synthetic experiments[22]

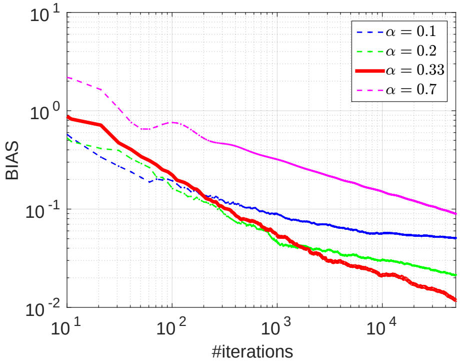
1. A standard Gaussian model:

$$x_i \sim \mathcal{N}(\theta, 1), \quad \theta \sim \mathcal{N}(0, 1), \quad i = 1, \cdots, 1000$$

2. Test function: $\phi(\theta) = \theta^2$.

Decreasing step sizes:

- Use step size sequence $h_l \propto l^{-\alpha}$.
- Optimal $\alpha = 1/3$ for the bias.



[22] C. Chen, N. Ding, and L. Carin. "On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators". In: *NIPS*. 2015.

Part Three: SG-MCMC for Stochastic Optimization

Outline

4 SG-MCMC for Stochastic Optimization

- Bridging the Gap between SG-MCMC and Stochastic Optimization

Motivation

1. A key problem in big-data era, especially in deep learning, is to design algorithms that better solve a complex and high-dimensional problem.
2. Stochastic optimization:
   - computationally efficient, fast convergence, prone to local optimal
3. Stochastic gradient MCMC:
   - computationally efficient, slower convergence, able to explore the parameter space
4. Can we combine advantages from both?
5. What is in between them?

Motivation

1. A key problem in big-data era, especially in deep learning, is to design algorithms that better solve a complex and high-dimensional problem.
2. Stochastic optimization:
    - computationally efficient, fast convergence, prone to local optimal
3. Stochastic gradient MCMC:
    - computationally efficient, slower convergence, able to explore the parameter space
4. Can we combine advantages from both?
5. What is in between them?

Motivation

1. A key problem in big-data era, especially in deep learning, is to design algorithms that better solve a complex and high-dimensional problem.
2. Stochastic optimization:
   - computationally efficient, fast convergence, prone to local optimal
3. Stochastic gradient MCMC:
   - computationally efficient, slower convergence, able to explore the parameter space
4. Can we combine advantages from both?
5. What is in between them?

Motivation

1. A key problem in big-data era, especially in deep learning, is to design algorithms that better solve a complex and high-dimensional problem.
2. Stochastic optimization:
   - computationally efficient, fast convergence, prone to local optimal
3. Stochastic gradient MCMC:
   - computationally efficient, slower convergence, able to explore the parameter space
4. Can we combine advantages from both?
5. What is in between them?

Stochastic optimization

1. Stochastic gradient descent (SGD):
   - basic stochastic optimization algorithm, without considering neither momentum and preconditioning
2. SGD with momentum (SGD-M):
   - extending SGD with momentum
3. RMSProp, Adadelta $\cdots$:
   - extending SGD with preconditioner
4. Adam:
   - extending SGD with both momentum and preconditioner

Stochastic optimization

1. Stochastic gradient descent (SGD):
   - basic stochastic optimization algorithm, without considering neither momentum and preconditioning
2. SGD with momentum (SGD-M):
   - extending SGD with momentum
3. RMSProp, Adadelta $\cdots$:
   - extending SGD with preconditioner
4. Adam:
   - extending SGD with both momentum and preconditioner

Stochastic optimization

1. Stochastic gradient descent (SGD):
   - basic stochastic optimization algorithm, without considering neither momentum and preconditioning
2. SGD with momentum (SGD-M):
   - extending SGD with momentum
3. RMSProp, Adadelta $\cdots$:
   - extending SGD with preconditioner
4. Adam:
   - extending SGD with both momentum and preconditioner

Stochastic optimization

1. Stochastic gradient descent (SGD):
    - basic stochastic optimization algorithm, without considering neither momentum and preconditioning
2. SGD with momentum (SGD-M):
    - extending SGD with momentum
3. RMSProp, Adadelta $\cdots$:
    - extending SGD with preconditioner
4. Adam:
    - extending SGD with both momentum and preconditioner

