ENM 5310: Data-driven Modeling and Probabilistic Scientific Computing

Lecture #10: Markov Chain Monte Carlo

Markov Chain Monte Carlo

from *SIAM News*, Volume 33, Number 4

The Best of the 20th Century: Editors Name Top 10 Algorithms

By Barry A. Cipra

Algos is the Greek word for pain. *Algor* is Latin, to be cold. Neither is the root for *algorithm*, which stems instead from al-Khwarizmi, the name of the ninth-century Arab scholar whose book *al-jabr wa'l muqabalah* devolved into today's high school algebra textbooks. Al-Khwarizmi stressed the importance of methodical procedures for solving problems. Were he around today, he'd no doubt be impressed by the advances in his eponymous approach.

Some of the very best algorithms of the computer age are highlighted in the January/February 2000 issue of *Computing in Science & Engineering*, a joint publication of the American Institute of Physics and the IEEE Computer Society. Guest editors Jack Don-garra of the University of Tennessee and Oak Ridge National Laboratory and Fran-cis Sullivan of the Center for Comput-ing Sciences at the Institute for Defense Analyses put togeth-er a list they call the "Top Ten Algorithms of the Century."

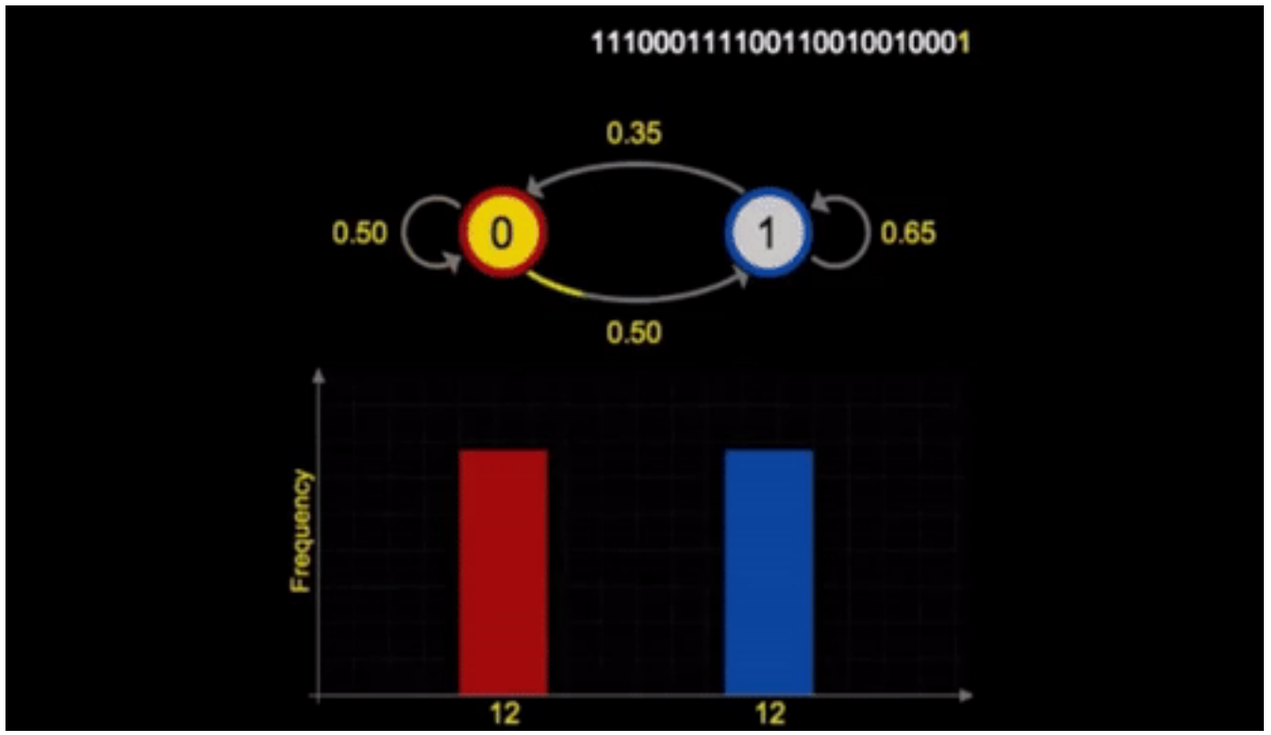
"We tried to assemble the 10 al-gorithms with the greatest influence on the development and practice of science and engineering in the 20th century," Dongarra and Sullivan write. As with any top-10 list, their selections—and non-selections—are bound to be controversial, they acknowledge. When it comes to picking the algorithmic best, there seems to be no best algorithm.

