Gradient-Based MCMC

CSC 412 Tutorial
March 2, 2017

Jake Snell

Many slides borrowed from: Iain Murray, MLSS '09*

http://homepages.inf.ed.ac.uk/imurray2/teaching/09mlss/slides.pdf

Overview

- Review of Markov Chain Monte Carlo (MCMC)
- Metropolis algorithm
- Metropolis-Hastings algorithm
- Langevin Dynamics
- Hamiltonian Monte Carlo
- Gibbs Sampling (time permitting)

Simple Monte Carlo

Statistical sampling can be applied to any expectation:

In general:

$$\int f(x)P(x)\,\mathrm{d}x \;\approx\; \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}), \quad x^{(s)} \sim P(x)$$

Example: making predictions

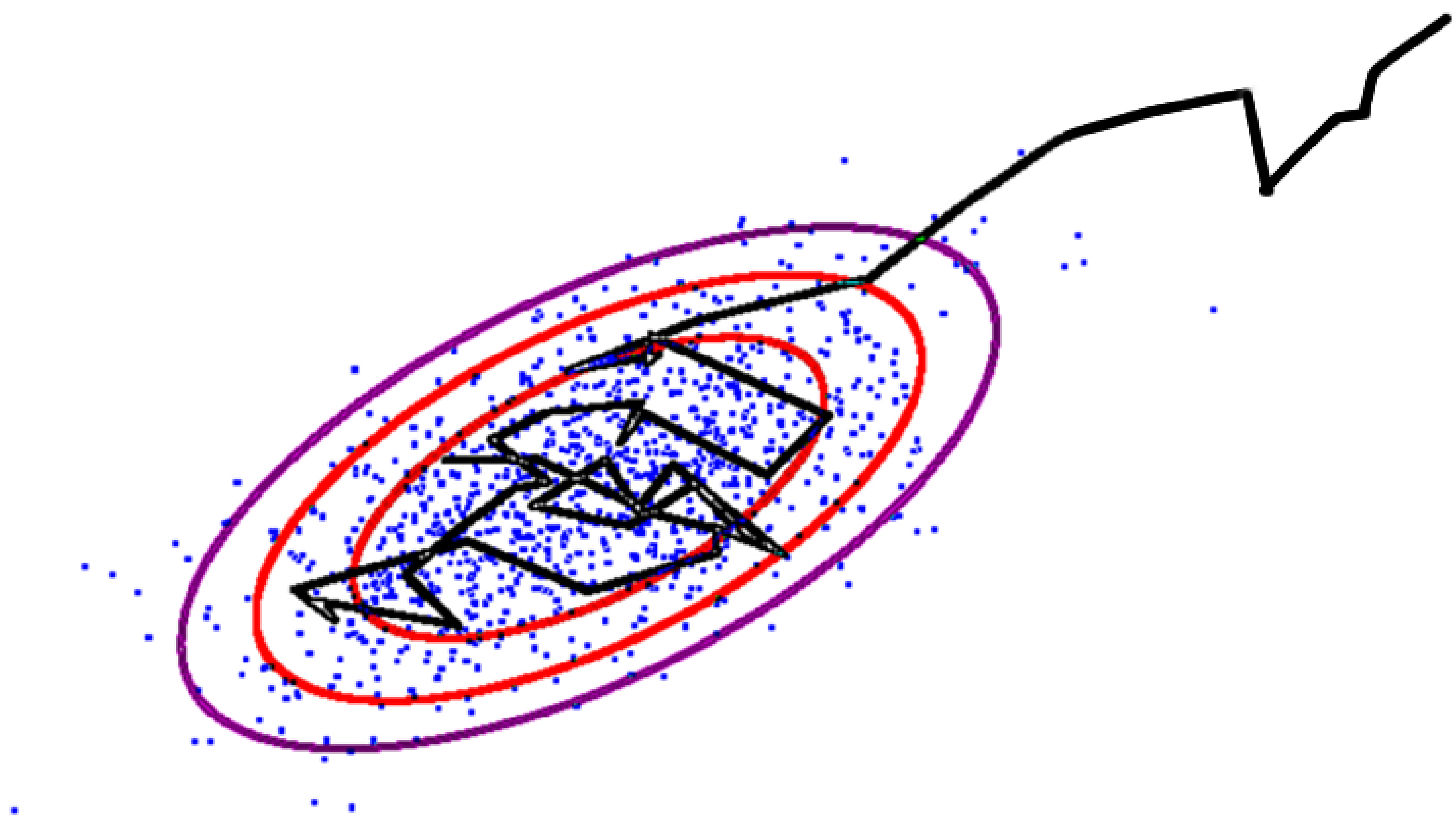
$$\begin{aligned} p(x|\mathcal{D}) \;&=\; \int P(x|\theta,\mathcal{D})P(\theta|\mathcal{D})\,\mathrm{d}\theta \\ &\approx\; \frac{1}{S}\sum_{s=1}^{S} P(x|\theta^{(s)},\mathcal{D}), \quad \theta^{(s)} \sim P(\theta|\mathcal{D}) \end{aligned}$$

