More Monte Carlo Methods

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

March 3, 2021

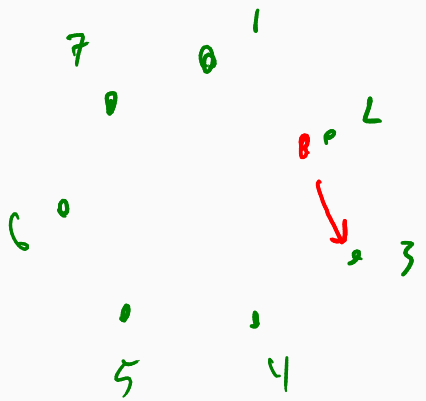
Recap: Metropolis (-Hastings)

Discrete example

$f(x_i)$ – failure rate for machine i.

A simple algorithm:

1. Order the machines $x_1, \ldots, x_7$.
2. Start at a random machine x_i.
3. Flip a coin to decide whether to propose x_{i-1} or x_{i+1}.
4. Say we chose x_{i+1}: with probability $f(x_{i+1})/f(x_i)$, move to x_{i+1}; otherwise, stay at x_i



Metropolis algorithm in general

More generally: Metropolis algorithm.

1. Start with a (possibly un-normalized) density function f on the sample space – this is our target distribution.
2. Establish a proposal distribution $q(x, y)$ that is a *symmetric* transition probability on the state space, meaning $q(x, y) = q(y, x)$.
3. At time t, when $X_t = x$, sample a value y from $q(x, y)$ – propose a transition $x \to y$
4. Compute the acceptance probability

$$\alpha(x, y) = \min\left(\frac{f(y)}{f(x)}, 1\right)$$

and jump from x to y with that probability, otherwise stay at x.

Why does this work?

Detailed balance equation for a stationary distribution:

$$\pi(x)p(x,y) = \pi(y)p(y,x)$$

For any Markov chain with transition function p, if π satisfies this equation, π is stationary. (Intuition: the flow of probability mass from x to y equals that from y to x.)

Why does this work?

Detailed balance equation for a stationary distribution:

$$\pi(x)p(x,y) = \pi(y)p(y,x)$$

For any Markov chain with transition function p, if π satisfies this equation, π is stationary. (Intuition: the flow of probability mass from x to y equals that from y to x.)

Transition probability for the Metropolis algorithm:

$$p(x,y) = q(x,y)\frac{f(y)}{f(x)}$$

Plug into the detailed balance equation, substituting $f(x)$ for $\pi(x)$:

$$f(x)q(x,y)\min\left(\frac{f(y)}{f(x)},1\right) = f(y)q(y,x)\min\left(\frac{f(x)}{f(y)},1\right)$$

Assume $f(x) > f(y)$: left hand side is $q(x,y)f(y)$, right hand side is $q(y,x)f(y)$. If q is symmetric, then we're good!

In code

Simple option: Gaussian proposal. Proposed y is $x + N(0, \sigma)$. Depends only on distance between x and y, so definitely symmetric.

In code

Simple option: Gaussian proposal. Proposed y is $x + N(0, \sigma)$. Depends only on distance between x and y, so definitely symmetric.

target density a function

```
def metro_step(x, p, sigma):
    proposal = x + sp.stats.norm(0,np.eye * sigma).rvs()
    alpha = min(1, p(proposal) / p(x))
    if np.random.rand() <= alpha:
        new_x = proposal
    else:
        new_x = x
```

Metropolis algorithm in general

More generally: Metropolis-Hastings algorithm.

1. Establish a proposal distribution $q(x, y)$ that is a transition probability on the state space
2. At time t, when $X_t = x$, sample a value from $q(x, y)$
3. Compute the acceptance probability

$$\alpha(x, y) = \min\left\{\frac{p(y)q(y, x)}{p(x)q(x, y)}, 1\right\}$$

eliminates symmetry requirement.

and jump from x to y with that probability, otherwise stay at x.

Notice one difference: factor of q in the acceptance probability. This is Hastings's contribution, and allows for $q(x, y) \neq q(y, x)$. Exercise: demonstrate that this works.

General difficulties

Two general issues.

- Burn-in: the chain takes time to approach the target distribution
- Autocorrelation: we are not sampling independent draws from the target distribution

General difficulties

Two general issues.

- Burn-in: the chain takes time to approach the target distribution
- Autocorrelation: we are not sampling independent draws from the target distribution
- "Solutions:"
  - Always use multiple chains so we can compare them to help assess convergence to the target distribution
  - Use proposals that efficiently explore the sample space

Difficulty

The practical difficulty of Metropolis-Hastings comes in trying to choose a good proposal distribution.

