6/19/19 (W2a-1)

MCMC Sampling III

- Today we will briefly follow-up on the evidence discussion and then talk mostly about Hamiltonian Monte Carlo (HMC) and its implementation in pymc3.

- First: Computational possibilities for evidence ⇒ (T2a-8)
  - Examples of Information Criteria ⇒ briefly given time.
  - For further information BDA3 (Bayesian Data Analysis 3 by Gelman et al.) is a good reference.

AIC: Akaike Information Criteria
- Essentially frequentist → relies on likelihood
- Want to minimize: $-2\log p(D|\hat{\theta}_{MLE}) + 2k$

  likelihood evaluated at Maximum Likelihood value of parameters (peak of likelihood)

  k: # of free parameters
- So it has ingredients: improved likelihood balanced by penalty for extra parameters. No priors.
- Not regarded well by Bayesians

BIC: Bayesian Information Criterio
- Gaussian approximation to Bayesian evidence in limit of large amount of data
- minimize $BIC = -2\log p(D|\hat{\theta}_{MLE}) + k\ln N$

  k: # fitted pts; N: # data pts
- Assumes Occam razor penalty is negligible.

DIC: Deviance Information Criteria
- replace $\hat{\theta}_{MLE}$ by $\hat{\theta}_{Bayes}$ ← maximum of posterior
- use effective # of parameters $p_{DIC} = 2\log p(D|\hat{\theta}_{Bayes}) - E[\log p(D|\theta)]$

  ↑ average θ over posterior
- So k replaced by data-based bias correction: $DIC \equiv -2\log p(D|\hat{\theta}_{Bayes}) + 2p_{DIC}$

widely applicable

WAIC: Favored by BDA-3 as more fully Bayesian

Sampling: $2\sum_{i=1}^{N_{data}}\left(\log \frac{1}{S}\sum_{s=1}^{S} p(y_i|\theta^s)\right) - \frac{1}{S}\sum_{s=1}^{S}\log p(D_i|\theta^s)$

averages over posterior distribution

6/19/19

Visualization of MCMC Sampling Revisited

- We return to the excellent set of interactive demos by Chi Feng at https://chi-feng.github.io/mcmc-demo/ and their adaptation by Richard McElreath at http://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/.

- This blog piece very strongly advocates abandoning Metropolis-Hastings sampling in favor of Hamiltonian Monte Carlo (HMC), our topic for the rest of today.
  - First recall the Random Walk MH
    1) make a random proposal for new parameter values (step in parameter space ⇒ arrow)
    2) Accept or reject proposal based on a Metropolis criterion.
  - Diffusion (random walk) so not efficient in exploring the space and needs special tuning to avoid too many rejections.
  - Donut shape is common is higher dimensions and it is hard to explore.

- "Better living through physics" ⇒ HMC simulation
  - The description is that we map our parameter vector to a particle in an n-dimensional space. The surface is an n-dimensional bowl with shape given by minus-log target distribution (e.g. posterior). See log Gaussian.
  - Treat as frictionless. "Flick" particle in random direction, so it flows across the bowl.
  - See the simulation: little gray arrow is flick. After travels some distance, decide whether to accept. Most are within high probability region, so high percentage accepted.
  - Chains can get far from starting point ⇒ efficient exploration of full shape.
  - More calculation but fewer samples ⇒ trade off wins.
  - Check the donut ⇒ looks very good!

W2a-3

6/19/19

- There is a further improvement called NUTS – no U-turn sampler. The idea is to address the problem that HMC needs to be told how many steps to take before another flick.
  - too few steps ⇒ samples too similar
  - too many steps ⇒ also too similar.

ask class: what is the drawback of paths in simulation (stem is NUTS)?

- tuning by hand is difficult with complex distribution.

- Solution is NUTS.
  - adaptively finds a good number of steps
  - simulates in both directions to figure out when the path turns around (U-turn) and stops it.
  - Other adaptive features – see documentation.

- Note that NUTS still has trouble with multimodal targets ⇒ can explore each high probability area, but has trouble going between.

6/19/19

look at these pictures. → "Hamiltonian Monte Carlo Explained" by Alex Rogozhnikov and "MCMC Using Hamiltonian Dynamics" by Radford Neal.

The basic idea is to translate a pdf for the distribution desired to a potential energy function and then add a momentum variable — fictitious! The Markov chain at each iteration resamples the momentum, creates a proposal using Hamiltonian dynamics, and then does a Metropolis update.

Recall Hamiltonian dynamics, now applied to d-dimensional position vector q and a d-dimensional momentum vector p ⇒ $2d$ phase space Hamiltonian is $H(q,p)$.

- Equations of motion describe time evolution:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i} \quad \text{and} \quad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i} \quad i=1,\dots,d$$

so these map states at t to states at $t+s$.