More examples: E-step statistics in EM, Boltzmann machine learning

Markov chain Monte Carlo

Construct a biased random walk that explores target dist $P^\star(x)$

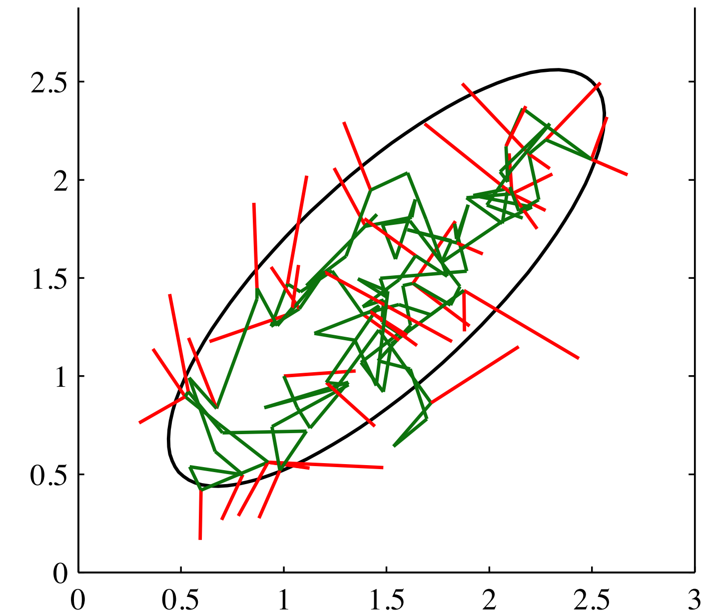
Markov steps, $x_t \sim T(x_t \leftarrow x_{t-1})$



MCMC gives approximate, correlated samples from $P^\star(x)$

Metropolis algorithm

- Perturb parameters: $Q(\theta'; \theta)$, e.g. $\mathcal{N}(\theta, \sigma^2)$
- Accept with probability $\min\left(1, \dfrac{\tilde{P}(\theta'|\mathcal{D})}{\tilde{P}(\theta|\mathcal{D})}\right)$
- Otherwise **keep old parameters**



This subfigure from PRML, Bishop (2006)

Detail: Metropolis, as stated, requires $Q(\theta'; \theta) = Q(\theta; \theta')$