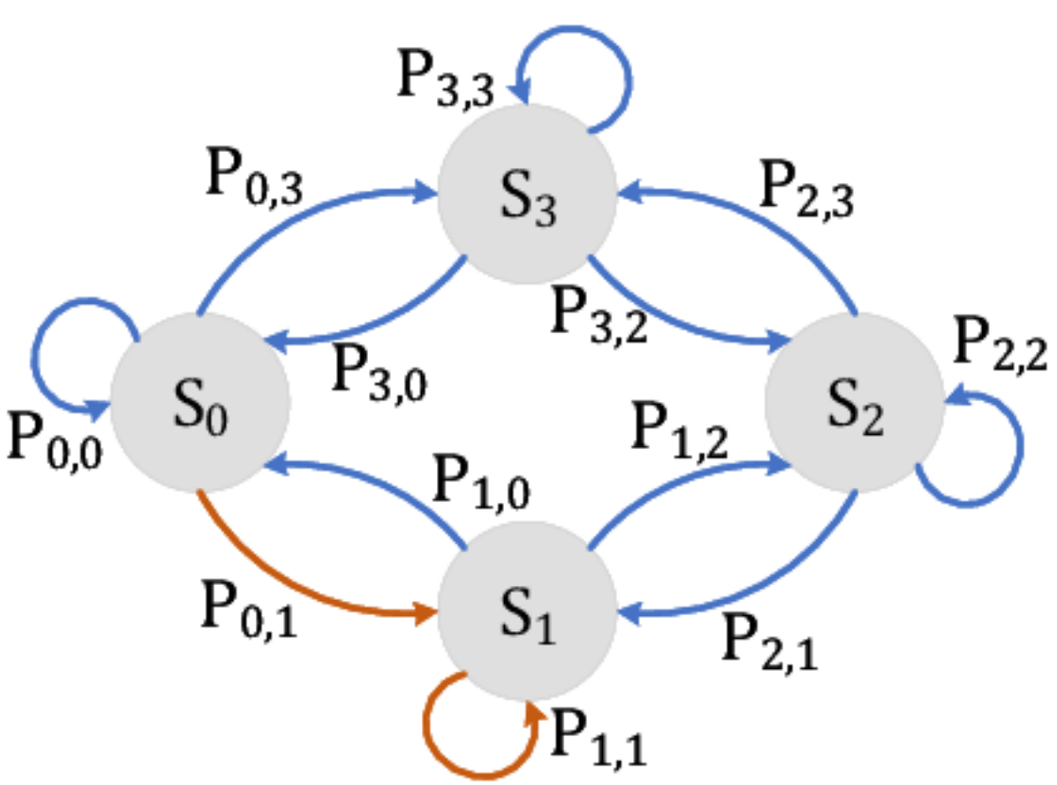
Without further ado, here's the CiSE top-10 list, in chronological order. (Dates and names associated with the algorithms should be read as first-order approximations. Most algorithms take shape over time, with many contributors.)

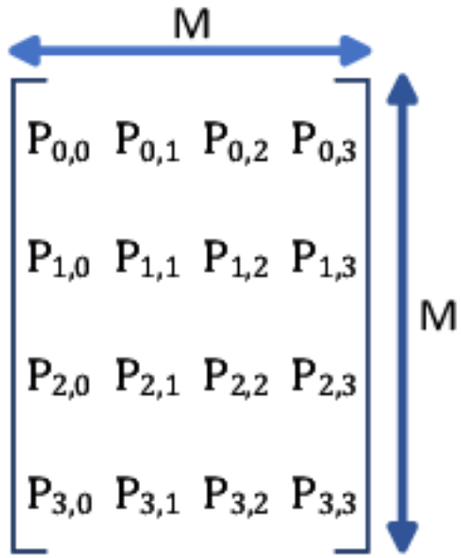
1946: John von Neumann, Stan Ulam, and Nick Metropolis, all at the Los Alamos Scientific Laboratory, cook up the Metropolis algorithm, also known as the **Monte Carlo method**.

The Metropolis algorithm aims to obtain approximate solutions to numerical problems with unmanageably many degrees of freedom and to combinatorial problems of factorial size, by mimicking a random process. Given the digital computer's reputation for deterministic calculation, it's fitting that one of its earliest applications was the generation of random numbers.

