Efficient Sampling Techniques, Take Two

Sergey Samsonov

HDI Lab,
HSE University

NATIONAL RESEARCH
UNIVERSITY

June 20-25, 2022

Hamiltionian Monte-Carlo (HMC)

- Following Neal [2011], introduce an auxiliary momentum variable r_i for each model variable θ_i, $i \in \{1, \ldots, d\}$;
- Consider the (unnormalized) joint density

$$p(\theta, r) \propto \exp\{-U(\theta) - \tfrac{1}{2} r^\top r\}\,, (\theta, r) \in \mathbb{R}^{2d}\,. \tag{1}$$

- We aim at sampling from the joint density $p(\theta, r)$, despite we are interested only in the θ marginal;
- $\theta \in \mathbb{R}^d$ - particle's position; r - momentum; $U(\theta)$ - potential energy, $\frac{1}{2} r^\top r$ is the kinetic energy of the particle.
- $H(\theta, r) = U(\theta) + \frac{1}{2} r^\top r$ - *Hamiltonian*.

HMC dynamics

Now we consider the evolution of the particle according to the *Hamiltonian dynamics*

$$\begin{cases} \frac{dr_i}{dt} & = \frac{\partial H}{\partial \theta_i} \\ \frac{d\theta_i}{dt} & = -\frac{\partial H}{\partial r_i} \, , i \in \{1, \ldots, d\} \, . \end{cases} \tag{2}$$

Example: 1d Gaussian

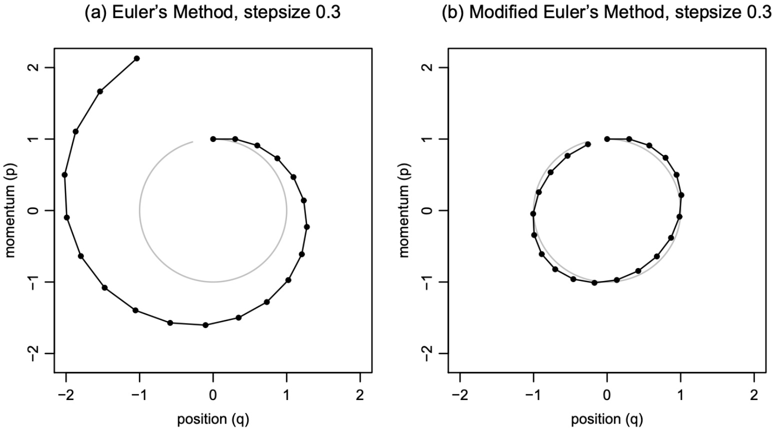
Gaussian case

Let $U(\theta) = \frac{\theta^2}{2}$. Write down the corresponding Hamiltonian dynamics.

Properties of Hamiltonian dynamics

- Hamiltonian dynamics (2) is reversible: mapping $T_s : (\theta_t, r_t) \mapsto (\theta_{t+s}, r_{t+s})$ is bijective, and has an inverse T_{-s};
- Hamiltonian $H(\theta, r)$ is invariant for the dynamics (2);
- Hamiltonian dynamics is volume-preserving in (θ, r)-space (Liouville's theorem) .

Different discretizations, Neal [2011]



HMC

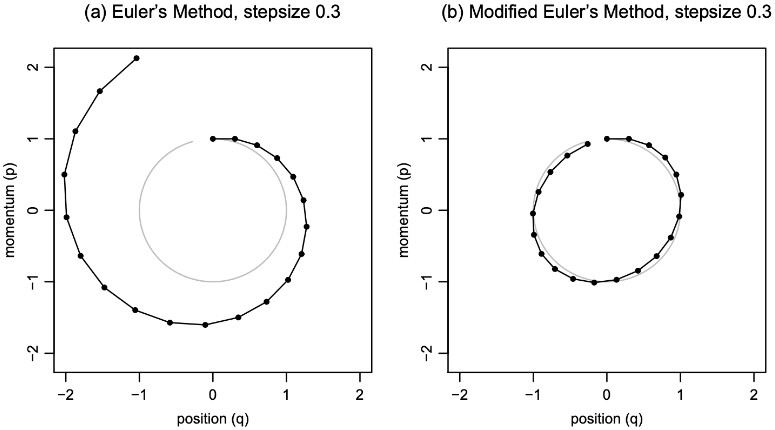
To simulate the evolution of the system over time, we can use the *Leapfrog integrator*

Leapfrog(θ_t, r_t, ϵ)

1. $r_{t+\epsilon/2} = r_t - (\epsilon/2)\nabla_\theta U(\theta_t)$;
2. $\theta_{t+\epsilon} = \theta_t + \epsilon r_{t+\epsilon/2}$;
3. $r_{t+\epsilon} = r_{t+\epsilon/2} - (\epsilon/2)\nabla_\theta U(\theta_{t+\epsilon})$.

