Stan

Shubham Gupta

June 29, 2019

1 Introduction

- We will talk about Stan and implement a few simple models in it.
- Stan uses HMC(Hamiltonian Monte Carlo) for generating Monte Carlo steps.
- HMC is more effective for large samples.
- Stan is written in C++. It offers more programming flexibility.

2 HMC Sampling

- Problems with Gaussian distributions: They will always be centered around the current position(for both univariate and multivariate distributions). This can cause problems for data that have large tails.
- HMC used proposal distribution that changes depending on current position.
- HMC wraps posterior distribution around **gradient**.
- Metropolis algorithm would produce proposal jumps such that anything above or below the current position is equally probable.
- HMC generates distribution based on negative log of posterior density(called **potential**).

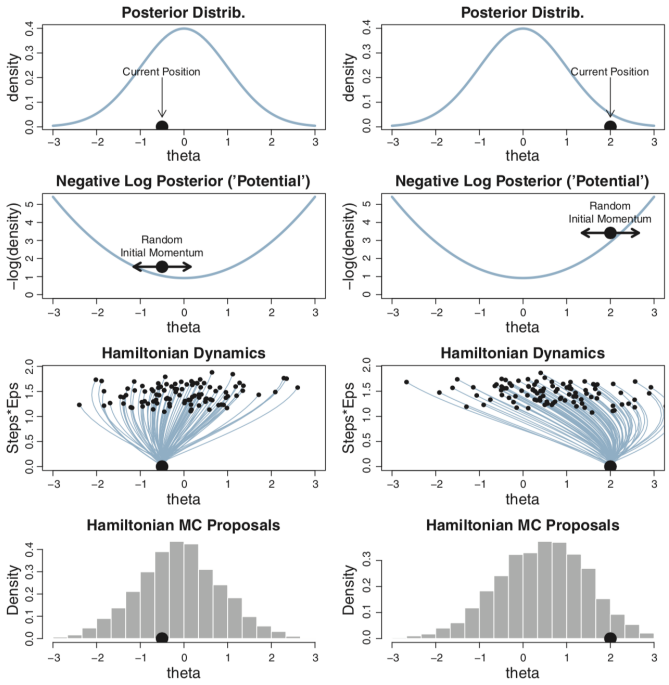

Figure 14.1 Examples of a Hamiltonian Monte Carlo proposal distributions. Two columns show two different current parameter values, marked by the large dots. First row shows posterior distribution. Second row shows the potential energy, with a random impulse given to the dot. Third row shows trajectories, which are the theta value (*x*-axis) as a function of time (*y*-axis marked Steps*Eps). Fourth row shows histograms of the proposals.

Figure 1: HMC proposal distributions

- Posterior tall $\implies$ Potential low and vice versa.
- Large dot represents the current position.
- Proposed position generated by flicking marble in **random** direction and letting it roll for some duration.
- Magnitude of flick is sampled from **zero mean Gaussian** distribution.
- Flick gives **momentum** to the ball. When time is up, the new position of the ball is the new proposed jump.
- Ball will be near the lower part of the curve i.e it will be in regions of higher posterior probability(because the curve is inverted posteior probability).
- **Potential** is similar to physics concept of potential energy.
- Third row in figure shows many trajectories taken by the ball from it's initial position and allowed to roll for random times.(impluses is called **Steps*Eps**. Trajectories show theta value as a function of time as the ball rolls after it receives the random initial push.
- Bottom row shows histogram of all proposed positions. Proposal distribution **is not** centered around initial position.

- Proposal distribution has moved towards the **mode** of the posterior distribution.
- **Metropolis decision rule**: Used to accept or reject proposal. ϕ is used to denote the momentum.
- In ideal system, all potential energy will be converted to kintetic energy when the marble is moving and vice versa when the marble is stable i.e lossless system.
- For ideal system, sum of potential($-log(p(\theta|D))$) and kinetic($-log(p(\phi))$ will be constant, and proposal will always be rejected.
- **However**, because there is always some loss in energy(due to friction) in real world scenarios, proposals will not always be accepted.

$$p_{accept} = min\left(\frac{p\theta_{proposed}|D)p(\phi_{proposed})}{p(\theta_{current}|D)p(\phi_{current})}, 1\right).$$

- Proposal distribution can be "tuned" by:
  1. Adjusting step size called **epsilon(or eps)**
  2. Adjusting number of steps taken.
  3. Standard deviation of the distribution from which the momentum is selected.
- Step size $\implies$ Time taken to make a step. Hence, it is called **Steps*Eps**
- General acceptance rate is 62%.
- Acceptance rate low $\implies$ Epsilon reduced
- Acceptance rate high $\implies$ Epsilon increased **and** changes in number of steps to maintain trajectory duration.
- Step size controls smoothness or jaggedness of trajectory.
- Duration(Steps*Eps) controls how far proposed steps are from the current position.