Generally we have a tradeoff:

- Highly variable proposal distribution: better exploration of the sample space (less time to reach the target distribution) but many rejected proposals
- Tight proposal distribution: more accepted proposals, but "random walk" behavior

This can be especially bad depending on the form of the posterior. Let's see some examples in action.

`https://chi-feng.github.io/mcmc-demo`

A better way to get proposals

Modern update: Hamiltonian Monte Carlo (aka Hybrid Monte Carlo).

- Hybrid because it mixes a deterministic simulation with the random nature of MCMC
- Hamiltonian because the simulation uses Hamilton's equations from physics
- Essentially a way to use the properties of the target distribution to generate proposals

There are some older "adaptive" methods, but they are mostly limited to, e.g., Gaussian proposals with correlations

HMC: Borrowing an idea from physics

Hamiltonian mechanics:

- Define space coordinates and momentum coordinates q_i, p_i to describe the state of a particle
- Summarize the momentum and potential energy by a function $H(q, p)$ — Hamiltonian
- System evolves according to the equations:

$$\frac{\partial q_i}{\partial t} = \frac{\partial H}{\partial p_i}$$
$$\frac{\partial p_i}{\partial t} = -\frac{\partial H}{\partial q_i}$$

Intuitively: think of the potential energy as a frictionless surface, and the particle slides around it by gravity

HMC details

How HMC works:

1. Pick a starting point in parameter space, and place a particle there with fixed mass.
2. Randomly select a momentum vector, and give the particle that momentum
3. Treating the (minus log) probability density of the target distribution as a potential energy, simulate the motion of the particle for a fixed time $\Delta t \times L$ — # of time step
4. Use the endpoint as a proposal for a Metropolis step

target_accept (by default this is 0.8)

HMC summary

- HMC explores parameter space efficiently because its proposals follow the geometry of the (log) posterior 
- Extra variables, particularly energy, provide diagnostics
    - Hamiltonian mechanics conserves energy, so if energy changes something is wrong

HMC summary

upsides

- HMC explores parameter space efficiently because its proposals follow the geometry of the (log) posterior
- Extra variables, particularly energy, provide diagnostics
  - Hamiltonian mechanics conserves energy, so if energy changes something is wrong

downside

- Requires computation of more stuff:
  - Need gradients of the log posterior, which are model-specific
  - Need to run the simulation
  - Extra parameters: momentum, mass
- Extra tuning: physics step size, number of steps

Tuning HMC

Two reasons you need to tune these:

1. Step size determines accuracy of the physics simulation. Small step size means more accurate physics, proposals accepted more often. But it means you need more steps to get the same amount of mixing, so more computation per sample
2. The U-turn problem

The NUTS sampler

The leading implementation of HMC currently is called NUTS[1] (No U-Turn Sampler).

The U-turn problem:

- If the simulation runs too long, the particle can cross a local minimum, turn around, and come back
- Leads to inefficient exploration of the sample space

Let's see a really terrible example.

[1] sorry...

The NUTS sampler

The leading implementation of HMC currently is called NUTS[1] (No U-Turn Sampler).

The U-turn problem:

- If the simulation runs too long, the particle can cross a local minimum, turn around, and come back
- Leads to inefficient exploration of the sample space

Let's see a really terrible example.

To avoid U-turns, tune the number of physics steps. NUTS runs some simple ML on the warm-up results to automatically tune L. Δt is still a potential tuning parameter.

[1] sorry...

When things go wrong

Recap

Last week, we saw an example of a hierarchical model (hierarchical normal model for 8 schools problem) where the sampler misbehaved.

We saw:

- warnings about divergences
- trace plots
- inconsistent results from different chains

} different results from different chains.

So:

- what do these warnings mean?
- what went wrong?
- how can we fix it?

What is a divergence?