In the above r_t and θ_t denote the values of the momentum and position variables r and θ at time t.

Different discretizations, Neal [2011]



Hamiltionian Monte-Carlo (HMC): algorithm, Hoffman et al. [2014]

Algorithm 1: Hamiltonian Monte Carlo

Input : θ_0, ϵ, L, $U(\theta)$, n:

Output: New sample Y_{j+1}

for $k = 1$ *to* n **do**

  Sample $r_0 \sim \mathcal{N}(0, \mathrm{I}_d)$;

  Set $\theta_k \leftarrow \theta_{k-1}, \tilde{\theta} \leftarrow \theta_{k-1}, \tilde{r} \leftarrow r_0$;

  for $i = 1$ *to* L **do**

    Set $\tilde{\theta}, \tilde{r} \leftarrow \mathit{Leapfrog}(\tilde{\theta}, \tilde{r}, \epsilon)$;

  With probability

$$\alpha = 1 \wedge \frac{\exp\{-H(\tilde{\theta}, \tilde{r})\}}{\exp\{-H(\theta_{k-1}, r_{k-1})\}} = 1 \wedge \frac{\exp\{-U(\tilde{\theta}) - \frac{1}{2}\tilde{r}^\top \tilde{r}\}}{\exp\{-U(\theta_{k-1}) - \frac{1}{2} r_{k-1}^\top r_{k-1}\}},$$

  accept $\theta_k \leftarrow \tilde{\theta}$, $r_k \leftarrow -\tilde{r}$.

HMC parameters

- What if ϵ is too large?

HMC parameters

- What if ϵ is too large?
- Acceptance rate is low, and the performance degrades;

HMC parameters

- What if ϵ is too large?
- Acceptance rate is low, and the performance degrades;
- What if ϵ is too small?

HMC parameters

- What if ϵ is too large?
- Acceptance rate is low, and the performance degrades;
- What if ϵ is too small?
- Same problems as ULA, HMC becomes computationally costly and produces correlated particles (can be partially compensated with L);
- Demo: `https://chi-feng.github.io/mcmc-demo`

Recap: Importance Sampling procedure

- Aim: sample from π and estimate $\pi(f) = \int_{\mathbb{R}^D} f(x)\pi(\mathrm{d}x)$;
- π is known up to a normalizing factor Z_Π, $\pi(\mathrm{d}x) = \tilde{\pi}(\mathrm{d}x)/\, Z_\Pi$;
- Importance Sampling (IS) consists of re-weighting samples from a proposal distribution Λ.
- Define *importance weights* as $\tilde{w}(x) = \tilde{\pi}(x)/\lambda(x)$;
- The *self-normalized importance sampling* (SNIS) estimator of $\pi(f)$ is then given by

$$\widehat{\pi}_N(f) = \sum_{i=1}^{N} \omega_N^i f(X^i)\,,$$

where

$$X^{1:N} \sim \Lambda\,, \omega_N^i = \frac{\tilde{w}(X^i)}{\sum_{j=1}^N \tilde{w}(X^j)}\,, i \in \{1, \ldots, N\}\,.$$

Iterated SIR (i-SIR) algorithm

Iterating samples from Λ, we arrive at iterated SIR algorithm (i-SIR , Andrieu et al. [2010], and Andrieu et al. [2018]).

Algorithm 2: Single stage of i-SIR algorithm

Input : Sample Y_j from previous iteration
Output: New sample Y_{j+1}

Set $X_{j+1}^1 = Y_j$ and draw $X_{j+1}^{2:N} \sim \Lambda$.
for $i \in [N]$ **do**
  compute the normalized weights $\omega_{i,j+1} = \tilde{w}(X_{j+1}^i) / \sum_{k=1}^N \tilde{w}(X_{j+1}^k)$.
Set $I_{j+1} = \mathrm{Cat}(\omega_{1,j+1}, \ldots, \omega_{N,j+1})$.
Draw $Y_{j+1} = X_{j+1}^{I_{j+1}}$.

