MCMC Samplers

Braden Kronheim

May 6, 2020

1 Metropolis-Hastings

1.1 Metropolis Paper[1]

The initial motivation of this paper was to calculate the equilibrium value of some quantity F for a 2d system of interacting particles such that the potential energy between two particles is determined by the smallest distance between them, d_{ij}. The overall potential energy of a system is then

$$E = \frac{1}{2} \sum_{i,j,i \neq j}^{N} V(d_{ij}).$$

As this quantity is independent of the momentum of the particles, the potential energy is determined only by their positions, so

$$\bar{F} = \frac{\int F \exp(-E/kT) d^{2n}q}{\int \exp(-E/kT) d^{2n}q}$$

according to Boltzmann statistics. An initially proposed means of doing this is to choose a series of random configurations and weight them with $\exp(-E/kT)$. This method is rejected though, as the weights would likely all be very small. Instead, the paper proposes choosing states with probability $\exp(-E/kT)$ and weighting the sampled states equally.

The proposed method of doing this is as follows. First, choose a maximum distance α. Then, for each of the 2N positions, choose a random number β_n between -1 and 1, and change that coordinated by adding $\alpha\beta_n$. If the new position has a lower energy the state will be accepted, otherwise it will be accepted with probability $\exp(-\Delta E/kT)$. If a state is rejected, the current state becomes the new state. To measure the value of $\bar{F}$, the authors use the formula

$$\bar{F} = \frac{1}{M} \sum_{j=1}^{M} F_j.$$

To prove this method does choose states with a probability $\exp(-E/kT)$, the authors first note that as particles can move within a square with sides 2α

with nonzero probability, it is possible for any point in the space to be reached, meaning the system is ergodic. Next, they note that a priori, the probability a move will carry the system from state r to state s is P_{rs} and that this equals P_{sr}, prior to applying the $\exp(-\Delta E/kT)$ criteria. This means that for a single step with the 2α square the transition probabilities are equal, and 0 outside of the square. If $E_r > E_s$, then the flow from r to s will be $v_r P_{rs}$ where v_r is the probability of state r, and the flow from s to r will be $v_s P_{sr} \exp(-(E_r - E_s)/kT)$ due to the weighting. The net transfer from s to r is then

$$P_{rs}(v_s \exp(-E_r/kT) - v_r)).$$

From this, we can conclude that an equilibrium state is reached when this value is 0, or $v_r = v_s \exp(-(E_r - E_s)/kT)$, which means

$$\frac{v_r}{\exp(-E_r/kT)} = \frac{v_s}{\exp(-E_s/kT)}.$$

This is exactly what we want given Boltzmann statistics. Further, given the ergodicity of the system this will eventually hold for all states in the system. The remainder of the paper simply describes the application of the method to a specific problem.

1.2 Hastings Paper [2]

The Hastings Paper serves to generalize the method from the Metropolis paper and explore the relevant theory. The paper sets up the theory somewhat differently than Metropolis. First it says $\mathbf{P} = p_{ij}$ is the transition matrix of an irreducible Markov chain with states 0,1,..,S. The p_{ij} are such that the probability of $X(t+1) = j$ if $X(t) = i$ is p_{ij}. Further, it says that $\pi = (\pi_0, \pi_1, ..., \pi_S)$ is a probability distribution with positive values and f is some function defined on the states, then the desired quantity to estimate is

$$I = E_\pi(f) = \sum_{i=0}^{S} f(i)\pi_i.$$

Next, $\mathbf{P}$ is chosen so that π is its unique stationary distribution, or that it is an eigenvector with eigenvalue 1. I find the uniqueness of this interesting, as I am not sure what guarantees the existence of this eigenvalue of 1 and why it must be unique. This Markov chain is then simulated for times $t = 1, ..., N$ and the estimate

$$\hat{I} = \sum_{t=1}^{N} f(X(t))/N.$$

Citing a book on Markov Chains, the paper notes that $\hat{I}$ is asymptotically normally distributed and that it approaches I in mean square as $N \to \infty$.

