**CS 329M: Topics in Artificial Intelligence** **Spring 2017**

# Lecture 4: April 24

*Lecturer: James Zou* *Scribe: Boxiang Liu, Evan Patterson, Paulo Orenstein, Teng Zhang*

Having studied spectral theory, we now turn to studying methods that allow us to sample from different distributions. That is useful in many instances, for example when we want to sample from the data distribution, or when we want to do inference, particularly within the realm of Bayesian statistics, where we would like to sample from distributions in parameter space. This week we'll discuss Hamiltonian Monte Carlo, and next week we'll talk about variational inference, two important methods in this setting.

The main question we are interested in is how to investigate or sample from very complicated distributions. Oftentimes, for example, we are interested in computing expectations, which we can approximate via

$$\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i), \qquad X_i \sim X,$$

so as long as we can simulate random variables $X_i$, we can obtain a good approximation to any expected value we care about.

## 4.1 Metropolis-Hastings

The classical algorithm that allows us to sample from a complex distribution is Metropolis-Hastings. The goal is to sample from a distribution $p(x)$ when we can only evaluate $p(x)$ up to a constant.

The method takes as input a proposal function $g(x' \mid x)$, which is usually taken to be a Gaussian, say $g(x' \mid x) \sim N(x, 1)$. The algorithm then runs as follows:

1. Initialize $x^{(0)}$;
2. For each $t = 1, \ldots, T$, propose $x' \sim g(x' \mid x^{(t-1)})$. Accept the proposal with probability

$$\min\left\{1, \frac{p(x')g(x^{(t-1)} \mid x')}{p(x^{(t-1)})g(x' \mid x^{(t-1)})}\right\},$$

   by making $x^{(t)} = x'$. Otherwise, reject the proposal and set $x^{(t)} = x^{(t-1)}$.

To pick the number $T$ of runs, we usually look at the autocorrelation function on the samples drawn, until the samples look like they are coming from a fixed distribution, namely $p(x)$.

On the one hand, Metropolis-Hastings is guaranteed to give exact samples from $p(x)$ (as $T \to \infty$). This is because the algorithm is designed to make the target distribution $p(x)$ invariant, that is, if $K(\cdot \mid \cdot)$ is the Metropolis-Hastings kernel, then

$$p(y) = \int K(y \mid x)p(x)dx.$$

On the other hand, besides requiring the ability to evaluate $p(x)$ fast, the Metropolis-Hastings algorithm needs a proposal function, which usually requires knowledge about $p(x)$ which might not be available.

In fact, this is also the reason why Metropolis-Hastings doesn't work in high-dimensions. Indeed, higher-dimensional spaces make multimodality a likely thing, which usually wrecks Metropolis-Hastings, since that requires moving through a space of very low probability.

A bigger problem, however, is that in high-dimensional space, as we saw in Lecture 1, most neighborhoods in the space have very low probability, and such neighborhoods are so common that Metropolis-Hastings ends up not exploring a significant part of the distribution. This significant part, where most probability mass is concentrated, is usually known as the 'typical neighborhood', and is a result of the tension between areas that have high probability density but low volume (such as the center of a Gaussian distribution), and areas that have low probability density but high volume (closer to the tails of a Gaussian distribution). An effective high-dimensional MCMC algorithm should be able to explore this typical set fairly well, and that is not the case for Metropolis-Hastings.

## 4.2 Hamiltonian Monte Carlo

The main improvement in Hamiltonian Monte Carlo to overcome the Metropolis-Hastings limitations is to introduce an auxiliary variable, so we try to sample from $p(x, y)$. While increasing our sample space makes exploring this new, augmented space harder, we will only care about the marginal $p(x)$. As long as we can easily and efficiently sample in the augmented space, we will have a better sampler than Metropolis-Hastings.

To pick the this augmented variable $y$ we rely on some physics intuition. Define the Hamiltonian as

$$H(x, y) = E(x) + \frac{\|y\|^2}{2\sigma^2}$$

where $E(x)$ is the potential energy, or the negative log-likelihood in statistics, so we define it via $p(x) \propto e^{-E(x)}$; in physical terms, we think of $x$ as the position and $y$ as the momentum. The Hamiltonian is just the addition of potential and kinetic energy, as well as the fact that they should be preserved under Newtonian mechanics. Indeed, the equations of motion are

$$\frac{dx}{dt} = \frac{\partial H}{\partial y}, \qquad \frac{dy}{dt} = -\frac{\partial H}{\partial x}$$

The solution to the equations, $(x(t), y(t))$, gives $H(x(t), y(t)) = \text{constant}$. Hence, we sample from

$$p(x, y) \propto e^{-H(x,y)} = e^{-E(x) - \frac{\|y\|^2}{2\sigma^2}}$$

The Hamiltonian Monte Carlo algorithm proceeds as follows:

1. Initialize $x^{(0)}, y^{(0)} \sim N(0, \sigma^2 I)$;
2. Initialize Hamiltonian equation at $(x^{(0)}, y^{(0)})$, and run it for time $t$ to obtain the solution $(x^{(0)}(t), y^{(0)}(t))$;
3. Set $x^{(1)} = x^{(0)}(t)$, $y^{(1)} = y^{(0)}(t)$;
4. Replace $y^{(1)} \sim N(0, \sigma^2 I)$;
5. Iterate, starting Hamiltonian equation at $(x^{(1)}, y^{(1)})$, advancing to the solution until we get to $(x^{(1)}(t), y^{(1)}(t))$, sample $y^{(2)}$, etc.

The main problem left is how to solve the Hamiltonian equations efficiently. The most commonly used algorithm is called the *Leapfrog method*. It works as follows:

1. $y\left(t+\frac{\varepsilon}{2}\right) = y(t) - \frac{\varepsilon}{2}\frac{\partial E}{\partial t}\Big|_t$;
2. $x(t+\varepsilon) = x(t) + \varepsilon\frac{y}{\sigma^2}\left(t+\frac{\varepsilon}{2}\right)$;
3. $y(t+\varepsilon) = y\left(t+\frac{\varepsilon}{2}\right) - \frac{\varepsilon}{2}\frac{\partial E}{\partial t}\Big|_{t+\varepsilon}$.

The main advantage of this algorithm in solving the Hamiltonian is its numerical stability.

Hence, putting everything together, the Hamiltonian Monte Carlo algorithm in full is:

1. Initialize $x^{(0)}, y^{(0)} \sim N(0, \sigma^2 I)$;
2. Leapfrog for $t$ iterations;
3. Accept $(x(t), y(t))$ with probability

$$\min\left\{1, \exp\left(H(x(t), y(t)) - H(x^{(0)}, y^{(0)})\right)\right\}$$

   Upon acceptance, set $x^{(1)} = x(t)$ and $y^{(1)} = y(t)$;
   Upon rejection, set $x^{(1)} = x^{(0)}$ and $y^{(1)} = y^{(0)}$;
4. Set $y^{(1)} \sim N(0, \sigma^2 I)$, and iterate.

There are also some similarities between Hamiltonian Monte Carlo and stochastic gradient descent (SGD). One can in fact view SGD as a "poor man's version" of Hamiltonian Monte Carlo, in which we only take one step ahead when solving the Hamiltonian equations. Specifically, if we ignore the rejection step, the resulting algorithm is called "Langevin dynamics", which is equivalent to SGD with noise.