Iterated SIR (i-SIR) algorithm

Iterating samples from Λ, we arrive at iterated SIR algorithm (i-SIR , Andrieu et al. [2010], and Andrieu et al. [2018]).

Algorithm 3: Single stage of i-SIR algorithm

Input : Sample Y_j from previous iteration
Output: New sample Y_{j+1}

Set $X_{j+1}^1 = Y_j$ and draw $X_{j+1}^{2:N} \sim \Lambda$.
for $i \in [N]$ **do**
  compute the normalized weights $\omega_{i,j+1} = \tilde{w}(X_{j+1}^i) / \sum_{k=1}^N \tilde{w}(X_{j+1}^k)$.
Set $I_{j+1} = \mathrm{Cat}(\omega_{1,j+1}, \ldots, \omega_{N,j+1})$.
Draw $Y_{j+1} = X_{j+1}^{I_{j+1}}$.

From i-SIR to Ex^{2}MCMC algorithm

- Main i-SIR drawback: absence of local exploration moves;
- Idea: apply a local MCMC kernel R (*rejuvenation kernel*) after each i-SIR step;
- R has π as invariant distribution;
- Here comes Ex^{2}MCMC : Exploration steps through i-SIR , Exploitation steps through $\mathrm{R}(x, \cdot)$;
- As our default choice we consider MALA as rejuvenation, but other ones (HMC, NUTS) are also possible.

Ex^{2}MCMC algorithm

Algorithm 4: Single stage of Ex^{2}MCMC algorithm with independent proposals

Procedure Ex^{2}MCMC $(Y_j, \Lambda, \mathrm{R})$**:**

Input : Previous sample Y_j;
proposal distribution Λ;
rejuvenation kernel R;

Output: New sample Y_{j+1};

Set $X^1_{j+1} = Y_j$, draw $X^{2:N}_{j+1} \sim \Lambda$;

for $i \in [N]$ **do**

compute the normalized weights $\omega_{i,j+1} = \tilde{w}(X^i_{j+1}) / \sum_{k=1}^N \tilde{w}(X^k_{j+1})$;

Set $I_{j+1} = \mathrm{Cat}(\omega_{1,j+1}, \ldots, \omega_{N,j+1})$;

Draw $Y_{j+1} \sim \mathrm{R}(X^{I_{j+1}}_{j+1}, \cdot)$.

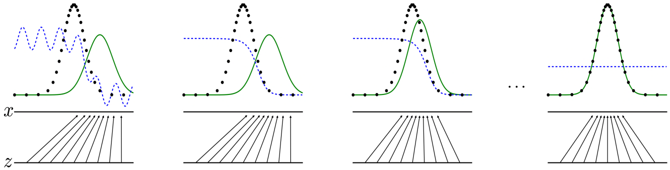
GANs framework

- Generator $G : \mathbb{R}^d \mapsto \mathbb{R}^{\mathrm{D}}$: takes a latent variable z from a prior density $p_0(z)$, $z \in \mathbb{R}^d$, produces $G(z) \in \mathbb{R}^{\mathrm{D}}$ in the observation space;
- Discriminator $D : \mathbb{R}^{\mathrm{D}} \mapsto [0, 1]$: takes a sample in the observation space, distinguishes between real examples and fake ones;

GAN training objective

$$L(g, D) := \mathbb{E}_{X \sim p_{\mathrm{data}}}[\log(D(X))] + \mathbb{E}_{Z \sim p_0}[\log(1 - D(g(Z)))] \to \min_{g \in \mathcal{G}} \max_{D \in \mathcal{D}} .$$

- Let $p_d(x)$ and $p_g(x)$ be the densities of real and fake observations;



-

$$\text{Optimal discriminator: } D^\star(x) = \frac{p_d(x)}{p_d(x) + p_g(x)} \tag{3}$$

GANs as an energy-based model

- Main drawback: information accumulated by discriminator is not used during the generation procedure;
- Let $d^\star(x) = \operatorname{logit} D^\star(x)$, therefore:

$$\frac{p_d(x)}{p_d(x) + p_g(x)} = \frac{1}{1 + \frac{p_g(x)}{p_d(x)}} = \frac{1}{1 + \exp(-d^\star(x))}$$

  Hence, we can express

$$p_d(x) = p_g(x) e^{d^\star(x)} .$$

- Let us introduce $d(x) = \operatorname{logit} D(x)$ and consider the corresponding energy-based model

$$\hat{p}_d(x) = p_g(x) e^{d(x)} / Z_0 ,$$

  where Z_0 is the normalizing constant. If $D(x) \approx D^\star(x)$, $\hat{p}_d(x)$ is close to $p_d(x)$;
- Sample from $\hat{p}_d(x)$ using MCMC.