Markov Chains

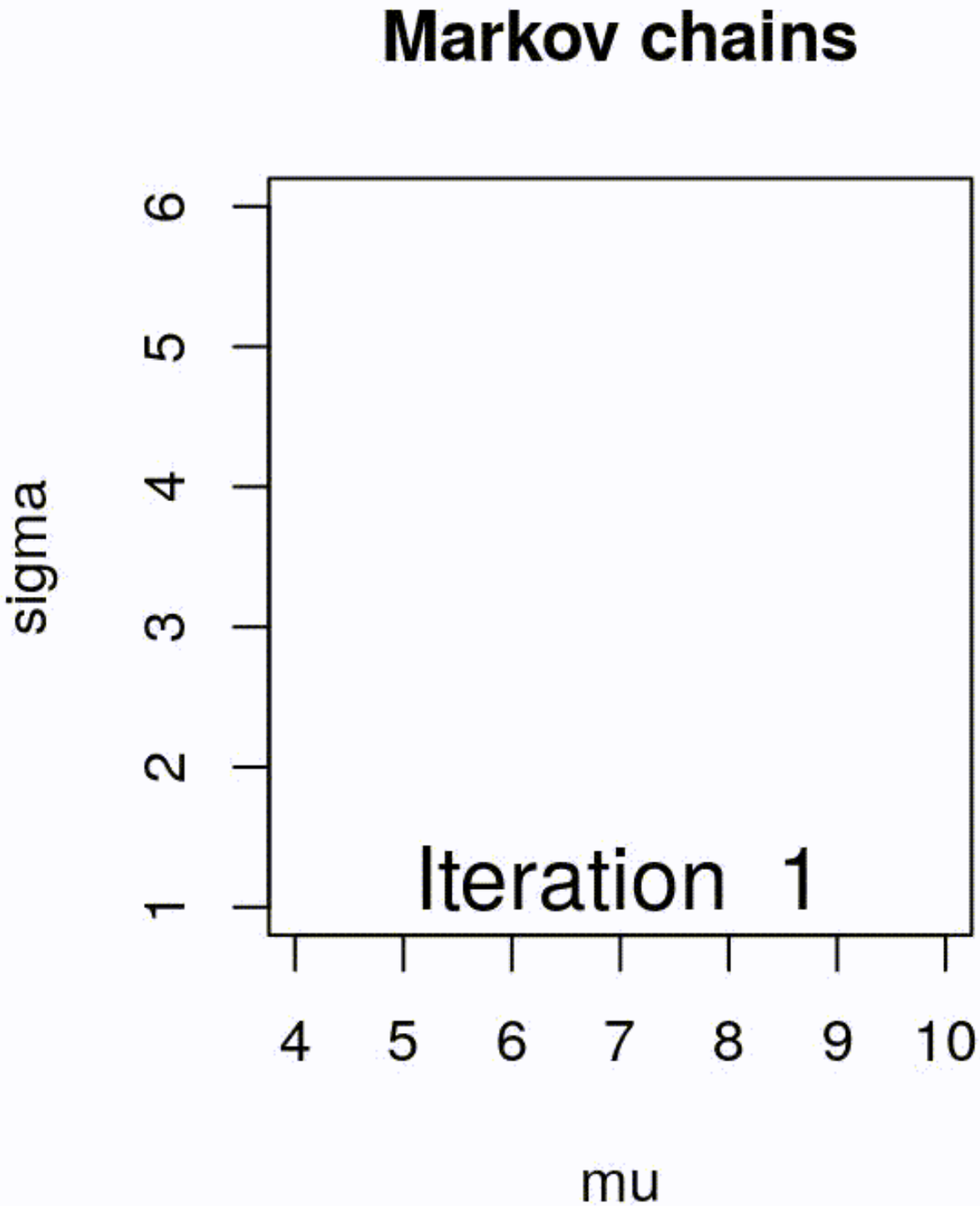


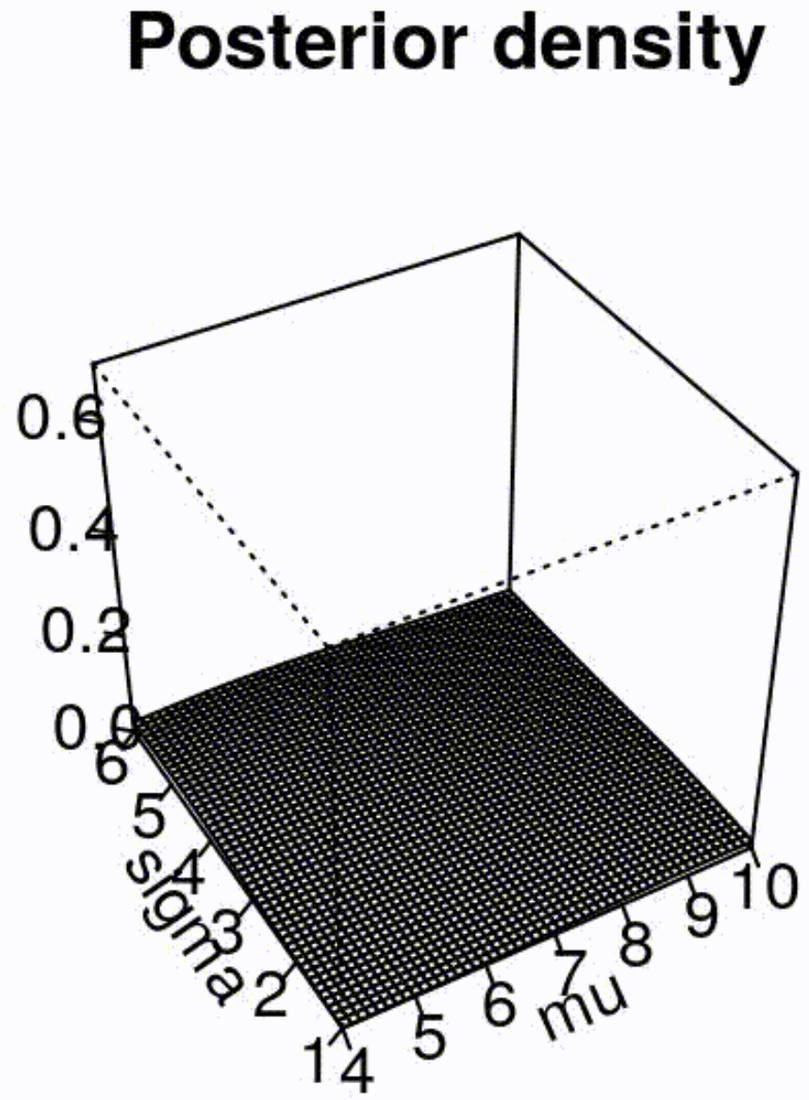




$$\forall x \sum_{i=0}^{N} P_{x,i} = 1$$

The Metropolis algorithm





The Metropolis algorithm for Bayesian inference

Goal: We want to sample from

$$p(\theta \mid y) = \frac{f(y \mid \theta)\pi(\theta)}{m(y)}.$$

Typically, we don't know $m(y)$.