Having shown this, next it is observed that as $X(t)$ is asymptotically stationary, the process $Y(t) = f(X(t))$ is as well, and for large enough N techniques for estimating the variance of the mean of a stationary process can be used.

Suppose we have a Markov Chain with N samples $y_i, \ldots, y_n$. The empirical mean of the samples is

$$\bar{Y} = E[Y] = E[\frac{1}{N}\sum_{n=1} Y_n] = \frac{1}{N}\sum_{n=1} E[Y_n]$$

and the variance is

$$Var(\bar{Y}) = Var(\frac{1}{N}\sum_{n=1} Y_n) = \frac{1}{N^2}Var(\sum_{n=1} Y_n) = \frac{1}{N^2}\sum_{i,j=1} Cov(Y_i, Y_j)$$

Now,

$$Cov(Y_i, Y_j) = E[Y_i Y_j] - E[Y_i]E[Y_j] = E[Y_i Y_j] - \bar{Y}^2$$

Define C_j to be the correlation, or

$$C_j = E[Y_i Y_{i+j}] - \hat{Y}^2$$

Thus,

$$Cov(Y_i, Y_j) = C_{j-i}.$$

Further, let $\rho_j = \frac{C_j}{C_0}$, so

$$Cov(Y_i, Y_j) = (E[Y_i^2] - E[Y_i]^2)\rho_{j-i} = \sigma^2 \rho_{j-i}.$$

From here, we can say that

$$Var(\bar{Y}) = \frac{\sigma^2}{N^2}\sum_{i,j=1} \rho_{j-i}.$$

Now, we note that there will be $N - |t|$ instances of $j - i = t - 1 - N$, so we rewrite the double sum as

$$Var(\bar{Y}) = \frac{\sigma^2}{N^2}\sum_{t=-(n-1)}^{n-1} (N - |t|)\rho_t = \frac{\sigma^2}{N}\sum_{t=-(n-1)}^{t-n-1} (1 - \frac{|t|}{N})\rho_t,$$

which is the first variance equation from the Hastings paper.

Now, we define the integrated time to be

$$\tau = \frac{1}{2}\sum_{t=-\infty}^{\infty} \rho_t.$$

Assuming a large N and that ρ_t is small for large t, we can approximate

$$Var(\bar{Y}) = \frac{1}{N}(2\tau)\sigma^2.$$

Assuming that all the samples are independent, this formula would have been

$$\frac{1}{N}\sigma^2,$$

so the effective sample size is

$$N_{eff} = \frac{N}{2\tau}$$

As far as I can tell, $2\tau = 2\pi g(0)$ where $g(\omega)$ is the spectral density of the chain at frequency ω, and that this just falls out by taking the correct formulation for the spectral density at plugging in a frequency of 0.

To obtain the next equation in the Hastings paper, we let

$$c_j = \sum_{t+1}^{N-j} Y(t)Y(t+j)/(N-j)$$

for $j \geq 0$ and $c_{-j} = c_j$. This is clearly an approximation of $E[Y_i Y_j]$, so C_j is about $c_j - \bar{Y}^2$, and ρ_j is about $(c_j - \bar{Y}^2)/\sigma^2$. Next, suppose that ρ_j is essentially zero for $j \geq j_0$.

Then we have

$$Var(\bar{Y}) = \frac{\sigma^2}{N^2} \sum_{t=-(j_0-1)}^{j_0-1} (N - |t|)(c_t - \bar{Y}^2)/\sigma^2.$$

Simplifying, we arrive at

$$Var(\bar{Y}) = \frac{1}{N} \sum_{t=-(j_0-1)}^{j_0-1} (1 - |t|/N)(c_t - \bar{Y}^2).$$

The issue though, is that according to the Hastings paper, the value out front should be

$$\frac{N}{(N-j_0)(N-j_0+1)}$$

I think my error is in the relationship between ρ_j and c_j, as saying

$$\rho_j = \frac{c_j - \hat{Y}^2}{\sigma^2} \frac{N^2}{(N-j_0)(N-j_0+1)}$$

would solve the issue, and it seems like this could stand.