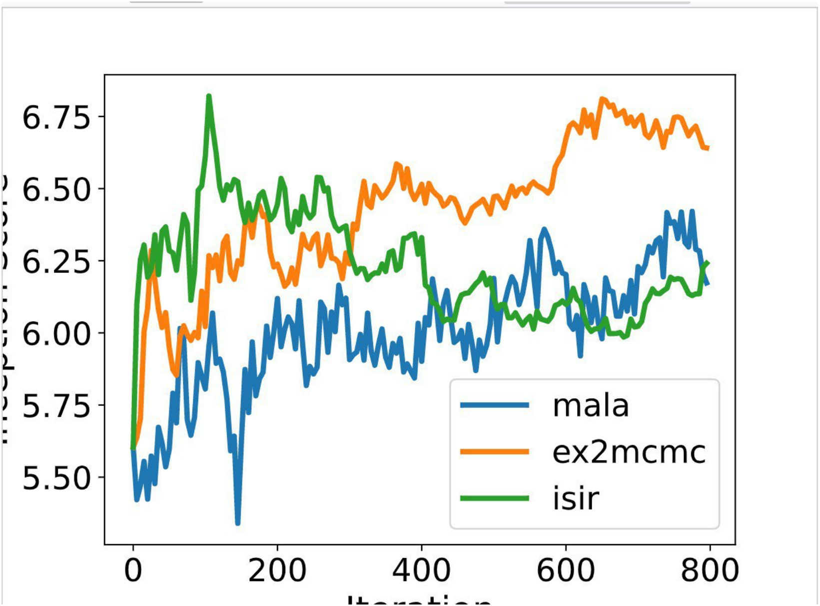
GANs as an energy-based model

- Similar idea considered in Turner et al. [2019]; main issue: MCMC in pixel space is highly inefficient;
- Che et al. [2020] suggested latent-space sampling from the model

$$\hat{p}_d(z) = p_0(z) \exp\left\{\operatorname{logit}\big(D(G(z))\big)\right\}, z \in \mathbb{R}^d,$$

  where $p_0(z)$ is the generator's prior distribution in the latent space;
- Sampling using Langevin-based algorithms, as suggested in Che et al. [2020], can be inefficient, especially if d is large.

IS metrics on CIFAR-10



Issues?

What are the potential issues of Ex^2MCMC ?

Issues?

What are the potential issues of $\mathrm{Ex}^2\mathrm{MCMC}$?

Bad global proposals

Recall that we still have not anything on the proposal distribution λ. At the same time, for poor λ, the acceptance rate will degrade quickly with dimension...

Adaptive proposals

- Consider family of proposals $\{\lambda_\theta\}, \theta \in \mathbb{R}^D$, chosen to match the target distribution $\tilde{\pi}$;
- Let $T\colon \mathbb{R}^d \to \mathbb{R}^d$ be smooth and invertible. Denote by $[T][\Lambda]$ the distribution of $Y = T(X)$ with $X \sim \lambda$;
- The corresponding density is given by $\lambda_T(y) = \lambda\big(T^{-1}(y)\big)\, \mathrm{J}_{T^{-1}}(y)$, where J_T denotes the Jacobian determinant of T;

Adaptive proposals: learning procedure

- Disperancy measure: linear combination of forward and backward KL divergence (generalizations to [Papamakarios et al., 2021] possible);
- Forward and backward KL:

$$\mathcal{L}^f(\theta) = \int \log \frac{\pi(x)}{\lambda_\theta(x)} \pi(x) \mathrm{d}x \,,$$

$$\mathcal{L}^b(\theta) = \int \log \frac{\lambda(x)}{\pi\big(T_\theta(x)\big) \, \mathrm{J}_{T_\theta}(x)} \lambda(x) \mathrm{d}x \,.$$

