Hamiltonian Monte Carlo
within Stan

Daniel Lee
Columbia University, Statistics Department
bearlee@alum.mit.edu

Why MCMC?

- Have data.
- Have a rich statistical model.
- No analytic solution.
- (Point estimate not adequate.)

Review: MCMC

- Markov Chain Monte Carlo. The samples form a Markov Chain.
- Markov property:
  $\Pr(\theta_{n+1} \mid \theta_1, \ldots, \theta_n) = \Pr(\theta_{n+1} \mid \theta_n)$
- Invariant distribution:
  $\pi \times \Pr = \pi$
- Detailed balance: sufficient condition:
  $\Pr(\theta_{n+1}, A) = \int_A q(\theta_{n+1}, \theta_n) dy$
  $\pi(\theta_{n+1}) q(\theta_{n+1}, \theta_n) = \pi(\theta_n) q(\theta_n, \theta_{n+1})$

Review: RWMH

- Want: samples from posterior distribution:
  $\Pr(\theta|x)$
- Need: some function proportional to joint model.
  $f(x, \theta) \propto \Pr(x, \theta)$
- Algorithm:
  Given $f(x, \theta)$, x, N, $\Pr(\theta_{n+1} \mid \theta_n)$
  For $n = 1$ to N do
  Sample $\hat{\theta} \sim q(\hat{\theta} \mid \theta_{n-1})$
  With probability $\alpha = \min\left(1, \frac{f(x,\hat{\theta})}{f(x,\theta_{n-1})}\right)$, set $\theta_n \leftarrow \hat{\theta}$, else $\theta_n \leftarrow \theta_{n-1}$

Review: Hamiltonian Dynamics

- (Implicit: d = dimension)
- q = position (d-vector)
- p = momentum (d-vector)
- $U(q)$ = potential energy
- $K(p)$ = kinectic energy
- Hamiltonian system: $H(q, p) = U(q) + K(p)$

Review: Hamiltonian Dynamics

- for $i = 1, \ldots, d$
  $\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$
  $\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$
- kinectic energy usually defined as $K(p) = p^T M^{-1} p/2$
- for $i = 1, \ldots, d$
  $\frac{dq_i}{dt} = [M^{-1}p]_i$
  $\frac{dp_i}{dt} = -\frac{\partial U}{\partial q_i}$

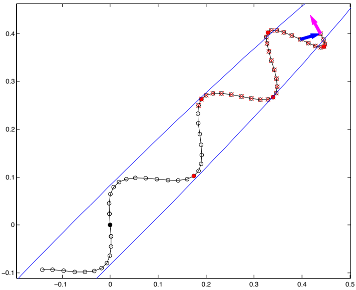
Connection to MCMC

- q, position, is the vector of parameters
- $U(q)$, potential energy, is (proportional to) the minus the log probability density of the parameters
- p, momentum, are augmented variables
- $K(p)$, kinectic energy, is calculated
- Hamiltonian dynamics used to update q.
- Goal: create a Markov Chain such that $q_1, \ldots, q_n$ is drawn from the correct distribution

Hamiltonian Monte Carlo

- Algorithm:
  Given $U(q) \propto -\log(\Pr(q, x))$, q_0, N, time (ϵ, L)
  For $n = 1$ to N do
    Sample $p \sim N(0, 1)$
    $q_{start} \leftarrow q_{n-1}$, $p_{start} \leftarrow p$
    Get q and p at time using Hamiltonian dynamics
    $p \leftarrow -p$
    With probability $\alpha = \min\left(1, \exp(H(q, p) - H(q_{start}, p_{start})\right)$,
      set $q_n \leftarrow q$, else $q_n \leftarrow q_{n-1}$.

HMC Example Trajectory



- Blue ellipse is contour of target distribution
- Initial position at black solid circle
- Arrows indicate a U-turn in momentum

HMC Review

- Correct MCMC algorithm; satisfies detailed balance
- Use Hamiltonian dynamics to propose new point
- Metropolis adjustment accounts for simulation error
- Explores space more effectively than RMWH
- Difficulties in implementation

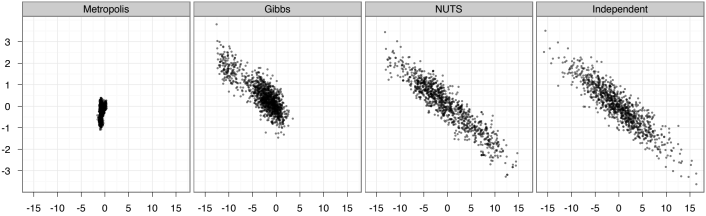
Nuances of HMC

- Simulating over discrete time steps: Error requires accept / reject step.
- leapfrog integrator. ϵ, L.
- Need to tune amount of time.
- Negatve momentum at end of trajectory for symmetric proposal.
- Need derivatives of $U(q)$ with respect to each q_i.
- Samples efficiently over unconstrained spaces. Needs continuity of $U(q)$.

Stan's solutions

- Autodiff: derivatives of $U(q) \propto -log(\Pr(q, x)$.
- Transforms: taking constrained variables to unconstrained.
- Tuning parameters: No-U-Turn Sampler.

Different MCMC algorithms



What is Stan?

1. Language for specifying statistical models

   $\Pr(\theta, X)$

2. Fast implementation of statistical algorithms; interfaces from command line, R, Python, Matlab

What is Stan trying to solve?

- Stan: model fitting on arbitrary user model
- Stan: speed, speed, speed
- Team: easily implement statistics research
- Team: roll out stats algorithms
- User: easy specification of model
- User: trivial to change model
- User: latest and greatest algorithms available

Language: applied Bayesian modeling

1. Design joint probability model for all quantities of interest including:
    - observable quantities (measurements)
    - unobservable quantities (model parameters or potentially observable quantities)
2. Calculate the posterior distribution of all unobserved quantities conditional on the observed quantities
3. Evaluate model fit

Language: features

- high level language; looks like stats;
  inspired by BUGS language
- Imperative declaration of log joint probability distribution
  $\log(\Pr(\theta, X))$
- Statically typed language
- Constrained data types
- Easy to change models!
- Vectorization, lots of functions, many distributions

Coin flip example

```
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  theta ~ beta(1,1);
  y ~ bernoulli(theta);
}
```

Language

- Discuss more later

- Data types
- Blocks
- Constraints and transforms

Implementation

- Stan v2.2.0. 2/14/2014.
- Stan v2.3.0. Will be out within a week.
- Stan written in templated C++. Model translated to C++.
- Algorithms:
  - MCMC: auto-tuned Hamiltonian Monte Carlo, no-U-turn Sampler (NUTS)
  - Optimization: BFGS

Stan Stats

- Team: ~12 members, distributed
- 4 Interfaces: CmdStan, RStan, PyStan, MStan
- 700+ on stan-users mailing list
- Actual number of users unknown
  - User manual: 6658 downloads since 2/14
  - PyStan: 1299 downloads in the last month
  - CmdStan / RStan / MStan: ?
- 75+ citations over 2 years
  - stats, astrophysics, political science
  - ecological forecasting: phycology, fishery
  - genetics, medical informatics