Stochastic gradient MCMC

1. Stochastic gradient Langevin dynamics (SGLD):
   - Bayesian analog of SGD, without considering neither momentum and preconditioning
2. Stochastic gradient Hamiltonian Monte Carlo (SGHMC):
   - Bayesian analog of SGD-M, with momentum
3. Preconditioned stochastic gradient Langevin dynamics (PSGLD):
   - Bayesian analog of RMSProp, with preconditioner
4. Stochastic gradient thermostats (SGNHT):
   - Bayesian sampling with adaptive momentum, does not have a stochastic optimization analog

Stochastic gradient MCMC

1. Stochastic gradient Langevin dynamics (SGLD):
   - Bayesian analog of SGD, without considering neither momentum and preconditioning
2. Stochastic gradient Hamiltonian Monte Carlo (SGHMC):
   - Bayesian analog of SGD-M, with momentum
3. Preconditioned stochastic gradient Langevin dynamics (PSGLD):
   - Bayesian analog of RMSProp, with preconditioner
4. Stochastic gradient thermostats (SGNHT):
   - Bayesian sampling with adaptive momentum, does not have a stochastic optimization analog

Stochastic gradient MCMC

1. Stochastic gradient Langevin dynamics (SGLD):
   - Bayesian analog of SGD, without considering neither momentum and preconditioning
2. Stochastic gradient Hamiltonian Monte Carlo (SGHMC):
   - Bayesian analog of SGD-M, with momentum
3. Preconditioned stochastic gradient Langevin dynamics (PSGLD):
   - Bayesian analog of RMSProp, with preconditioner
4. Stochastic gradient thermostats (SGNHT):
   - Bayesian sampling with adaptive momentum, does not have a stochastic optimization analog

Stochastic gradient MCMC

1. Stochastic gradient Langevin dynamics (SGLD):
   - Bayesian analog of SGD, without considering neither momentum and preconditioning
2. Stochastic gradient Hamiltonian Monte Carlo (SGHMC):
   - Bayesian analog of SGD-M, with momentum
3. Preconditioned stochastic gradient Langevin dynamics (PSGLD):
   - Bayesian analog of RMSProp, with preconditioner
4. Stochastic gradient thermostats (SGNHT):
   - Bayesian sampling with adaptive momentum, does not have a stochastic optimization analog

Bridging the gap

1. Santa: the Stochastic AnNealing Thermostats with Adaptive momentum algorithm.

Table: SG-MCMC algorithms and their optimization counterparts.

Algorithms	**SG-MCMC**		**Optimization**
Basic	SGLD	$\Longleftrightarrow$	SGD
Precondition	pSGLD	$\Longleftrightarrow$	RMSprop
Momentum	SGHMC	$\Longleftrightarrow$	SGD-M
Thermostat	SGNHT	$\approx$	Santa

1. What is in between them?
   - it is about the noise

Example: noise in SGLD

1. Update equation for SGLD:

$$\boldsymbol{\theta}_{l+1} = \boldsymbol{\theta}_l - \nabla_{\boldsymbol{\theta}} \tilde{U}_l(\boldsymbol{\theta}) h_l + \sqrt{2h_l}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

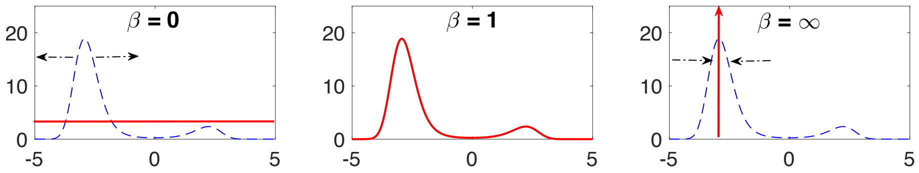
2. How about adding $\sqrt{2h_l/\beta}\mathcal{N}(\mathbf{0}, \mathbf{I})$ noise instead of $\sqrt{2h_l}\mathcal{N}(\mathbf{0}, \mathbf{I})$?
   - it would end up sampling from an annealed distribution: $\rho_\beta(\boldsymbol{\theta}) \propto e^{-\beta U(\boldsymbol{\theta})}$
   - when $\beta = 0$, $\rho_\beta(\boldsymbol{\theta})$ is a uniform distribution
   - when $\beta = \infty$, $\rho_\beta(\boldsymbol{\theta})$ is a spike located at $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$