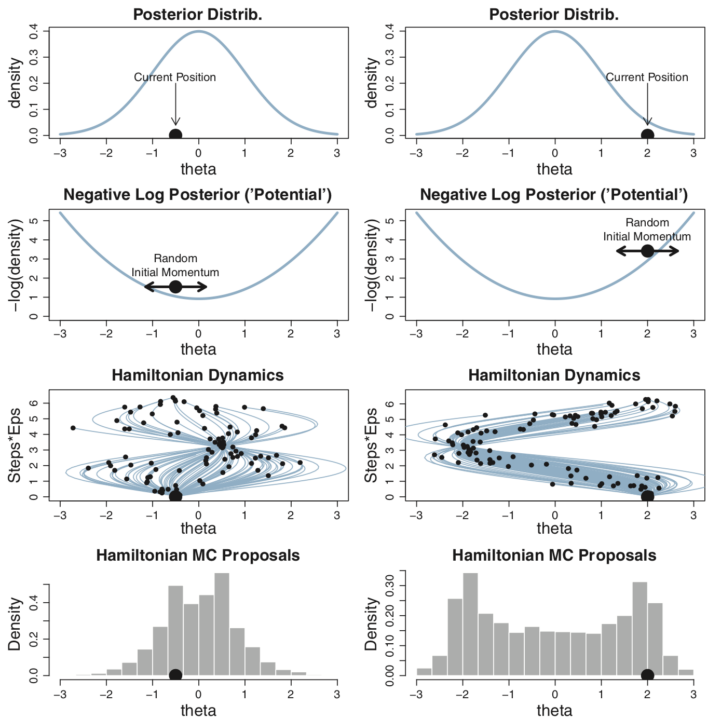

Figure 14.2 Examples of a Hamiltonian Monte Carlo proposal distributions for two different current parameter values, marked by the large dots, in the two columns. For this figure, a large range of random trajectory lengths (Steps*Eps) is sampled. Compare with Figure 14.1.

Figure 2: HMC proposal distributions for different trajectory lengths

- Duration is important because we want proposed position closer to mode of posterior distribution.
- As shown in figure, long trajectories overshoot the mode and return to the current position.
- **NUTS**(No U-Turn Sampling) algorithm stops trajectories before they make a U-turn and return to the starting position.

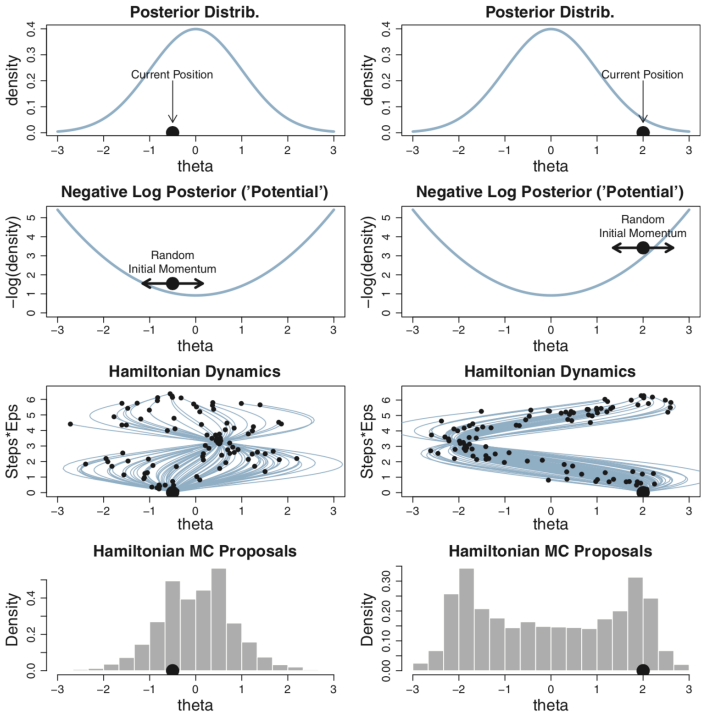

Figure 14.2 Examples of a Hamiltonian Monte Carlo proposal distributions for two different current parameter values, marked by the large dots, in the two columns. For this figure, a large range of random trajectory lengths (Steps*Eps) is sampled. Compare with Figure 14.1.

Figure 3: HMC proposal distributions for different standard deviation

- Standard deviation of momentum distribution wider $\implies$ Wider posterior
- SD of momentum distribution is selected in Stan using adaptive algorithms. It usually matches the SD of the posterior.
- Because it is difficult to find the gradient of functions spanning multiple dimensions, usual method is:
  1. Take half step in direction of gradient and update momentum
  2. Alternate full steps along gradients of potential and momentum
  3. Finally another half step of momentum(called **leapfrog** method).

2.1 Stan

- For number of iterations, use the following formula for stan:

$$chains = \frac{\text{(desired chains) * (iter - warmup)}}{\text{thin}}.$$

2.1.1 General structure

- Stan code can be organized as follows:

```
data {
... declarations ...
}
transformed data {
... declarations ... statements ...
}
parameters {
... declarations ...
}
transformed parameters {
... declarations ... statements ...
}
model {
... declarations ... statements ...
}
generated quantities {
... declarations ... statements ...
}
```

Figure 4: General structure of stan code

- Stan HMC works with probability of log posterior distribution and it's gradient. There is **no direct random sampling** of parameters from distributions.
- Hence, $y \sim normal(mu, sigma) \implies$ current posterior probability$*$density of normal distribution at$y$
- Therefore,

$$normal(mu, sigma) = \text{incrementlogprob}(\text{normallog}(y, mu, sigma)).$$

2.1.2 Sampling the prior in Stan

- Comment out likelihood statement from model definition to sample from prior.