We take the form of H to be $H(q,p) = U(q) + K(p)$
- potential energy $U(q)$ is minus the log probability density of the distribution for q we seek to sample.
- $K(p)$ is kinetic energy $K(p) = p^T M^{-1} p/2$ and M is symmetric, positive "mass matrix" typically diagonal and even $m\mathbb{1}_d$
  - This is minus the log probability density (plus constant) of zero mean Gaussian with covariance matrix M.

- What are we going to do with this? We consider a canonical distribution

$$P(q,p) = \frac{1}{Z} e^{-H(q,p)/T} = \frac{1}{Z} e^{-U(q)/T} e^{-K(p)/T}$$

so q and p are independent. We are interested in q; p is fake to make things work. Usually $U(q)$ is a posterior: $-\log[p(q|D)\, p(q)]$

θ, for example.

W2a-5

6/19/19

Two steps of HMC algorithm:

1) New values for the momentum variables are randomly drawn from their Gaussian distribution, independent of current position values.
   - This means p_i has mean zero and variance m_i if M diagonal.
   - q isn't changed, p is from correct conditional distribution given q, so canonical joint distribution invariant

2) Proposal from Hamiltonian dynamics for new state. Simulate from (q, p) with L steps of size $\varepsilon \Rightarrow$ parameters.
At end the momenta are flipped sign a new proposed state (q^*, p^*) accepted with probability

$$\min\left[1, e^{-H(q^*, p^*) + H(q,p)}\right] = \min\left[1, e^{-U(q^*) + U(q) - K(p^*) + K(p)}\right]$$

- momentum flip makes proposal symmetrical, but not done in practice
- So probability density for (q, p) (almost) unchanged because energy is conserved, but in terms of $\underline{q}$ we get a very different probability density.

- You can show that HMC leaves canonical distribution invariant because detailed balance holds, which is what we need. It will also be ergodic
$\Rightarrow$ doesn't get stuck in subset of state space but samples all.
   - Some subtlety about periodic orbit but ok.

Essential features:
- Reversibility needed so that desired distribution is invariant.
- Conservation of Hamiltonian (which is the energy)
- Volume preservation - preserves volume in (q, p) space - This is Liouville's Theorem. (If we take a cluster of points and follow them, the volume occupied is unchanged.)

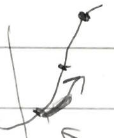
$\Rightarrow$ This is critical because a change in volume would mean we would have to make a nontrivial adjust to the proposal (the normalization Z would change)

6/19/19

W2a-6

- These requirements are satisfied by exact equation, but we are <u>approximating</u> differential equations. This requires a symplectic integration.

  - Ordinary Runge-Kutta-type ODE solver won't work. We need something like leapfrog:



Euler not time reversal invariant

$$p_i(t+\varepsilon/2) = p_i(t) - (\varepsilon/2)\frac{\partial U}{\partial q_i}(q(t)) \quad \leftarrow \text{half step}$$

$$q_i(t+\varepsilon) = q_i(t) + \varepsilon\, p_i(t+\varepsilon/2)/m_i \quad \leftarrow \text{use intermediate } p$$

$$p_i(t+\varepsilon) = p_i(t+\varepsilon/2) - (\varepsilon/2)\frac{\partial U}{\partial q_i}(q(t+\varepsilon)) \quad \leftarrow \text{other half step}$$

using p at t or $t+\varepsilon$ wouldn't be symmetric.

- See pictures in Fig 1 of Neal.
- Benefits of HMC seen in Figures 3–6

- Many tuning features are automatic with pymc3, so lets dive in and take a look.

W2e-7

6/19/19

- Lightning intro to PyMC3

- Plan for this afternoon: your job is to go through some of the notebooks and make changes to see how it works. Later we'll have a notebook for you to step through the lighthouse problem.
- Bayes2019/topics/mcmc-sampling/PyMC3/

- Start with PyMC3_intro.ipynb (A)
  - Then do PyMC3_docs_getting_started.ipynb (B)

(A) Initial problem is sampling to find the posterior for μ, the mean of a distribution, given data generated according to a normal distribution (actual mean is zero).
  - Things to try: changing the true μ, sampling sigma as well.
  - The standard deviation is initially fixed at 1.
- In the code one specifies priors, then the likelihood. By Bayes' theorem this predicts the posterior for μ:

"cores" is # of chains to run in parallel

$$p(\mu|D,\sigma) \propto p(D|\mu,\sigma)\, p(\mu|\mu_\mu^0, \sigma_\mu^0)$$

distribution to sample — likelihood, specify last — hyperparameters — $\sim N(0,1)$, specify first

$D \to$ observed $\quad p(D|\mu,\sigma) \sim N(\mu,\sigma)$