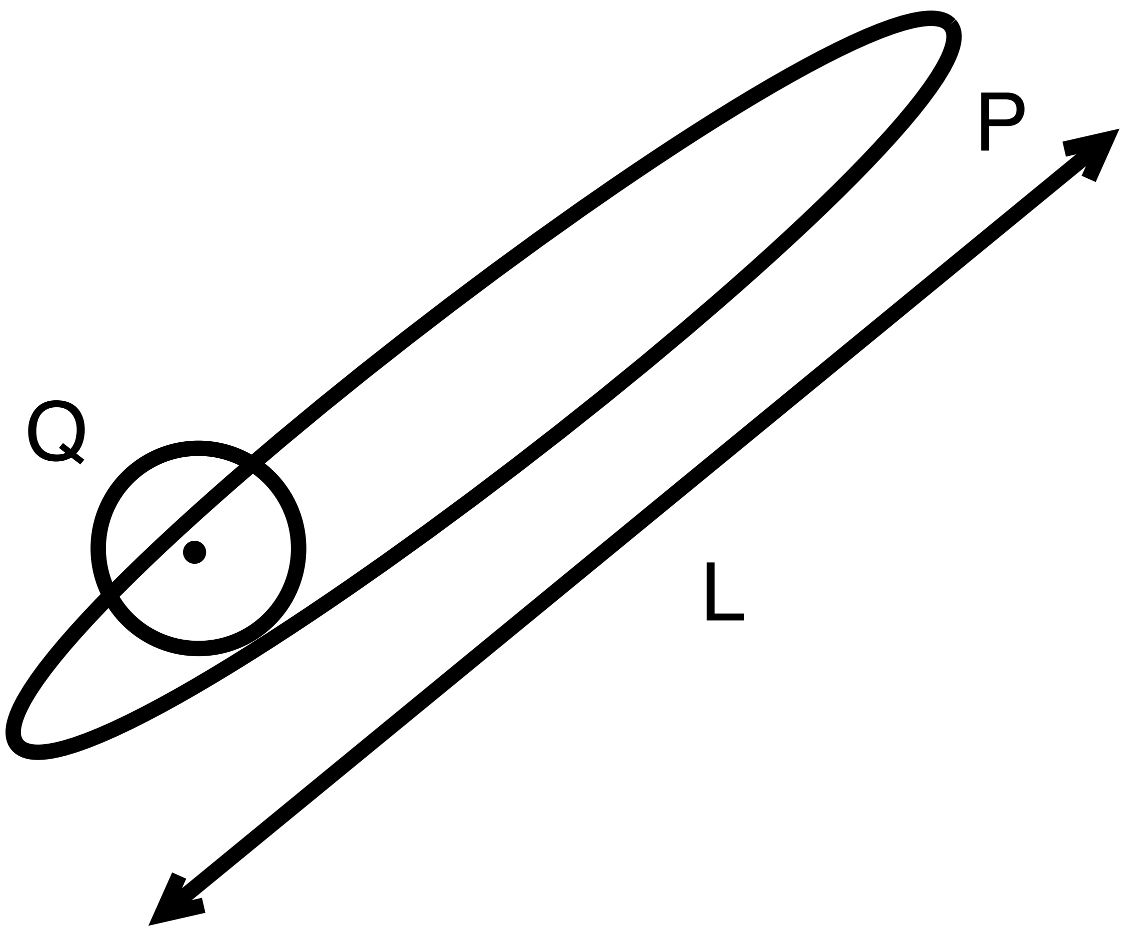
After n steps (where n is large), $\theta_n \approx P(\theta|\mathcal{D})$

Metropolis Demo in 1D

```python
# Code ported from slides by Iain Murray, MLSS 2009
# http://homepages.inf.ed.ac.uk/imurray2/teaching/09mlss/slides.pdf
def metropolis(x_init, log_ptilde, iters, sigma):
    samples = np.zeros(iters)
    n_accept = 0

    # initialize current state and unnormalized log prob
    x = x_init
    log_p_x = log_ptilde(x)

    for i in xrange(iters):
        # make proposal
        prop = x + sigma * np.random.randn()
        log_p_prop = log_ptilde(prop)

        # compute acceptance probability
        p_accept = np.minimum(1., np.exp(log_p_prop - log_p_x))

        if np.random.rand() < p_accept:
            # accept
            x = prop
            log_p_x = log_p_prop
            n_accept += 1

        samples[i] = x

    return samples, 1. * n_accept / iters
```

http://nbviewer.jupyter.org/gist/jakesnell/aea6284fd6f102bdc54648c03566c48d

Metropolis limitations



Generic proposals use
$Q(x'; x) = \mathcal{N}(x, \sigma^2)$

σ large $\rightarrow$ many rejections

σ small $\rightarrow$ slow diffusion:
$\sim(L/\sigma)^2$ iterations required

Optimal σ is “just right”: acceptance rate far from 0 and 1

Metropolis Hastings algorithm

MH is defined as follows:

Sample $x' \sim Q(x'|x)$

Compute $$p = \min\left(1, \frac{\tilde{P}(x')Q(x|x')}{\tilde{P}(x)Q(x'|x)}\right)$$

With probability p, set $x \leftarrow x'$

Repeat

MH gives us flexibility to choose an asymmetric proposal distribution, where $Q(x'|x) \neq Q(x|x')$

Recover Metropolis as a special case if symmetric

Valid MCMC operators

Define *transition probabilities* $T(x' \leftarrow x) = P(x'|x)$

Marginals: $P(x') = \sum_x P(x'|x)P(x)$

A transition distribution is *invariant*, *or* *stationary*, wrt a Markov chain if each step leaves that distribution invariant

So the target distribution is invariant if *TP* = P**

$$\sum_x T(x' \leftarrow x)P^*(x) = P^*(x')$$

Also, need to show that distribution converges to required invariant distribution for any initial distribution: *ergodic*

Then *P** is called the *equilibrium distribution*

Detailed balance

Detailed balance means that $\rightarrow x \rightarrow x'$ and $\rightarrow x' \rightarrow x$ are equally probable

$$T(x' \leftarrow x)\, P^*(x) = T(x \leftarrow x')\, P^*(x')$$

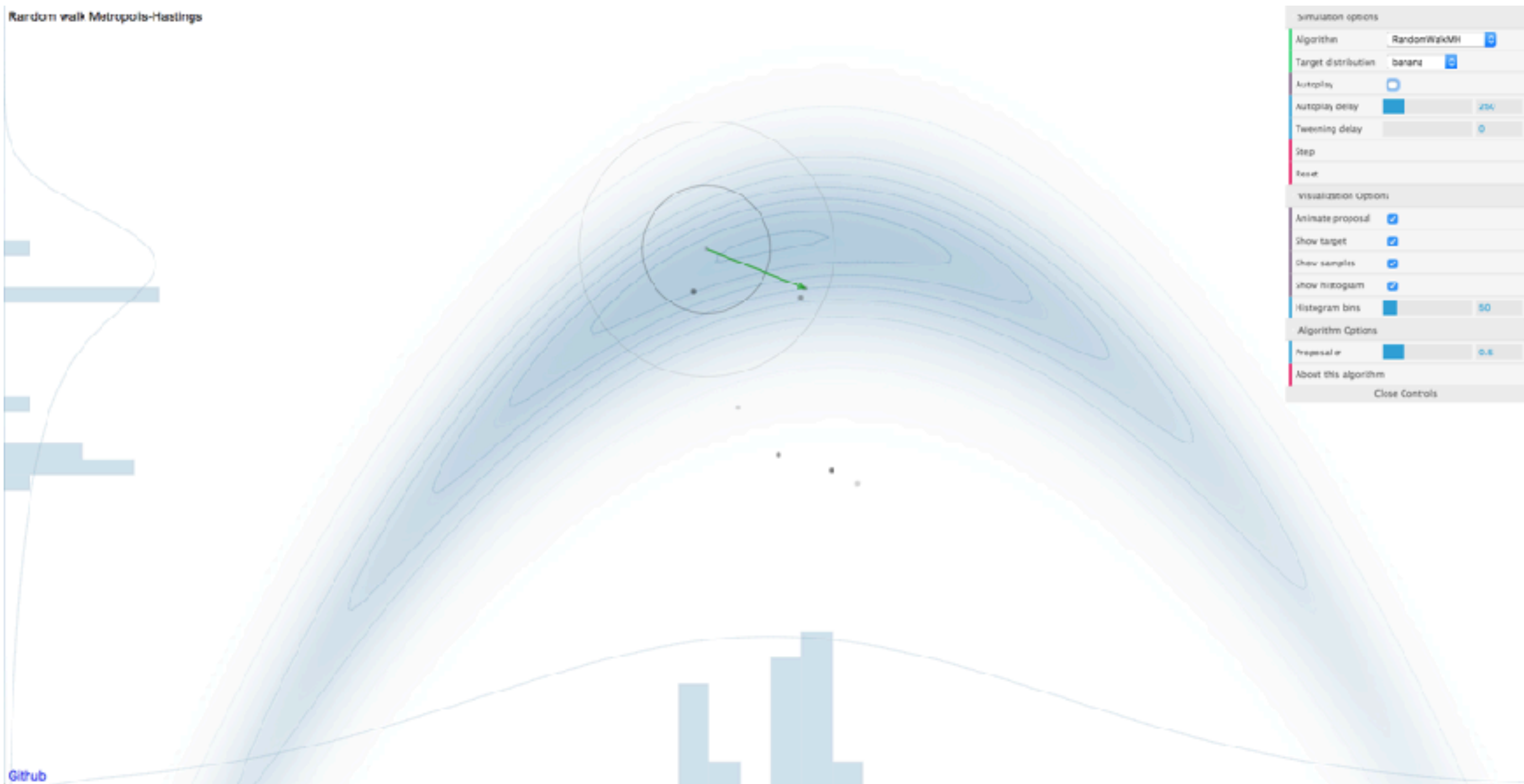
Detailed balance implies the invariant condition