The notation is a bit more complicated, but the set up is the same.

We'll approach it a bit differently, but the idea is exactly the same.

We know $\pi(\theta)$ and $f(y \mid \theta)$, so we can can draw samples from these.

Our notation here will be that we assume parameter values
$\theta_1, \theta_2, \ldots, \theta_s$ which are drawn from $\pi(\theta)$.

We assume a new parameter value comes in that is θ^*.

The Metropolis algorithm for Bayesian inference

The Metropolis algorithm proceeds as follows:

1. Sample $\theta^* \sim J(\theta \mid \theta^{(s)})$.
2. Compute the acceptance ratio (r):

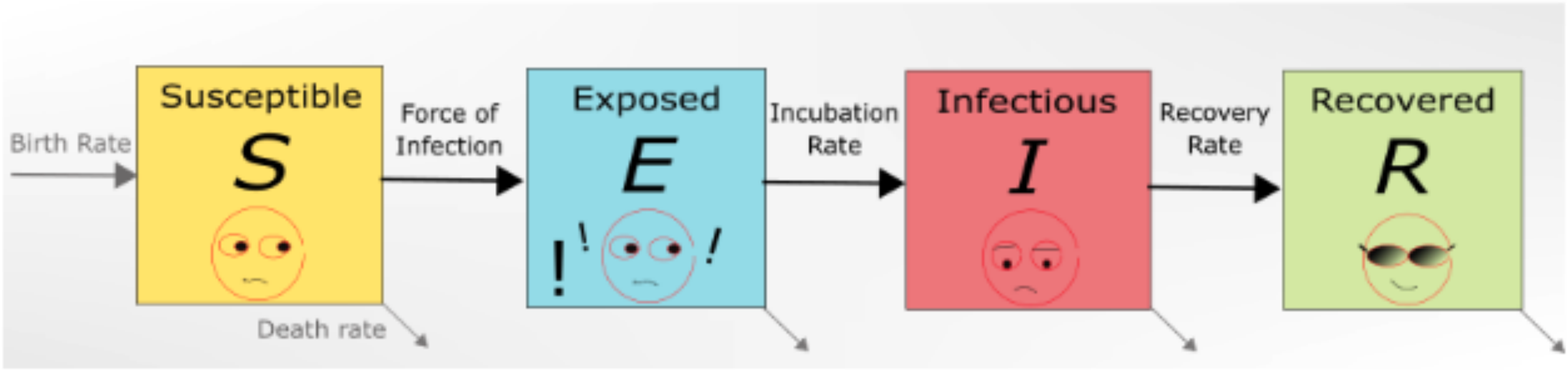
$$r = \frac{p(\theta^*|y)}{p(\theta^{(s)}|y)} = \frac{p(y \mid \theta^*)p(\theta^*)}{p(y \mid \theta^{(s)})p(\theta^{(s)})}.$$

3. Let

$$\theta^{(s+1)} = \begin{cases} \theta^* & \text{with prob min(r,1)} \\ \theta^{(s)} & \text{otherwise.} \end{cases}$$

Remark: Step 3 can be accomplished by sampling $u \sim \text{Uniform}(0, 1)$ and setting $\theta^{(s+1)} = \theta^*$ if $u < r$ and setting $\theta^{(s+1)} = \theta^{(s)}$ otherwise.