The final equation is this section of the Hastings paper uses the pilot estimate correct for the mean of the the spectral density function at zero frequency. Given that I am unsure of exactly how the spectral density function relates, I am also unsure of how to prove this relationship. The method given is to divide the observations into L groups of K consecutive observations each, mark the mean of the ith block as

$$\bar{Y}_i = \sum_{t=1}^{K} Y\{(i-1)K + t\}/K,$$

and use the estimate

$$\sum_{i=1}^{L} (\hat{Y}_i - \hat{Y})^2/\{L(L-1)\}.$$

Without trying to prove this per say, the formula does bear a resemblance to

$$\tau = \frac{1}{2} \sum_{t=-\infty}^{\infty} \rho_t,$$

which makes sense as the spectral density function is also related to the value of τ, which is what is inside the parenthesis.

After examining these variance calculations, the paper examines how to generate these stationary distributions by how $\mathbf{P}$ can be built. First, it assumes that

$$p_{ij} = q_{ij}\alpha_{ij} (i \neq j)$$

and

$$p_{ii} = 1 - \sum_{j \neq i} p_{ij}$$

where $\mathbf{Q}$ is the transition matrix for some arbitrary Markov chain of the states 0, ..., S, and α_{ij} is

$$\alpha_{ij} = \frac{s_i j}{1 + \frac{\pi_i}{\pi_j}\frac{q_{ij}}{q_{ji}}}.$$

Here, s_{ij} is symmetric and chosen so that $0 \leq \alpha_{ij} \leq 1$ for all i, j. From this, it follows easily that

$$\pi_i p_{ij} = \frac{\pi_i q_{ij} s_{ij}}{1 + \frac{\pi_i}{\pi_j}\frac{q_{ij}}{q_{ji}}} = \frac{\pi_i q_{ij} s_{ij} \pi_j q_{ji}}{\pi_j q_{ji} + \pi_i q_{ij}} = \frac{\pi_j q_{ji} s_{ji}}{1 + \frac{\pi_j}{\pi_i}\frac{q_{ji}}{q_{ij}}} = \pi_j p_{ji}$$

when $i \neq j$ and trivially $\pi_i p_{ii} = \pi_i p_{ii}$ when $i = j$. Thus, in the stationary distribution we are indeed stationary, and there is not a net flow from one state to another.

This leads Hastings to propose the following simulation process. First, take $X(t) = i$ to be the state. Select a state j using the distribution of the ith row of $\mathbf{Q}$. Next, we say $X(t+1) = j$ with probability α_{ij} and $X(t+1) = i$ with probability $1 - \alpha_{ij}$. When choosing s_{ij}, Hastings only considers functions of the quantity $(\pi_j q_{ji})/(\pi_i q_{ij})$. Setting $q_{ij} = q_{ji}$ and

$$s_{ij} = \begin{cases} 1 + \frac{\pi_i q_{ij}}{\pi_j q_{ji}} & \frac{\pi_j q_{ji}}{\pi_i q_{ij}} \geq 1 \\ 1 + \frac{\pi_j q_{ji}}{\pi_i q_{ij}} & \frac{\pi_j q_{ji}}{\pi_i q_{ij}} \leq 1 \end{cases}$$

gives

$$\alpha_{ij} = \begin{cases} \frac{1+\frac{\pi_i}{\pi_j}}{1+\frac{\pi_i}{\pi_j}} & \pi_j \geq \pi_i \\ \frac{1+\frac{\pi_j}{\pi_i}}{1+\frac{\pi_i}{\pi_j}} & \pi_j \leq \pi_i \end{cases}$$

which becomes

$$\alpha_{ij} = \begin{cases} 1 & \pi_j \geq \pi_i \\ \frac{\pi_j}{\pi_i} & \pi_j \leq \pi_i \end{cases}$$

. This is the accept reject formula for the Metropolis sampler. A more general form of s_{ij} that can be picked is

$$s_{ij} = g[\min\{(\pi_i q_{ij})/(\pi_j q_{ji}), (\pi_j q_{ji})/(\pi_i q_{ij})\}].$$

We must pick $g(x)$ so that $0 \leq g(x) \leq 1 + x for 0 \leq x \leq 1$ and $g(x)$ is symmetric in i and j.