$$\sum_x T(x' \leftarrow x) P^*(x) = \sum_x T(x \leftarrow x') P^*(x') = P^*(x') \sum_x P(x|x') = P^*(x')$$

A Markov chain that respects detailed balance is *reversible*

To show that P is an invariant distribution can show that detailed balance is satisfied*

Metropolis-Hastings Demo



https://chi-feng.github.io/mcmc-demo/app.html#RandomWalkMH,banana

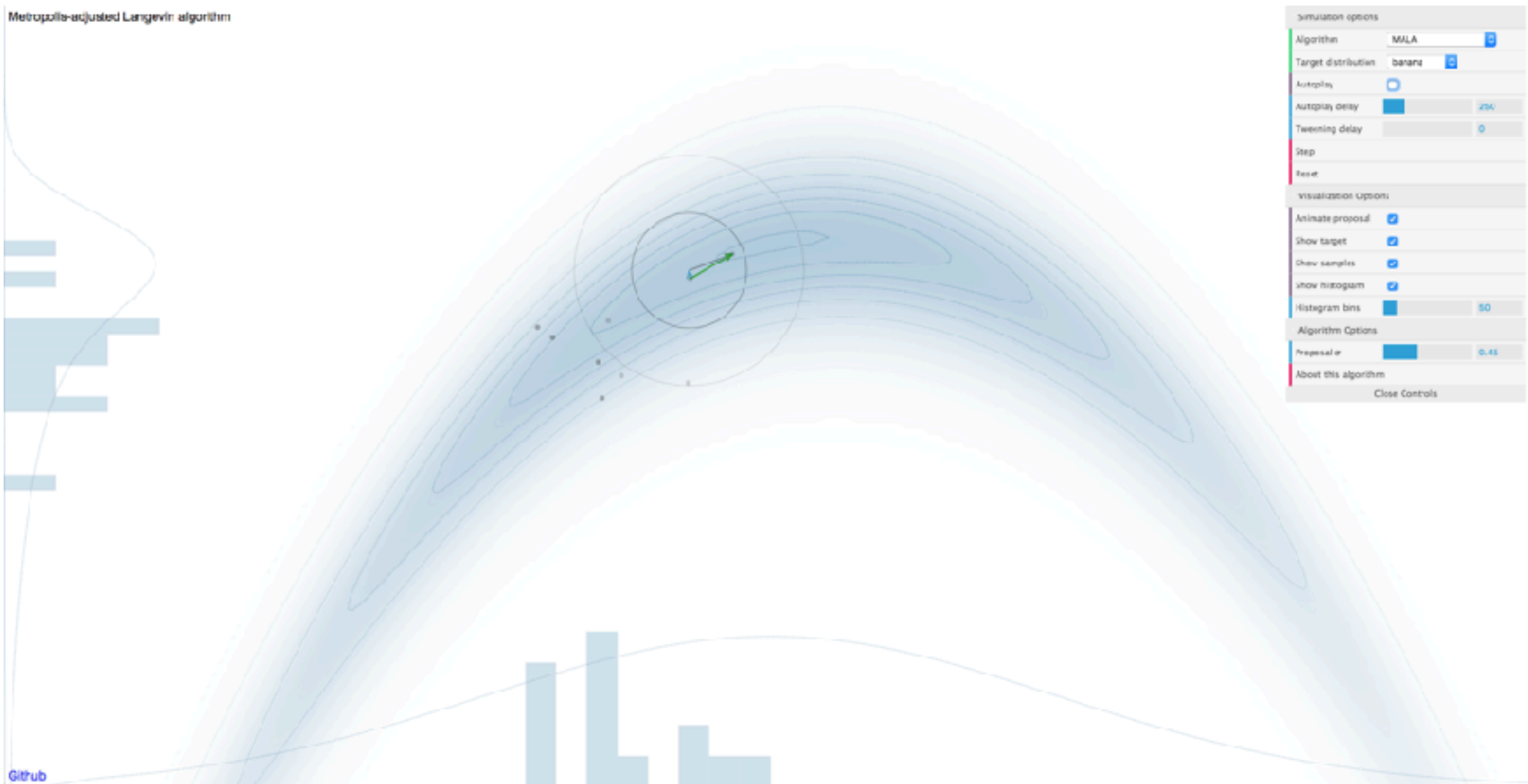
Langevin Dynamics

- Use proposal distribution

$$Q(x'|x) = \mathcal{N}\left(x + \frac{1}{2}\sigma^2 \nabla \log p(x), \sigma^2 I\right)$$

- Special case of MH
- Tries to move in directions of increasing $\tilde{P}$
- Looks a lot like SGD with noise!

Langevin Demo



https://chi-feng.github.io/mcmc-demo/app.html#MALA,banana

Auxiliary variables

The point of MCMC is to marginalize out variables, but one can introduce more variables:

$$\int f(x)P(x)\,\mathrm{d}x = \int f(x)P(x,v)\,\mathrm{d}x\,\mathrm{d}v$$
$$\approx \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}), \quad x,v \sim P(x,v)$$

We might want to do this if

- $P(x|v)$ and $P(v|x)$ are simple
- $P(x,v)$ is otherwise easier to navigate

Hamiltonian Monte Carlo

Define a joint distribution:

- $P(x, v) \propto e^{-E(x)} e^{-K(v)} = e^{-E(x)-K(v)} = e^{-H(x,v)}$
- Velocity is independent of position and Gaussian distributed

Markov chain operators

- Gibbs sample velocity
- Simulate Hamiltonian dynamics then flip sign of velocity
  - Hamiltonian 'proposal' is deterministic and reversible $q(x', v'; x, v) = q(x, v; x', v') = 1$
  - Conservation of energy means $P(x, v) = P(x', v')$
  - Metropolis acceptance probability is 1