Example: Epidemiology models



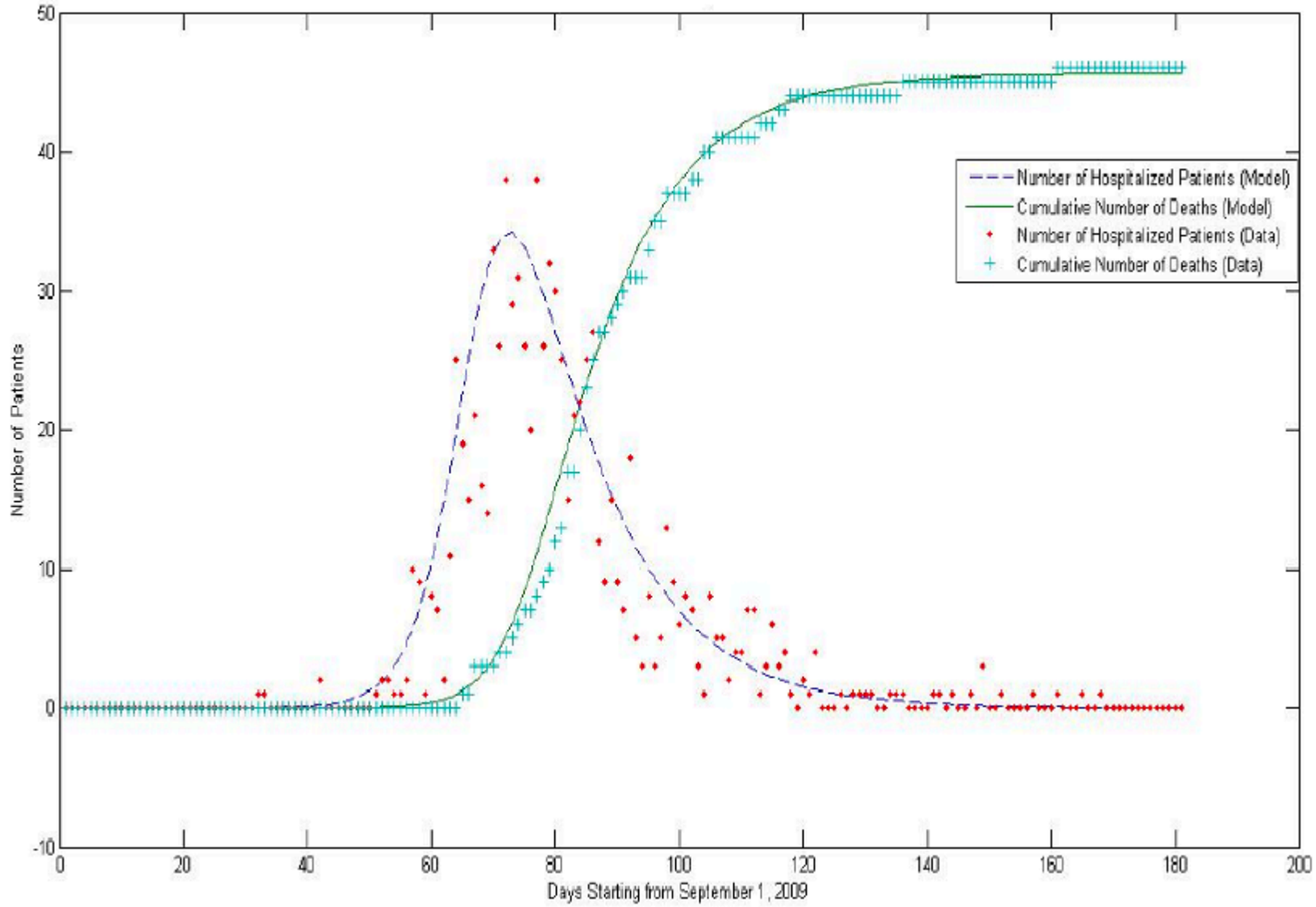
$$\frac{dS}{dt} = -\frac{\beta S I}{N}$$

$$\frac{dE}{dt} = \frac{\beta S I}{N} - \sigma E$$

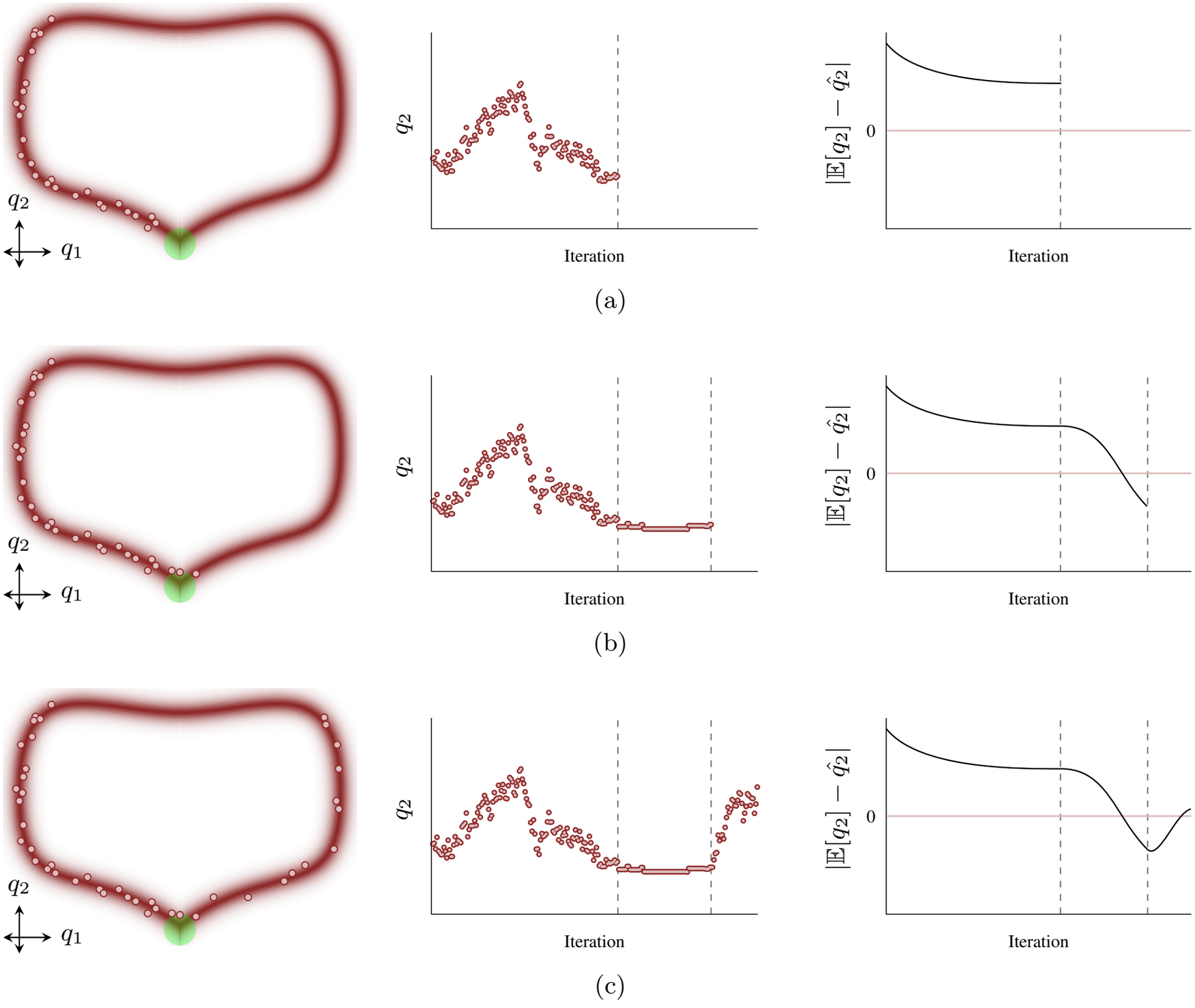
$$\frac{dI}{dt} = \sigma E - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

