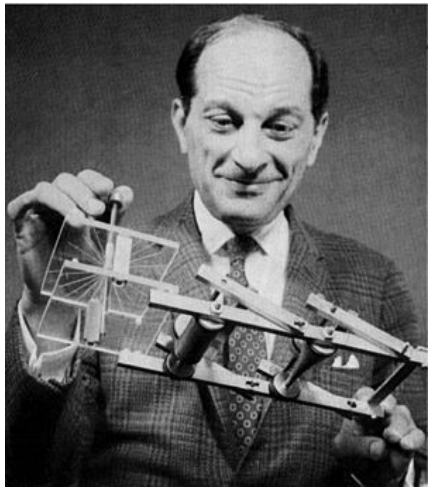
Using Stan for Spatial and Spatio-Temporal Modeling

Matthew Simpson

Department of Statistics, University of Missouri

February 20, 2017

Stan is...



Stanislaw Ulam, inventor of Monte Carlo methods.

`http://mc-stan.org/`

A **probabilistic programming language** that implements **Hamiltonian Monte Carlo (HMC)**, variational Bayes, and penalized maximum likelihood estimation.

Available on Linux, Mac, and Windows with interfaces in **R**, **Python**, shell (command line), MATLAB, Julia, Stata, and Mathematica.

Markov chain Monte Carlo (MCMC)

Goal: sample from some density $\pi(\boldsymbol{q})$.

Create a Markov chain with transition density $k(\boldsymbol{q}'|\boldsymbol{q})$.

- Start with arbitrary $\boldsymbol{q}^{(0)}$ and repeatedly sample $\boldsymbol{q}^{(t+1)} \sim k(\boldsymbol{q}'|\boldsymbol{q}^{(t)})$.
- Under some conditions $\boldsymbol{q}^{(t)} \to \boldsymbol{q}$ in distribution.
- Additionally with geometric ergodicity:

$$\frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{q}^{(t)}) \to \mathrm{N}\left(\mathrm{E}[f(\boldsymbol{q})], \frac{\mathrm{var}[f(\boldsymbol{q})]}{ESS}\right).$$

Auxillary variable MCMC: construct a variable $\boldsymbol{p}$ with joint density $\pi(\boldsymbol{q}, \boldsymbol{p}) = \pi(\boldsymbol{p}|\boldsymbol{q})\pi(\boldsymbol{q})$.

- Construct a Markov chain for $(\boldsymbol{q}, \boldsymbol{p})$ and throw away the sampled $\boldsymbol{p}^{(t)}$s.
- Ex: data augmentation, slice sampling, HMC, etc.

HMC in Theory

Construct $\boldsymbol{p}$ in a special way. Let $\boldsymbol{q}, \boldsymbol{p} \in \Re^n$ and:

$$V(\boldsymbol{q}) = -\log \pi(\boldsymbol{q}) \text{ — } \textit{potential energy}.$$
$$T(\boldsymbol{q}, \boldsymbol{p}) = -\log \pi(\boldsymbol{p}|\boldsymbol{q}) \text{ — } \textit{kinetic energy}.$$
$$H(\boldsymbol{q}, \boldsymbol{p}) = V(\boldsymbol{q}) + T(\boldsymbol{q}, \boldsymbol{p}) \text{ — } \textit{Hamiltonian}, \text{ total energy.}$$

where $\boldsymbol{p}$ denotes *position* and $\boldsymbol{q}$ denotes *momentum*.

Energy-preserving evolution in time is defined by Hamilton's equations:

$$\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial H}{\partial \boldsymbol{q}}; \qquad \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = +\frac{\partial H}{\partial \boldsymbol{p}}.$$

How to implement HMC (in theory):

1. Sample $\boldsymbol{p}' \sim \pi(\boldsymbol{p}|\boldsymbol{q}^{(t)})$.
2. Run Hamiltonian evolution forward in time from $(\boldsymbol{q}^{(t)}, \boldsymbol{p}')$ for a some amount of *integration time* to obtain $(\boldsymbol{q}^{(t+1)}, \boldsymbol{p}^{(t+1)})$.

HMC in Pictures

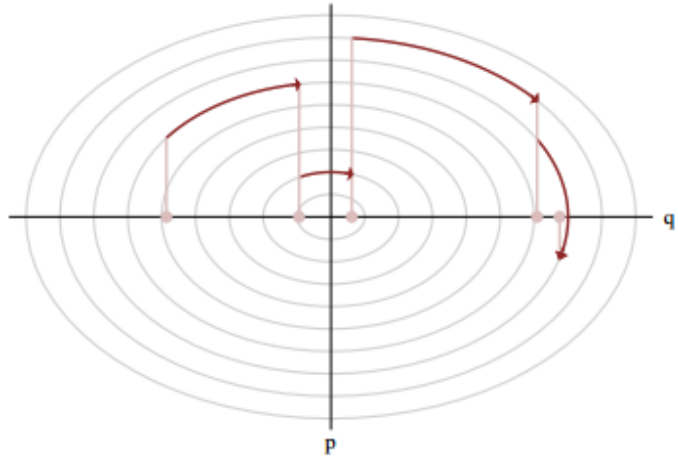


FIG 3. *Every Hamiltonian Markov chain alternatives between a deterministic Hamiltonian flow that explores a single level set (dark red) and a momentum resampling that transitions between level sets with a random walk (light red). The longer the flow is integrated the more efficiently the Markov chain can explore each level set and the smaller the autocorrelations will be. When the flow is integrated for only an infinitesimally small time the Markov chain devolves into a Langevin diffusion.*

HMC samples a level set, then deterministically moves along that set.