Example: noise in SGLD

1. Update equation for SGLD:

$$\boldsymbol{\theta}_{l+1} = \boldsymbol{\theta}_l - \nabla_{\boldsymbol{\theta}} \tilde{U}_l(\boldsymbol{\theta}) h_l + \sqrt{2h_l}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

2. How about adding $\sqrt{2h_l/\beta}\mathcal{N}(\mathbf{0}, \mathbf{I})$ noise instead of $\sqrt{2h_l}\mathcal{N}(\mathbf{0}, \mathbf{I})$?
   - it would end up sampling from an annealed distribution: $\rho_\beta(\boldsymbol{\theta}) \propto e^{-\beta U(\boldsymbol{\theta})}$
   - when $\beta = 0$, $\rho_\beta(\boldsymbol{\theta})$ is a uniform distribution
   - when $\beta = \infty$, $\rho_\beta(\boldsymbol{\theta})$ is a spike located at $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$



A more expressive framework

1. What is lacking in recent stochastic optimization algorithms?
   - lacking of simultaneously element-wise adaptive preconditioner and adaptive momentum
2. SGNHT comes to rescue:
   - the thermostat variable adaptively learns the momentum
   - the annealing idea turns the SG-MCMC algorithm into stochastic optimization

A more expressive framework

1. What is lacking in recent stochastic optimization algorithms?
   - lacking of simultaneously element-wise adaptive preconditioner and adaptive momentum
2. SGNHT comes to rescue:
   - the thermostat variable adaptively learns the momentum
   - the annealing idea turns the SG-MCMC algorithm into stochastic optimization

$$\mathrm{d}\begin{pmatrix}\theta_t \\ \mathbf{p}_t \\ \xi_t\end{pmatrix} = \begin{pmatrix}\mathbf{p}_t \\ -\xi_t\,\mathbf{p}_t - \nabla_\theta U(\theta_t) \\ \mathbf{p}_t^T\,\mathbf{p}_t/D - 1\end{pmatrix}\mathrm{d}t + \sqrt{2A}\begin{pmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{pmatrix}\mathrm{d}\,\mathbf{w}_t$$

The Santa algorithm

1. The Stochastic AnNealing Thermostats with Adaptive momentum (Santa) algorithm extends SGNHT with preconditioners and annealing temperature.
2. Itô diffusion form:

$$\begin{cases} \mathrm{d}\theta &= G_1(\theta)\boldsymbol{p}\mathrm{d}t \\ \mathrm{d}\boldsymbol{p} &= \Big(-G_1(\theta)\nabla_\theta U(\theta) - \Xi\boldsymbol{p} + \frac{1}{\beta}\nabla_\theta G_1(\theta) \\ \quad +G_1(\theta)(\Xi - G_2(\theta))\nabla_\theta G_2(\theta)\Big)\,\mathrm{d}t + (\frac{2}{\beta}G_2(\theta))^{\frac{1}{2}}\mathrm{d}w \\ \mathrm{d}\Xi &= \Big(\mathrm{diag}(\boldsymbol{p}\odot\boldsymbol{p}) - \frac{1}{\beta}I\Big)\,\mathrm{d}t\ , \end{cases} \tag{1}$$

   where $\boldsymbol{G}_1(\theta)$ and $\boldsymbol{G}_2(\theta)$ are some preconditioners, typically constructed using RMSProp.
3. Santa algorithm is derived by solving (1) numerically with an increasing sequence of inverse temperatures β.

The Santa algorithm

1. The Stochastic AnNealing Thermostats with Adaptive momentum (Santa) algorithm extends SGNHT with preconditioners and annealing temperature.
2. Itô diffusion form:

$$\left\{\begin{array}{ll} \mathrm{d}\boldsymbol{\theta} & = G_1(\boldsymbol{\theta})\boldsymbol{p}\mathrm{d}t \\ \mathrm{d}\boldsymbol{p} & = \Big(-G_1(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}U(\boldsymbol{\theta}) - \boldsymbol{\Xi}\boldsymbol{p} + \frac{1}{\beta}\nabla_{\boldsymbol{\theta}}G_1(\boldsymbol{\theta}) \\ & \quad +G_1(\boldsymbol{\theta})(\boldsymbol{\Xi} - G_2(\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}}G_2(\boldsymbol{\theta})\Big)\,\mathrm{d}t + (\frac{2}{\beta}G_2(\boldsymbol{\theta}))^{\frac{1}{2}}\mathrm{d}w \\ \mathrm{d}\boldsymbol{\Xi} & = \Big(\mathrm{diag}(\boldsymbol{p}\odot\boldsymbol{p}) - \frac{1}{\beta}I\Big)\,\mathrm{d}t\ , \end{array}\right. \tag{1}$$

where $\boldsymbol{G}_1(\boldsymbol{\theta})$ and $\boldsymbol{G}_2(\boldsymbol{\theta})$ are some preconditioners, typically constructed using RMSProp.

3. Santa algorithm is derived by solving (1) numerically with an increasing sequence of inverse temperatures β.

The Santa algorithm

1. The Stochastic AnNealing Thermostats with Adaptive momentum (Santa) algorithm extends SGNHT with preconditioners and annealing temperature.
2. Itô diffusion form:

$$\begin{cases} \mathrm{d}\boldsymbol{\theta} &= G_1(\boldsymbol{\theta})\boldsymbol{p}\mathrm{d}t \\ \mathrm{d}\boldsymbol{p} &= \Big(-G_1(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}U(\boldsymbol{\theta}) - \boldsymbol{\Xi}\boldsymbol{p} + \frac{1}{\beta}\nabla_{\boldsymbol{\theta}}G_1(\boldsymbol{\theta}) \\ & \quad +G_1(\boldsymbol{\theta})(\boldsymbol{\Xi} - G_2(\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}}G_2(\boldsymbol{\theta})\Big)\,\mathrm{d}t + (\frac{2}{\beta}G_2(\boldsymbol{\theta}))^{\frac{1}{2}}\mathrm{d}w \\ \mathrm{d}\boldsymbol{\Xi} &= \Big(\mathrm{diag}(\boldsymbol{p} \odot \boldsymbol{p}) - \frac{1}{\beta}I\Big)\,\mathrm{d}t\ , \end{cases} \tag{1}$$

   where $\boldsymbol{G}_1(\boldsymbol{\theta})$ and $\boldsymbol{G}_2(\boldsymbol{\theta})$ are some preconditioners, typically constructed using RMSProp.
3. Santa algorithm is derived by solving (1) numerically with an increasing sequence of inverse temperatures β.

The Santa algorithm

```
Input: $\eta_t$ (learning rate), $\sigma$, $\lambda$, burnin, $\beta = \{\beta_1, \beta_2, \cdots\} \to \infty$,
       $\{\zeta_t \in \mathbb{R}^p\} \sim N(\mathbf{0}, \mathbf{I}_p)$.
Initialize $\theta_0$, $\boldsymbol{u}_0 = \sqrt{\eta} \times N(\mathbf{0}, \mathbf{I}_p)$, $\boldsymbol{\alpha}_0 = \sqrt{\eta} C$, $\boldsymbol{v}_0 = 0$ ;
for $t = 1, 2, \ldots$ do
    Evaluate $\tilde{\boldsymbol{f}}_t \triangleq \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{t-1})$ on the $t^{\text{th}}$ mini-batch;
    $\boldsymbol{v}_t = \sigma \boldsymbol{v}_{t-1} + \frac{1-\sigma}{N^2} \tilde{\boldsymbol{f}}_t \odot \tilde{\boldsymbol{f}}_t$ ;
    $\boldsymbol{g}_t = 1 \oslash \sqrt{\lambda + \sqrt{\boldsymbol{v}_t}}$ ;
    if $t <$ burnin then
        /* exploration */
        $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + (\boldsymbol{u}_{t-1} \odot \boldsymbol{u}_{t-1} - \eta/\beta_t)$;
        $\boldsymbol{u}_t = \frac{\eta}{\beta_t} \left(1 - \boldsymbol{g}_{t-1} \oslash \boldsymbol{g}_t\right) \oslash \boldsymbol{u}_{t-1} + \sqrt{\frac{2\eta}{\beta_t} \boldsymbol{g}_{t-1}} \odot \zeta_t$
    else
        /* refinement */
        $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}$;    $\boldsymbol{u}_t = \mathbf{0}$;
    end
    $\boldsymbol{u}_t = \boldsymbol{u}_t + (1 - \boldsymbol{\alpha}_t) \odot \boldsymbol{u}_{t-1} - \eta \boldsymbol{g}_t \odot \tilde{\boldsymbol{f}}_t$;    $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \boldsymbol{g}_t \odot \boldsymbol{u}_t$;
end
```