$$N = S + E + I + R$$



Hamiltonian Monte Carlo



In practice, pathological regions of the typical set usually cause Markov chains to get "stuck".

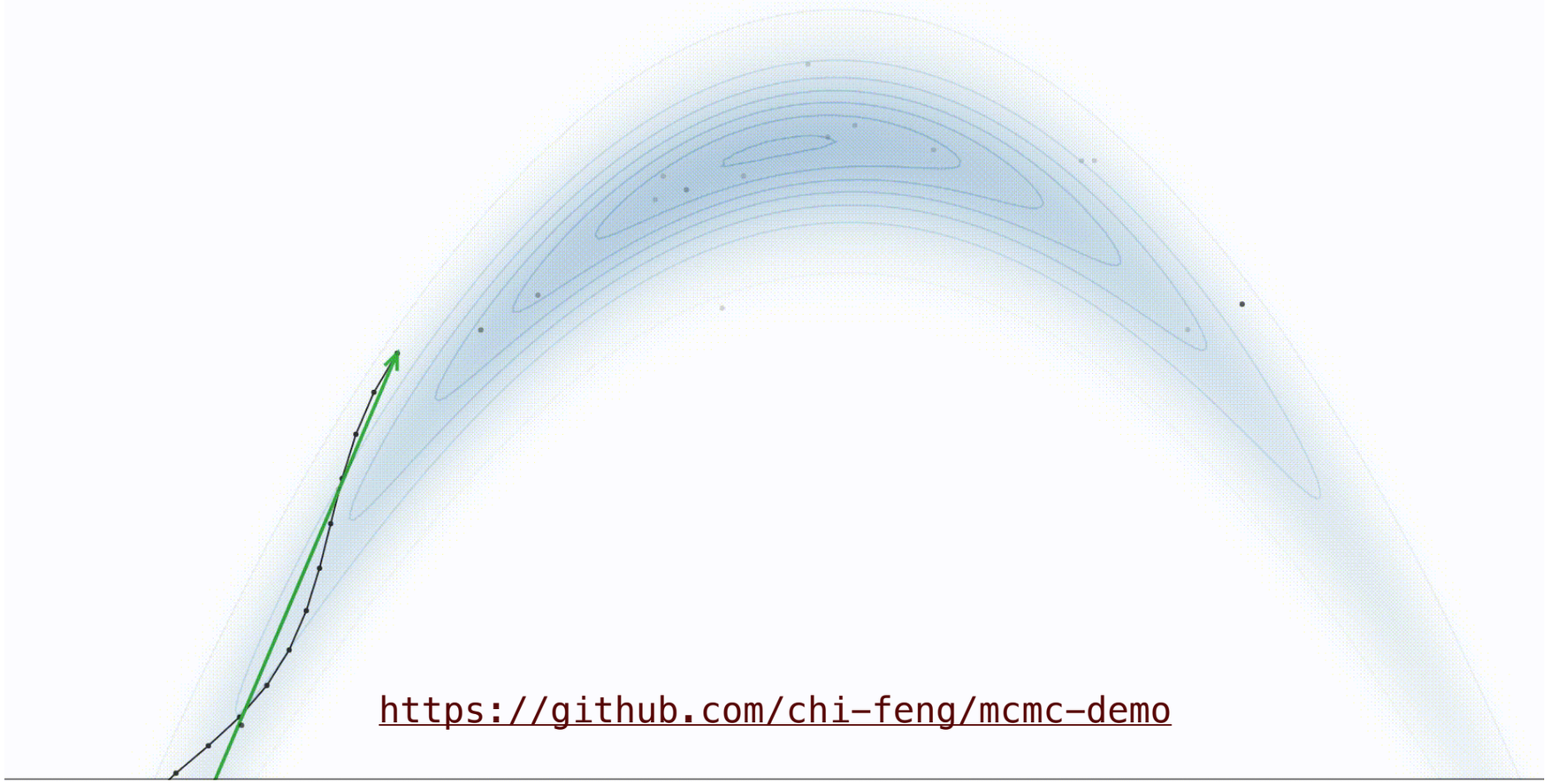
(a) Initially the Markov chain explores well-behaved regions of the typical set, avoiding the pathological neighborhood entirely and biasing Markov chain Monte Carlo estimators.

(b) If the Markov chain is run long enough then it will get stuck near the boundary of the pathological region, slowly correcting the Markov chain Monte Carlo estimators.