Long integration time $\implies$ essentially zero autocorrelation in the chain.

(Picture stolen from `https://arxiv.org/pdf/1601.00225.pdf`)

HMC in Practice

To implement HMC for a differentiable target $\pi(\boldsymbol{q})$ you need:

1. No discrete valued parameters in $\boldsymbol{q}$.
   - Usually can integrate them out, e.g. mixture models.
2. No constrained parameters in $\boldsymbol{q}$.
   - Stan: transform and compute the log-Jacobian automatically.
3. The gradient vector of $\log \pi(\boldsymbol{q})$.
   - Stan: use C++ autodiff library to do this automatically and accurately.
4. Choose a kinetic energy, i.e. $\pi(\boldsymbol{p}|\boldsymbol{q})$.
   - Stan: $\mathrm{N}(\mathbf{0}, \boldsymbol{M})$ and tune $\boldsymbol{M}$ during warmup. (Typical HMC)
   - More intelligent: $\boldsymbol{M}(\boldsymbol{q})$. (Riemannian HMC; future Stan)

HMC in Practice (continued)

To implement HMC for a differentiable target $\pi(\boldsymbol{q})$ you need:

5. Numerical integrator for Hamiltonian's equations.
   - Need to make an adjustment to the Hamiltonian flow and use a Metropolis correction to ensure detailed balance.
   - Typically use leapfrog integration $\Longrightarrow$ how many leapfrog steps ?
   - Stan: adapt number of steps to hit a target Metropolis acceptance rate.

6. An integration time. How long is long enough?
   - Old Stan: No U-Turn Criterion / No U-Turn Sampler (NUTS)

     "stop when we start heading back toward where we started."
   - New Stan: eXhaustive HMC (XHMC/XMC/better NUTS)

     "stop when it looks like autocorrelation should be low."

Why Hamiltonian Monte Carlo?

The long answer:

- `https://www.youtube.com/watch?v=DJ0c7Bm5Djk&feature=youtu.be&t=4h40m10s`
- `https://arxiv.org/pdf/1701.02434.pdf`
- `https://arxiv.org/pdf/1312.0906.pdf`

The short answer:

- Works in high dimensions.
- More robust.
- Makes noise when it fails.

Works in high dimensions

We are interested in expecations of the form $\int f(\boldsymbol{q})\pi(\boldsymbol{q})d\boldsymbol{q}$.

- Naively: focus on areas where $f(\boldsymbol{q})\pi(\boldsymbol{q})$ is large.
- Better: focus on areas where where $f(\boldsymbol{q})\pi(\boldsymbol{q})d\boldsymbol{q}$ is large.

FIG 4. *In high-dimensional parameter spaces probability mass,* $\pi(q)\,\mathrm{d}q$*, and hence the dominant contributions to expectations, concentrates in a neighborhood called the* typical set*. In order to accurately estimate expectations we have to be able to identify where the typical set lies in parameter space so that we can focus our computational resources where they are most effective.*

In high dimensions, this is essentially a surface.

- Random walk type methods (including Gibbs) often fail.
- Methods based on modes often fail.

Robustness and Noisy Failure

HMC has guaranteed geometric ergodicity in a larger class of target densities than alternatives.

When geometric ergodicity fails, HMC often won't sample due to numerically infinite gradients.

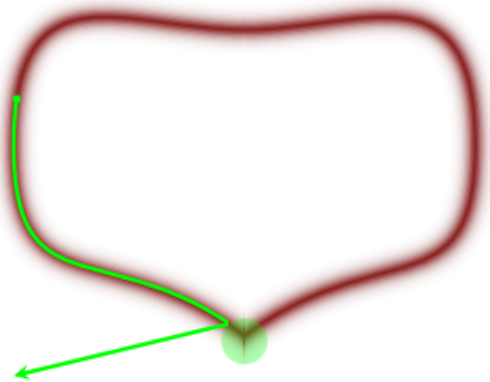


FIG 35. *Neighborhoods of high curvature in the typical set (green) that frustrate geometric ergodicity are also pathological to symplectic integrators, causing them to diverge. This confluence of pathologies is advantageous in practice because we can use the easily-observed divergences to identify the more subtle statistical pathologies.*

- Caused by weird posterior geometries (reparameterize).
- Common in hierarchical models.
- Also a problem in Gibbs samplers, but they still give output.

Using Stan

Define the model in a `.stan` file:

```
data {
  int nobs;
  vector[nobs] y;
}
parameters {
  real mu;
  real<lower = 0> sigma;
}
model {
  y ~ normal(mu, sigma);
  mu ~ normal(0, 10.0);
  sigma ~ normal(0.0, 10.0);
}
```