The rejection rate is defined as the proportion of the times t where $X(t+1) = X(t)$.

2 Gibbs Sampler

2.1 Geman and Geman paper[3]

The objects of interest in the Gibbs Sampler are as follows. $X = \{X_S, s \in S\}$ is a Markov Random Field over a graph $\mathcal{G}, s \in S$ with state spaces Λ_s, configuration space $\Omega = \Pi_s \Lambda_s$ and Gibbs distribution $\pi(\omega) = e^{-U(\omega)/T}/Z, \omega \in \Omega$. We define X to be a a Markov Random field (MRF) with respect to a neighborhood system $\mathcal{G}$ iff $\pi(\omega) = P(X = \omega)$ is a Gibbs distribution with respect to $\mathcal{G}$. A neighborhood system $\mathcal{G}$ is defined for a set of sites $S = s_1, s_2, \cdots, x_N$ as any collection of subsets for S such that $s \notin \mathcal{G}_s$ and $s \in \mathcal{G}_r$ iff $r \in \mathcal{G}_s$. Essentially, this means an element can't be its own neighbor, and if a is a neighbor of b, then b is a neighbor of a. A Gibbs distribution for a set of site S and neighborhoods $\mathcal{G}$ is a probability measure π on Ω such that

$$\pi(\omega) = \frac{1}{Z} e^{-U(\omega)/T}.$$

Further, the energy function $U(\omega)$ is defined as

$$U(\omega) = \sum_{C \in \mathcal{C}} V_C(\omega).$$

Here, $\mathcal{C}$ denotes the set of cliques of $\mathcal{G}$, where a subset $C \subseteq S$ is a clique if every pair of distinct sites in C are neighbors. The V_c are functions on Ω such that $V_C(\omega)$ depends only on the coordinates x_s of ω for which $s \in C$. The family $\{V_c, C \in \mathcal{C}\}$ is a potential. Z is a normalizing constant, or the partition function. T is a temperature where a small value emphasizes the mode.

The general goals of the Gibbs sampler are similar to those of the Metropolis-Hastings sampler. Namely, it seeks to:

1) sample from the distribution π

2) minimize U over Ω

3) compute expected values

The first and third goals are truly the same goals as the Metropolis-Hastings sampler. The second goal, however, is new and is a form of maximum a posteriori estimates, achieved through a relaxation algorithm where the temperature stars high and then decreases. This goal is the main focus of the paper as they are

seeking to improve degraded images. For the purpose of this class, however, the first and third goals are much more interesting.

The authors of the paper describe a highly parallel algorithm where each of sites s is given a processor, and each processor is connected the processors which are neighbors of its site. At a time t, each site has state $X_s(t)$ and the overall configuration is

$$X(t) = (X_{s1}(t), \cdots, X_{sN}(t)).$$

At time $t = 0$ the states are arbitrary. For each increment of t only one site will have the possibility of changing, the others will remain constant. When it is time to update a site, the processor draw a sample from the local characteristics of π for $s = n_t$ and $\omega = X(t-1)$. Stated another way, a state $x \in \Lambda_{nt}$ is chosen from the conditional distribution of X_{nt} given the observed states of the neighboring sites.

To prove that this works, we note first that $\{X(t), t \geq 0\}$ is in fact a Markov Chain given its construction. Let us fix t and $\omega \in \Omega$. For any $x \in \Lambda$, we say ω^x is the configuration which has value x and site n_t and agrees with ω everywhere else. The transition matrix from ω to η is then

$$(M_t)_{\eta,\omega} = \begin{cases} \pi(X_{n_t} = x_{n_t} | X_s = x_s, x \neq n_t), & \exists x \in \Lambda \text{ s.t. } \eta = \omega^x \\ 0 & \text{else} \end{cases}$$

Note that this chain is not stationary as the majority of states will be not reached in one step, though it is irreducible since $\pi(\omega) > 0$.