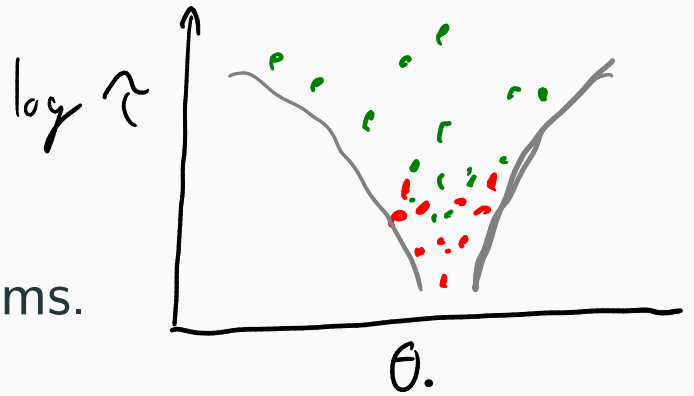
The core sampling step of HMC is a physics simulation:

- The Hamiltonian $H(q, p)$ represents the total energy of the system
- Hamiltonian systems conserve energy

A "divergence" in HMC is when the energy at the start of the simulation doesn't match the energy at the end. It means something went wrong with the physics.

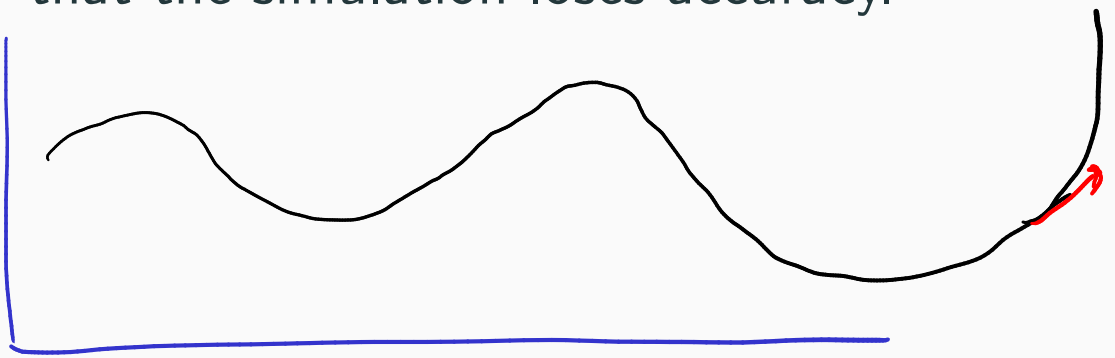
there is a threshold in the settings somewhere.

Divergence: causes and effects



What causes a divergence: numerical problems.

- The physics simulation uses discrete time steps to model a continuous process
- If our time steps are too large, our simulation is too "coarse"
- Time steps need to be smaller when the potential energy (i.e. log posterior) has high curvature (2nd derivative) meaning that the simulation loses accuracy.



How to fix divergences

When we get divergences, the sampler has some suggestions for us:

```
There were 282 divergences after tuning.
Increase `target_accept` or reparameterize.
```

So we have two main options: increase `target_accept` or reparameterize

Simple approach: change target acceptance

The simplest way to deal with divergences: decrease the step size in the physics simulation. This is what `target_accept` controls.

- This decreases the efficiency of the sampling, because it requires more physics steps per sample.
- Default for `target_accept` is 0.8; try setting to, e.g., 0.9, 0.95

This works for random "false positive" divergences, but sometimes won't help. Divergences that won't go away are a serious problem.

The folk theorem

The "folk theorem of statistical computing:"

[Gelman] When you have computation problems, often there's a problem with your model.

- Check simple things first. Things I have done that have led to computational problems:
  - Forgetting to set the shape of a parameter
  - Forgetting to link up two parameters
- Set reasonable weakly regularizing priors
  - Avoid flat priors

Excessive curvature in the 8 schools problem

Where is the problem in the 8 schools model?

Let's first look at some plots.

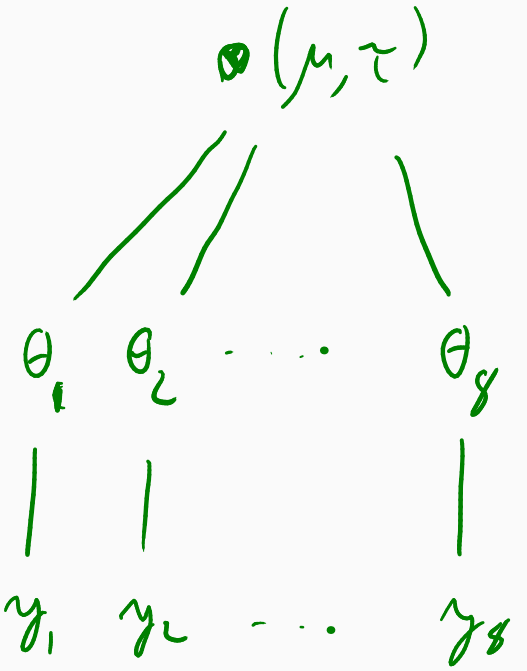
Excessive curvature in the 8 schools problem

Where is the problem in the 8 schools model?