- Given a sample $Y_k \sim \pi$ and $Z_k \sim \lambda$ for $k \in [K]$, by

$$\widehat{\nabla \mathcal{L}^f}(Y^{1:K}, \theta) = -\frac{1}{K} \sum_{k=1}^{K} \nabla \log \lambda_\theta(Y_k) \,,$$

$$\widehat{\nabla \mathcal{L}^b}(Z^{1:K}, \theta) = -\frac{1}{K} \sum_{k=1}^{K} \nabla \log \big(\tilde{\pi}(T_\theta(Z_k) \, \mathrm{J}_{T_\theta}(Z_k)\big) .$$

- Following Gabrié et al. [2021], we consider

$$\widehat{\mathcal{L}}(Y^{1:K}, Z^{1:K}, \theta) = \alpha \widehat{\mathcal{L}^f}(Y^{1:K}, \theta) + \beta \widehat{\mathcal{L}^b}(Z^{1:K}, \theta) \,.$$

FIEx^{2}MCMC algorithm with adaptive proposals

Algorithm 5: Single stage of FIEx^{2}MCMC. Steps of Ex^{2}MCMC are done in parallel with common values of proposal parameters θ_j. Step 4 updates the parameters using the gradient estimate obtained from all the chains.

Input : weights θ_j, batch $Y_j^{1:K}$

Output: new weights θ_{j+1}, batch $Y_{j+1}^{1:K}$

1 **for** $k \in [K]$ **do**

2 $\quad Y_{j+1,k} = \mathrm{Ex}^2\mathrm{MCMC}\ (Y_{j,k}, [T_{\theta_j}][\Lambda], \mathrm{R})$

3 Draw $\bar{Z}^{1:K} \sim \Lambda$.

4 Update $\theta_{j+1} = \theta_j - \gamma \widehat{\nabla \mathcal{L}}(Y_{j+1}, \bar{Z}, \theta_j)$.

Practical note

In our experiments: T_θ is modelled as a normalizing flow based on RealNVP architecture (Dinh et al. [2017]).

Normalizing flows

- Suppose that we observe $X_1, \ldots, X_n \in X$ - independent random variables with unknown probability distribution $\pi(x)$;
- Is there a way to sample from $\pi(x)$ or evaluate $\pi(x)$?
- One possible solution - Real-valued non-volume preserving normalizing flows (Real NVP), special case of normalizing flows.

Changing variable

- Consider the so-called latent space Z, typically $\dim(Z) < \dim(X)$ and a simple probability density $q(z), z \in Z$.
- Let $f : X \to Z$ be a bijection with $g = f^{-1}$;
- Using the change of variable formula, for $x \in X$:

$$\pi(x) = q\big(f(x)\big) \left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|$$

or, equivalently,

$$\log\left(\pi(x)\right) = \log\Big(q\big(f(x)\big)\Big) + \log\left(\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right),$$

where $\frac{\partial f(x)}{\partial x^T}$ is the Jacobian of f at x.

Training and Sampling

- Given a class of transformations $\mathcal{G}$, we maximize the likelihood

$$\sum_{i=1}^{n} \log\Big(q\big(f(X_i)\big)\Big) + \log\left(\left|\det\left(\frac{\partial f(X_i)}{\partial x^T}\right)\right|\right) \to \max_{f \in \mathcal{G}}$$

- Samples from the resulting distribution can be generated with the inverse transform. We only need to sample $Z_1, \ldots, Z_m$ with the distribution $q(z)$, then $f^{-1}(Z_1), \ldots, f^{-1}(Z_m)$ is an approximate sample from π.

Thank you!

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342, 2010.

Christophe Andrieu, Anthony Lee, Matti Vihola, et al. Uniform ergodicity of the iterated conditional SMC and geometric ergodicity of particle Gibbs samplers. *Bernoulli*, 24(2):842–872, 2018.

Tong Che, Ruixiang ZHANG, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is Secretly an Energy-based Model and You Should Use Discriminator Driven Latent Sampling. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12275–12287. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/90525e70b7842930586545c6f1c9310c-Paper.pdf.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Marylou Gabrié, Grant M. Rotskoff, and Eric Vanden-Eijnden. Adaptive Monte Carlo augmented with normalizing flows. *arXiv preprint arXiv:2105.12603*, 2021.

Matthew D Hoffman, Andrew Gelman, et al. The no-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, pages 113–162, 2011.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Ryan Turner, Jane Hung, Eric Frank, Yunus Saatchi, and Jason Yosinski. Metropolis-Hastings generative adversarial networks. In *International Conference on Machine Learning*, pages 6345–6353. PMLR, 2019.