If we have a starting distribution μ_0, then the distribution of $X(t)$ will be formed by multiplying the transition matrix by itself t times and acting this on the vector, or

$$\mu_0 \Pi_{j=1}^t M_j.$$

We can also get that

$$P_{\mu_0}(X(t) = \omega) = \sum_\eta P(X(t) = \omega | X(0) = \eta) \mu_0(\eta).$$

Now, consider the action of the chain on the unique invariant vector π. We get

$$(\pi M_t)_\omega = \sum_\eta P(X(t) = \omega | X(0) = \eta) \pi(\eta).$$

Let $\omega = \{x_s\}$ and t be fixed. Then

$$\begin{aligned} (\pi M_t)_\omega &= \sum_\eta (M_t)_{\eta,\omega} \\ &= \sum_{x \in \Lambda} \pi(\omega^x)(M_t)_{\omega^x,\omega} \\ &= (M_t)_{\omega^{x'},\omega} \sum_{x \in \Lambda} \pi(\omega^x) \text{ (for any } x' \in \Lambda) \\ &= (\pi(X_{n_t} = x_{n_t} | X_s = x_s, s \neq n_t) \pi(X_s = x_s, s \neq n_t)) \\ &= \pi(\omega) \end{aligned} \tag{1}$$

Theorem A: Assume that for each $s \in S$, the sequence $\{n_t, t \geq 1\}$ contains s infinitely often. Then for every starting configuration $\eta \in \Omega$ and for every $\omega \in \Omega$,

$$\lim_{t\to\infty} P(X(t) = \omega | X(0) = \eta) = \pi(\omega).$$

Let $T_0 = 0$ and $T_1 < T_2 < \cdots$ be times such that

$$S \subseteq \{n_{1+T_{k-1}}, n_{2+T_{k-1}}, \cdots, n_{T_k}\}, k = 1, 2, \cdots .$$

What this represents, is a single loop through all of the sites for each k. This can be done as each site is visited infinitely often, and implies that at least k full sweeps of all the sites have occurred by time T_k, or $kN \leq T_k < \infty \; \forall \; k$. We define $K(t)$ to be the time of the last full sweep as, or $K(t) = \sup\{k : T_k < t\}$. Clearly, $K(t)$ will approach infinity as time does. To continue from here, we state the following lemma: Lemma 1: There exists a constant $r, 0 \leq r < 1$ such that for every $t = 1, 2, \cdots$,

$$\sup_{\omega,\eta',\eta''} |P(X(t) = \omega | X(0) = \eta') - P(X(t) = \omega | X(0) = \eta'')| < r^{K(t)}.$$

Essentially, this means that the impact on the starting point of the chain goes to zero in infinity norm as time goes to infinity. For now, we assume this lemma is true. Then, since π is an invariant vector for the chain,

$$\lim_{t\to\infty} \sup_{\omega,\eta} |P(X(t) = \omega | X(0) = \eta) - \pi(\omega)| =$$

$$\lim_{t\to\infty} \sup_{\omega,\eta} \left| \sum_{\eta'} \pi(\eta') \{P(X(t) - \omega | X(0) = \eta) - P(X(t) = \omega | X(0) = \eta')\} \right|$$

We can see that the following quantity is larger,

$$\lim_{t\to\infty} \sup_{\omega,\eta',\eta''} |P(x(t) = \omega | X(0) = \eta') - P(X(t) = \omega | X(0) = \eta'')|$$

Finally, from the lemma this must converge to 0, so

$$\lim_{t\to\infty} P(X(t) = \omega | X(0) = \eta) = \pi(\omega).$$

The actual proof of the lemma can be seen in the origianl paper as it is fairly technical.