Except we can't simulate Hamiltonian dynamics exactly

Leap-frog dynamics

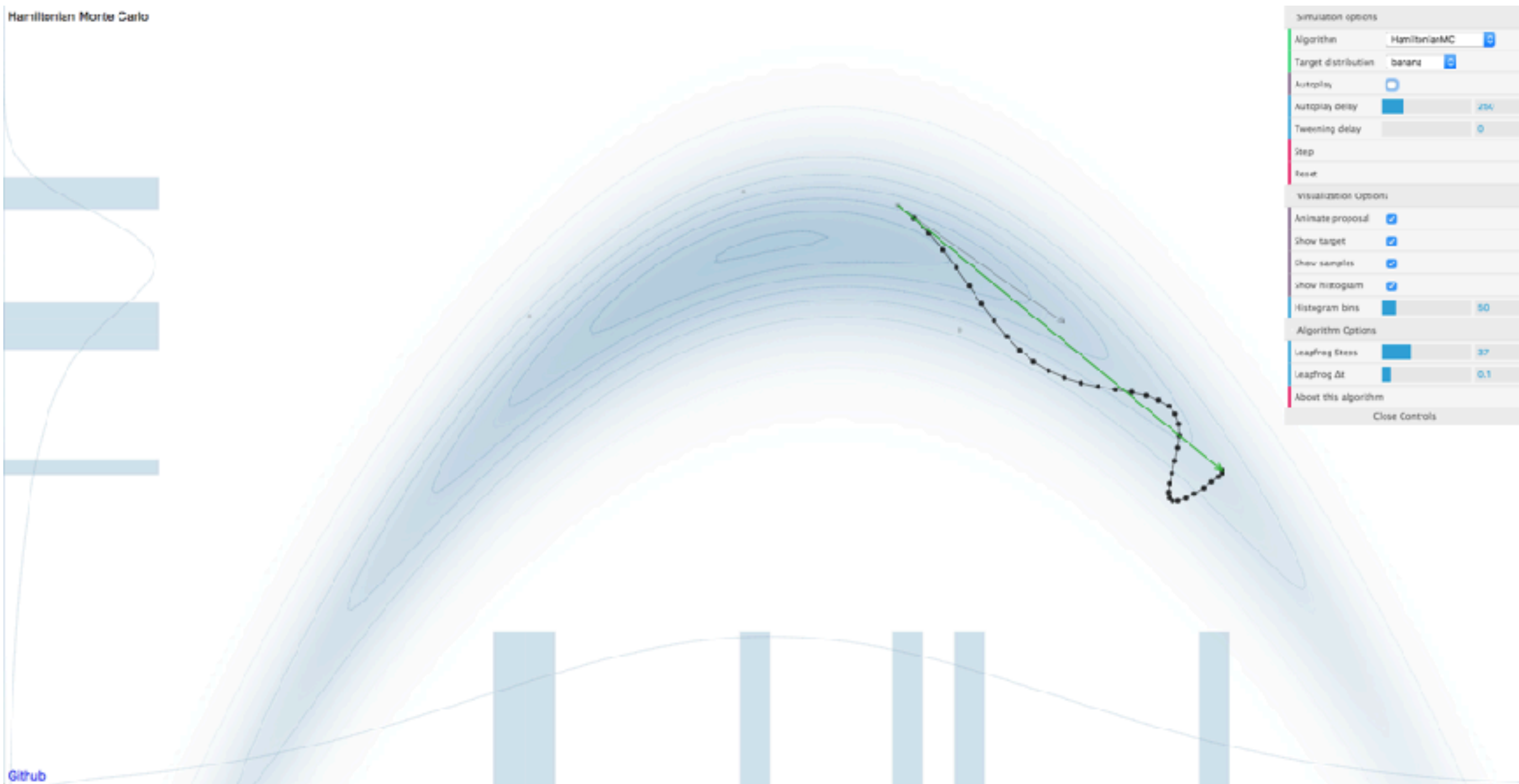
a discrete approximation to Hamiltonian dynamics:

$$
\begin{aligned}
v_i(t + \tfrac{\epsilon}{2}) &= v_i(t) - \frac{\epsilon}{2} \frac{\partial E(x(t))}{\partial x_i} \\
x_i(t + \epsilon) &= x_i(t) + \epsilon\, v_i(t + \tfrac{\epsilon}{2}) \\
p_i(t + \epsilon) &= v_i(t + \tfrac{\epsilon}{2}) - \frac{\epsilon}{2} \frac{\partial E(x(t + \epsilon))}{\partial x_i}
\end{aligned}
$$

- H is not conserved
- dynamics are still deterministic and reversible
- Acceptance probability becomes $\min[1, \exp(H(v, x) - H(v', x'))]$

Looks a lot like SGD + momentum followed by accept step

HMC Demo



https://chi-feng.github.io/mcmc-demo/app.html#HamiltonianMC,banana

Summary

We need approximate methods to solve sums/integrals

Monte Carlo does not explicitly depend on dimension, although simple methods work only in low dimensions

Markov chain Monte Carlo (MCMC) can make local moves. By assuming less it is more applicable to higher dimensions

It produces approximate, correlated samples

Simple computations → easy to implement