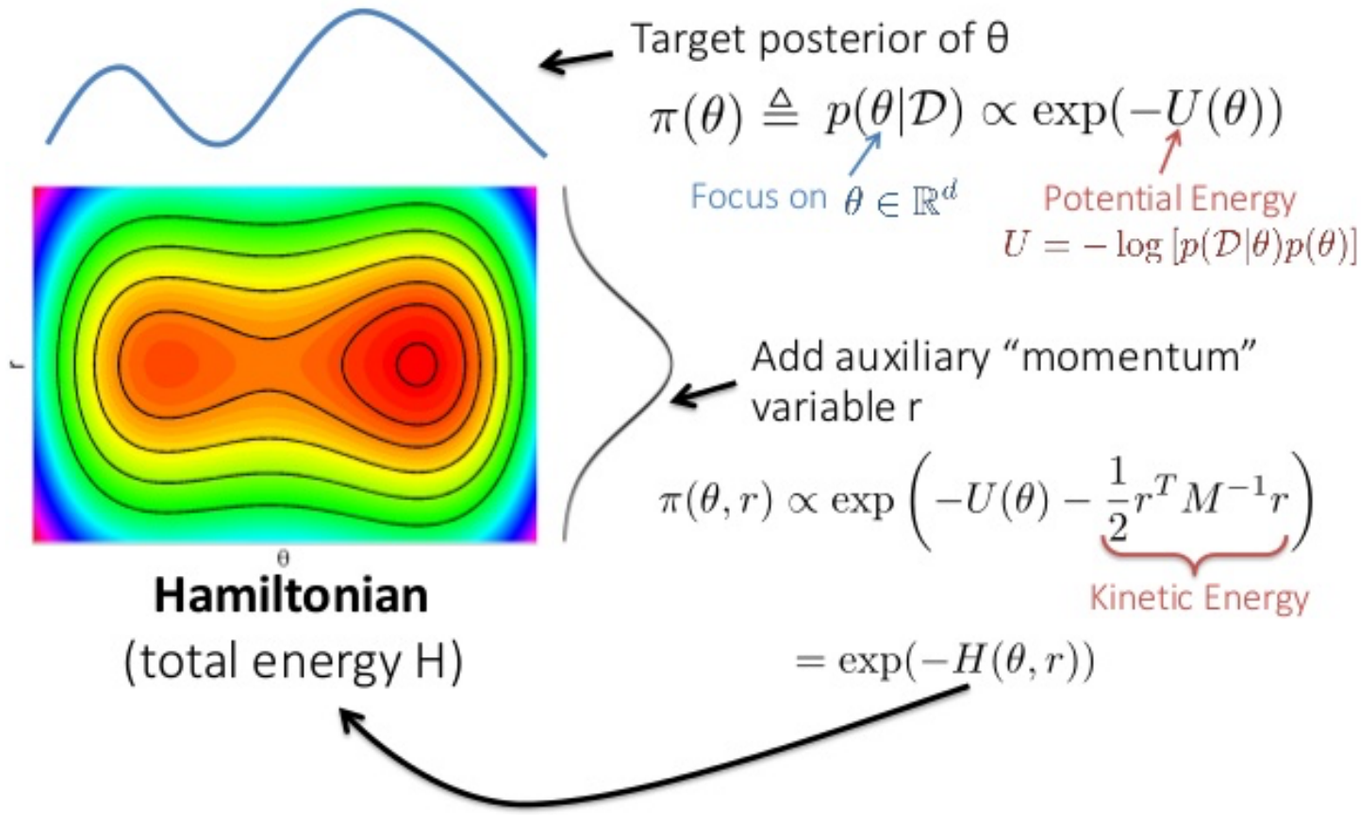
(c) Eventually the Markov chain escapes to explore the rest of the typical set. This process repeats, causing the resulting estimators to oscillate around the true expectations and inducing strong biases unless the chain is improbably stopped at exactly the right time.

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. arXiv preprint arXiv:1701.02434.

Hamiltonian Monte Carlo



https://github.com/chi-feng/mcmc-demo



Gibbs sampling

Gibbs sampling works as follows: suppose we have two parameters θ_1 and θ_2 and some data x.

Our goal is to find the posterior distribution of $p(\theta_1, \theta_2 \| x)$.

Gibbs sampling algorithm:

1. Pick some initial $\theta_2^{(i)}$.
2. Sample $\theta_1^{(i+1)} \sim p(\theta_1 \| \theta_2^{(i)}, x)$
3. Sample $\theta_2^{(i+1)} \sim p(\theta_2 \| \theta_1^{(i+1)}, x)$

Then increment i and repeat K times to draw K samples.

This is equivalent to sampling new values for a given variable *while holding all others constant*.