3 Slice Sampler[4]

The single variable slice sampler algorithm for sampling from the distribution $f(x)$ starting at x_0 consists of the following steps:

1. Uniformly select y from the range $(0, f(x_0))$. This defines a horizontal slice such that $S = \{x : y < f(x).\}$

2. Determine an interval $I = (L, R)$ around x_0 that contains the most if not all of the slice

3. Sample a new point x_1 from the part of the slice in the interval

To prove that this works, suppose we have an initial state x_0 which has distribution $f(x)$. In the first step of the algorithm a value y is drawn uniformly from $(0, f(x))$. This makes the joint distribution of $y, f(x)$

$$p(x, y) = \begin{cases} 1/Z, & 0 < y < f(x) \\ 0 & \text{else} \end{cases}$$

where $Z = \int f(x)dx$. The reason for this is that any combination of x, y must be under the curve of $f(x)$. However, it is clear that any point under the curve is equally likely, so they have probability proportional to 1 over the area, or $1/Z$. The marginal distribution for x of this joint distribution is

$$\int_{y \in \mathbb{R}} p(x, y)dy = \int_0^{f(x)} (1/Z)dy = f(x)/Z.$$

Now, suppose we move from x_0 to x_1 and leave the joint distribution invariant. If we discard y the distribution of x_1 will be the marginal we just calculated, which is just the desired one from $f(x)$. This means we just have to prove that steps 2 and 3 of the sampler leave the joint distribution constant. In other words, this means we have to prove that the move from x_0 to x_1 produces x, y pairs with the correct probability. If it does, we can ignore the selected y value and just take the sample of x.

Now, the step of moving from x_0 to x_1 keep y the same, but changes x. Thus, $p(y)$ will stay the same. As $p(x, y) = p(x|y)p(y)$, we have to show the conditional distribution $p(x|y)$ is not affected by our movement. The conditional distribution is uniform over the slice $S = \{x : y < f(x)\}$, so we must show this distribution stays uniform. This will be true if for any x_0 and x_1 the probability of transitioning from x_0 to x_1 given a y value is equal to the probability of transitioning from x_1 to x_0, as this means there will be no net flow of probability from one of these states to the other. The general algorithm says nothing about how these regions are picked or sampled from, but assuming they are done in such a way as to satisfy this condition, they will work. A stronger, but more useful form of this requirement is

$$P(\text{next state} = x_1, \text{intermediate choices} = r | \text{ current state} = x_0)$$

$$= P(\text{next state} = x_0, \text{intermediate choices} = \pi(r) | \text{ current state} = x_1)$$

where $\pi(r)$ is some one to one function with Jacobian of 1. Here, if we integrate over all the possible r values we get the previous condition. The r values represent possible random decisions made in the selection of the interval or the sampled value.

This can easily be extended to higher dimensions. Suppose we are in n dimensions with state $\mathbf{x_0}$. Now we evaluate $f(\mathbf{x_0})$ and sample a y between 0 and this value which we use to define an n dimensional slice. Next, we define some sort of n dimensional interval that contains all or most of the slice, and sample from that slice. The proof that this works is basically the same, except the area Z is replaced by the volume of the $n+1$ dimensional space beneath the n dimensional probability density. Again, if we can prove that

$$P(\text{next state} = \mathbf{x_1}, \text{intermediate choices} = r | \text{ current state} = \mathbf{x_0})$$

$$= P(\text{next state} = \mathbf{x_0}, \text{intermediate choices} = \pi(r) | \text{ current state} = \mathbf{x_1})$$

where $\pi(r)$ is some one to one function with Jacobian of 1, then the multidimensional slice sampler will work.