The Santa algorithm

1. It is an stochastic optimization algorithm that starts from Bayesian sampling.
2. It is able to jump out of local modes easier than traditional stochastic optimization algorithms.
3. Under certain conditions, it is proved to converge in expectation to the global mode.
4. It converges fast in empirical studies.

Illustration

1. Optimizing the double-well potential:

$$U(\theta) = (\theta + 4)(\theta + 1)(\theta - 1)(\theta - 3)/14 + 0.5 \; .$$

2. Start close to a local mode.
3. RMSProp gets stuck, while Santa is able to jump out of the local mode.

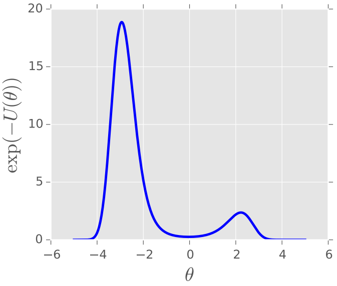
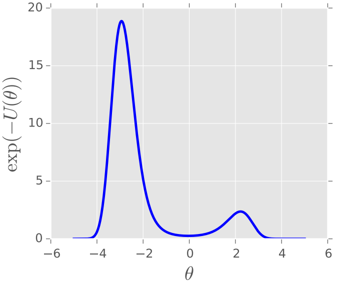


Figure: (Left) Double-well potential. (Right) The evolution of θ using Santa and RMSprop algorithms.

Illustration

1. Optimizing the double-well potential:
$$U(\theta) = (\theta + 4)(\theta + 1)(\theta - 1)(\theta - 3)/14 + 0.5 \ .$$
2. Start close to a local mode.
3. RMSProp gets stuck, while Santa is able to jump out of the local mode.



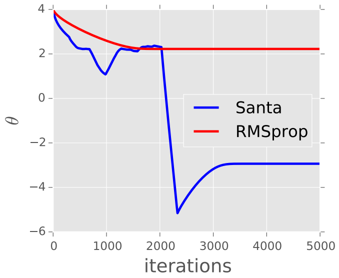


Figure: (Left) Double-well potential. (Right) The evolution of θ using Santa and RMSprop algorithms.

Feedforward neural networks and convolutional neural networks

- Detailed parameter setting is given in the paper[23].
- Santa outperforms other algorithms in most cases.

Table: Test error on MNIST classification using FNN and CNN.

Algorithms	FNN-400	FNN-800	CNN
Santa	**1.21%**	**1.16%**	**0.47%**
Adam	1.53%	1.47%	0.59%
RMSprop	1.59%	1.43%	0.64%
SGD-M	1.66%	1.72%	0.77%
SGD	1.72%	1.47%	0.81%
SGLD	1.64%	1.41%	0.71%
BPB$^\diamond$	1.32%	1.34%	–
SGD, Dropout$^\diamond$	1.51%	1.33%	–
Stoc. Pooling$^\triangleright$	–	–	0.47%
NIN, Dropout$^\circ$	–	–	0.47%
Maxout, Dropout*	–	–	0.45%

[23] C. Chen et al. “Bridging the Gap between Stochastic Gradient MCMC and Stochastic Optimization”. In: *AISTATS*. 2016.

Recurrent neural networks (RNN)

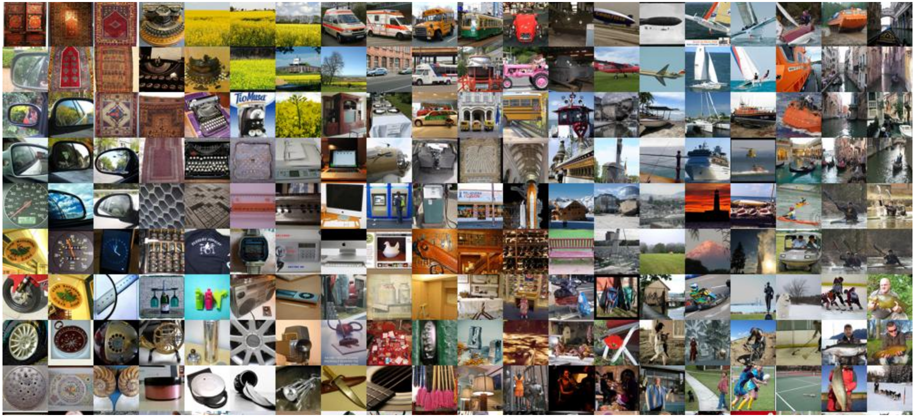
1. Language modeling with vanilla RNN.
2. Test on four publicly available datasets.