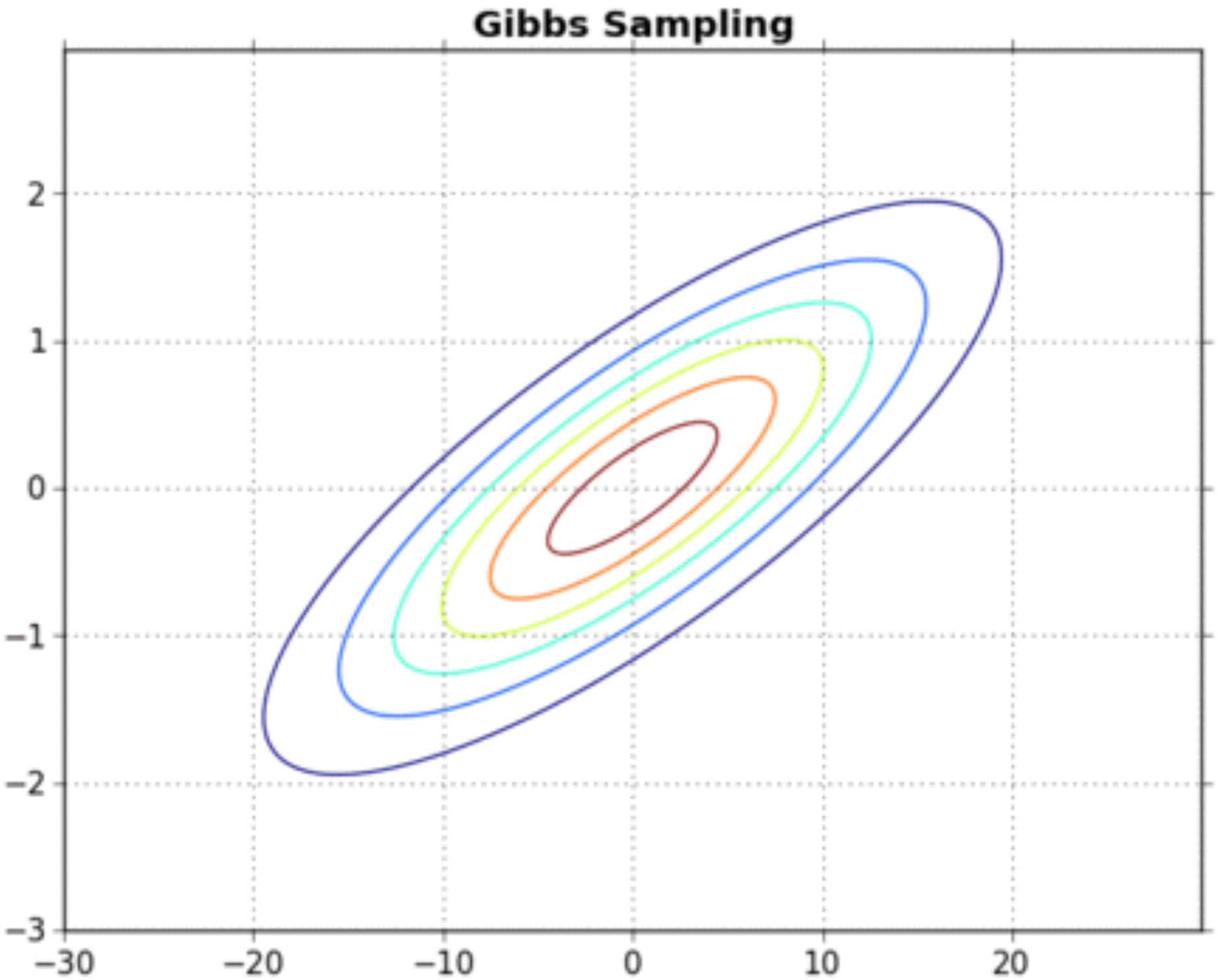
The general approach to deriving an update for a variable is

1. Write down the posterior conditional density in log-form
2. Throw away all terms that don't depend on the current sampling variable
3. Pretend this is the density for your variable of interest and all other variables are fixed. What distribution does the log-density remind you of?
4. That's your conditional sampling density!

Pros: No parameters need to be tuned (e.g. vs MCMC that needs a proposal distribution)

Cons: It might be hard to analytically derive the conditional distributions.

Gibbs sampling



Bayesian linear regression

$$y_i \sim \mathcal{N}(\beta_0 + \beta_1 x_i, 1/\tau)$$

or equivalently

$$y_i = \beta_0 + \beta_1 x_i + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1/\tau)$$

The likelihood for this model may be written as the product over N iid observations

$$L(y_1, \ldots, y_N, x_1, \ldots, x_N | \beta_0, \beta_1, \tau) = \prod_{i=1}^{N} \mathcal{N}(\beta_0 + \beta_1 x_i, 1/\tau)$$

Priors on the model parameters:

$$\beta_0 \sim \mathcal{N}(\mu_0, 1/\tau_0)$$

$$\beta_1 \sim \mathcal{N}(\mu_1, 1/\tau_1)$$

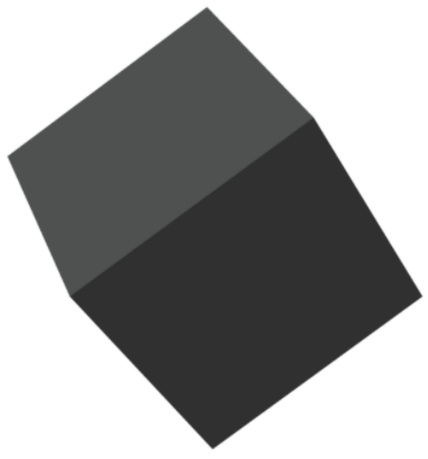
$$\tau \sim \text{Gamma}(\alpha, \beta)$$

Probabilistic programming

http://mc-stan.org/

https://github.com/pymc-devs/pymc3

Edward



http://edwardlib.org/

https://github.com/uber/pyro