

# COGS 205B: Computational Lab Skills for Cognitive Scientists I

---

Joachim Vandekerckhove

Spring 2021

## Part 7: Metropolis-Hastings sampling

---

# Metropolis-Hastings sampling

- A widely applicable algorithm for numerical integration is the **Metropolis-Hastings** (MH) algorithm

# Metropolis-Hastings sampling

- A widely applicable algorithm for numerical integration is the **Metropolis-Hastings** (MH) algorithm
- In the MH algorithm, we will randomly generate **candidate samples** from some simple distribution, and then decide to accept or reject the candidate

# Metropolis-Hastings sampling

- A widely applicable algorithm for numerical integration is the **Metropolis-Hastings** (MH) algorithm
- In the MH algorithm, we will randomly generate **candidate samples** from some simple distribution, and then decide to accept or reject the candidate
- MH algorithms need some customization and fine-tuning to be most efficient

# Metropolis-Hastings sampling

- A widely applicable algorithm for numerical integration is the **Metropolis-Hastings** (MH) algorithm
- In the MH algorithm, we will randomly generate **candidate samples** from some simple distribution, and then decide to accept or reject the candidate
- MH algorithms need some customization and fine-tuning to be most efficient
- One intuition for the MH algorithm is that it is a rejection sampler with an adaptive envelope

# Metropolis-Hastings sampler

Given a function  $f(\theta) \propto p(\theta|D)$ , and a symmetric **candidate generating distribution**  $Q(a|b) = Q(b|a)$ , a Metropolis-Hastings sampling algorithm proceeds as follows:

- 1 **Set**  $i \leftarrow 1$  and **choose**  $R \leftarrow 1000$
- 2 **Choose**, arbitrarily,  $\theta^{(0)}$
- 3 **Draw** a randomly selected  $\theta^c$  from  $Q(\theta|\theta^{(i-1)})$
- 4 **Compute** the acceptance ratio  $\alpha = \frac{f(\theta^c)}{f(\theta^{(i-1)})} = \frac{p(\theta^c|D)}{p(\theta^{(i-1)}|D)}$
- 5 **Draw** a randomly selected  $u$  from  $U(0, 1)$ . **If**  $\alpha > u$ , **set**  $\theta^{(i)} \leftarrow \theta^c$ , **otherwise set**  $\theta^{(i)} \leftarrow \theta^{(i-1)}$
- 6 **Set**  $i \leftarrow i + 1$ . **If**  $i \leq R$ , **return** to Step 3, **otherwise halt**

# Metropolis-Hastings sampler

- Because the accepted values may approach the correct distribution only slowly, we have to make sure the chain of samples has converged to a stationary sampling state before using the samples for inference



# Metropolis-Hastings sampler

- Because the accepted values may approach the correct distribution only slowly, we have to make sure the chain of samples has converged to a stationary sampling state before using the samples for inference
- Specifically, we will discard a number of initial samples known as the **burn-in**:

$$\hat{\beta}_0 = \frac{1}{R - B} \sum_{i=B}^R \beta_0^{(i)}$$

# Metropolis-Hastings sampler

- Because the accepted values may approach the correct distribution only slowly, we have to make sure the chain of samples has converged to a stationary sampling state before using the samples for inference
- Specifically, we will discard a number of initial samples known as the **burn-in**:

$$\hat{\beta}_0 = \frac{1}{R - B} \sum_{i=B}^R \beta_0^{(i)}$$

- We also repeat the procedure a few times with different values for  $\theta^{(0)}$  to ensure that the algorithm converges to the same stationary distribution. Several convergence statistics exist, with Geweke's and Gelman's  $\hat{R}$  being the most popular

# Sampling in MATLAB

## Exercise: Metropolis sampling in MATLAB

Write a new class called `Metropolis()`

The constructor should work by taking an anonymous function as input (call the property `Target` of type `function_handle` ), with other inputs such as parameters of the algorithm optional (with defaults)

In addition to the standard methods, `Metropolis()` should have at least a method `DrawSamples(N)` to draw  $N$  samples

You should assume that `TargetLogPdf` has exactly one input, but the input may be a vector

Finally, make sure the `disp()` method outputs something informative about the current state of the sampler