Table: Test negative log-likelihood on 4 datasets.

Algorithms	Piano.	Nott.	Muse.	JSB.
Santa	**7.60**	**3.39**	**7.20**	**8.46**
Adam	8.00	3.70	7.56	8.51
RMSprop	7.70	3.48	7.22	8.52
SGD-M	8.32	3.60	7.69	8.59
SGD	11.13	5.26	10.08	10.81
HF$^\diamond$	7.66	3.89	**7.19**	8.58
SGD-M$^\diamond$	8.37	4.46	8.13	8.71

ImageNet visual recognition challenge[24]

1. More than 10 million annotated natural images, with 1000 classed.
2. Use to compete different machine learning algorithms, dominated by deep learning recent years.



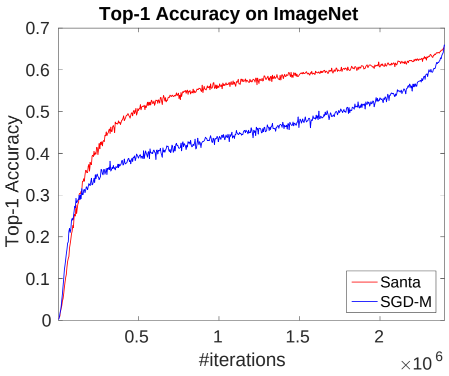
[24] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR*. 2009.

GoogleNet for ImageNet classification

1. Use ILSVRC 2012 for training and testing.
2. Compared with SGD with momentum, other algorithms did not seem to work.
3. Did not tune the parameters, use the default setting for GoogleNet provided in the Caffe package.

GoogleNet for ImageNet classification

1. Santa converges much faster than SGD-M.
2. Use the default step size: $h_t = a\sqrt{1 - t/T}$, can not run more than T iterations.



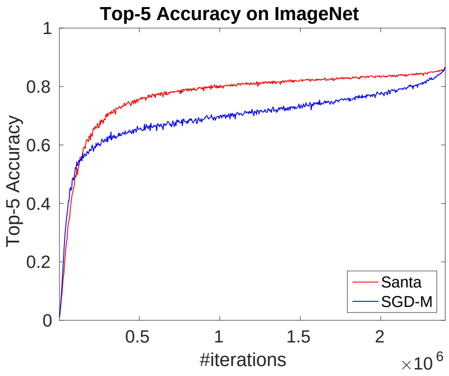


Figure: Santa vs. SGD with momentum on ImageNet.

Why adding gradient noise improves DNN training ?

1. A recent paper[25] finds that adding gradient noise helps train very deep network:
   - the reason was not very clear
2. It essentially adds small random Gaussian noise in parameter updates.
3. Equivalent to sampling from an annealed distribution: $\rho_\beta(\boldsymbol{\theta}) \propto e^{-\beta U(\boldsymbol{\theta})}$, with some large β.
4. The good performance can be explained by the Santa algorithm:
   - noise makes the algorithm jump out of local modes easier
   - large β smooths the objective function heavier, thus ends up better local modes
5. Conclusion holds when the gradient noise is not Gaussian:
   - as long as it has zero mean and finite variance
   - theoretical analysis follows similarly, with a little modification

[25] A. Neelakantan et al. "Adding Gradient Noise Improves Learning for Very Deep Networks". In: *ICLR workshop*. 2016.

Conclusion

I have covered:

1. Basic concepts in MCMC.
2. Basic ideas in SG-MCMC, a review of basic SG-MCMC algorithms.
3. Theory related to stochastic differential equations and Itó diffusions.
4. Convergence theory.
5. How to extend SG-MCMC for stochastic optimization.

Thank You