Let's first look at some plots.

Recall the model:

$$
\begin{aligned}
y_j &\sim \text{Normal}(\theta_j, \sigma_j) \\
\theta_j &\sim \text{Normal}(\mu, \tau) \\
\mu &\sim \text{Normal}(0, 5) \\
\tau &\sim \text{HalfCauchy}(5)
\end{aligned}
$$



The source of the problem

Remember the physics "surface" is the log posterior. So let's examine that.

The source of the problem

Remember the physics "surface" is the log posterior. So let's examine that.

$$p(\mu, \tau, \theta|y) \propto p(\mu, \tau, \theta)p(y|\mu, \tau, \theta) \quad \text{(prior, likelihood)}$$
$$= p(\mu, \tau)p(\theta|\mu, \tau)p(y|\theta)$$

Bayes theorem

Taking logs,

$$log p(\mu, \tau, \theta|y) = \text{const} + \log p(\mu, \tau)$$

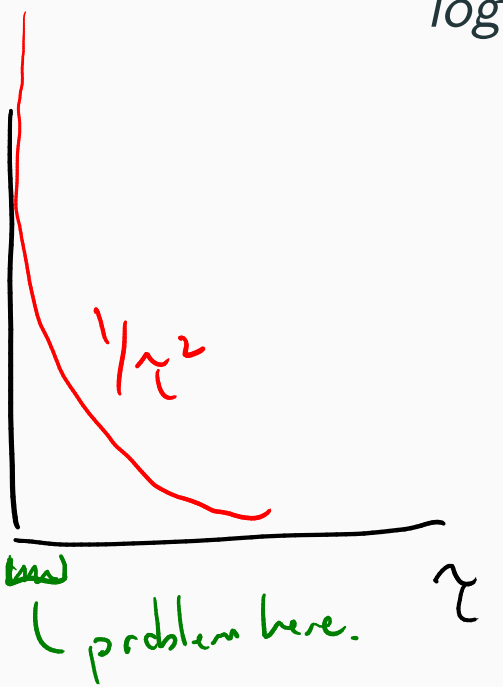
from prior on μ, τ

$$- \frac{1}{2}\sum_{j=1}^{J}\left(\frac{\theta_j - \mu}{\tau}\right)^2$$

from dependence of θ on μ, τ..

$$- \frac{1}{2}\sum_{j=1}^{J}\left(\frac{y_j - \theta_j}{\sigma_j}\right)^2$$

likelihood



Reparameterizing the 8 schools

We can reparameterize to a "non-centered" parameterization.

- If $x \sim N(0, 1)$, then $\sigma x \sim N(0, \sigma)$
- So, the following is equivalent to our original model:

$z = \frac{x - \mu}{\sigma}$

$x = \sigma z + \mu$

$$
\begin{aligned}
y_j &\sim \text{Normal}(\theta_j, \sigma_j) \\
\eta_j &\sim \text{Normal}(0, 1) \\
\theta_j &= \mu + \tau \eta_j \\
\mu &\sim \text{Normal}(0, 5) \\
\tau &\sim \text{HalfCauchy}(5)
\end{aligned}
$$

} extra variable

We add a latent variable and move all the variance of θ_j into it; τ no longer appears in the denominator in the log posterior.

Diagnostic statistics

Some other diagnostic statistics that you can use:

- $\hat{R}$ (a.k.a. the Gelman-Rubin statistic), measures the ratio of the estimated variance of the parameter, pooling several chains, to the variance within a chain. Should be nearly 1 if all chains have converged to the same distribution. You'll get warnings if $\hat{R}$ is too high for any parameter. (See BDA sec. 11.4)
- Effective sample size: estimate of the equivalent number of samples *without* autocorrelation. You'll get warnings if this is really low. (See BDA sec. 11.5)

Both calculated by `pm.summary`; let's take a look.

Summary

Further reading for practical advice on using MCMC methods:

- Divergences in the 8 schools model. `https://colcarroll.github.io/pymc3/notebooks/Diagnosing_biased_Inference_with_Divergences.html`
- General notes on using HMC in PyMC3. Lots of good small tips and tricks here. `https://eigenfoo.xyz/_posts/2018-06-19-bayesian-modelling-cookbook/`
- Notes on choosing priors. Written by Stan users, but most of it is really language-agnostic (because it's more about the modeling side). `https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations`

Next week:

- Information criteria for model evaluation