Of the various implementations of these sampler that were discussed, the one that seemed best to me was the mirror sampler. In this method, the interval is defined as all parts of the slice, though the exact bounds of the slice are never calculated. Instead, a random momentum $\mathbf{p}$ is given and for a certain number of steps T the value x is updated via the formula $\mathbf{x} = \mathbf{x_0} + \mathbf{p}\omega$, where ω is a scale parameter. After each step, f is evaluated at the new spot. If it is found that the new spot is outside of the slice, the momentum is reflected across the gradient of f at the point according to the formula

$$\mathbf{p}' = \mathbf{p} - 2\mathbf{h}\frac{\mathbf{p} \cdot \mathbf{h}}{|\mathbf{h}|^2}.$$

After proceeding for T steps, the final state is accepted if inside the slice, and rejected otherwise.

Now, we must show that for a given y, the probability of transitioning from$\mathbf{x_0}$ to $\mathbf{x_1}$ is the same as the probability of transitioning from $\mathbf{x_1}$ to $\mathbf{x_0}$. Suppose the state is rejected. In order to do this, we chose some $\mathbf{p}$ in state $\mathbf{x_0}$ which had a specific probability. In order to transition from the end state to the start state, we must simply choose the same starting momentum, as this will result in a rejection. The probability of choosing the momentum in the first case is equal to the probability of choosing it in the second case trivially, so a rejected state preserves the flow of probability. Next, suppose the state is accepted. We can traverse the path from the new state back to the first state simply by reversing the momentum. Thus, the probability of going from the first state to the second state is the same as the probability of going from the second state to the first state, if starting momentum occurs with the same likelihood as the negative of the ending momentum.

Suppose the momentum are chosen from a normal distribution in each coordinate with mean 0 and standard deviation 1. Clearly, the probability of $\mathbf{p}$ and $-\mathbf{p}$ will be the same as the normal distribution is symmetric. If we can show that reflection about an arbitrary axis leaves the likelihood unchanged then we will be done. Clearly, our distribution of $\mathbf{p}$ is rotationally symmetric, as our distribution has a covariance matrix which is the identity matrix. For a vector,

reflection across another vector is equivalent to rotation through twice the angle between them, so the distribution will be left unchanged through reflection as well. Thus, the probability of $\mathbf{p}$ is left unchanged through reflections, so in all cases the probability of transitioning from $\mathbf{x_0}$ to $\mathbf{x_1}$ is the same as the probability of transitioning from $\mathbf{x_1}$ to $\mathbf{x_0}$.

4 Multi Try Monte Carlo[5]

The idea behind multi try Monte Carlo is an attempt to increase the possible step size of a generalized Metropolis-Hastings sampler by drawing multiple samples for each step. This is done in such a way as to boost the overall acceptance rate of new states. The algorithm allows for a proposal transition function $T(x, y)$ which need only satisfy $T(x, y) > 0$ iff $T(y, x) > 0$. We also let $\lambda(x, y)$ be a non-negative symmetric function in x and y which the user is free to choose. Beyond symmetry, it must also satisfy $\lambda(x, y) > 0$ when $T(x, y) > 0$. Let the current state be x. We define

$$\omega(x, y) = \pi(x)T(x, y)\lambda(x, y)$$

Then we transition to the next state as follows.

1. Draw k independent and identically distributed trial proposals $y_1, \ldots, y_k$ from $T(x, \cdot)$ and compute $\omega(y_j, x)$ for $j = 1, \ldots, k$
2. Select y from the set $\{y_1, \ldots, y_k\}$ with probability proportional to $\omega(y_j, x)$. Next, select $x_1^*, \ldots, x_{k-1}^*$ from $T(y, \cdot)$ and let $x_k^* = x$.
3. Finally, state y is accepted with probability

$$r_g = \min\left\{1, \frac{\omega(y_1, x) + \cdots \omega(y_k, x)}{\omega(x_1^*, y) + \cdots \omega(x_k^*, y)}\right\}$$

In order to prove that this sampler works, we just must prove that it satisfies detailed balance, thus producing a reversible Markov chain with π as its invariant distribution. To do this, we define $A(x, y)$ to be the actual transition probability of traveling from x to y in the sampler. Trivially, the probability of traveling from x to y will be the same in both directions when $x = y$. Suppose $x \neq y$ and I is the y_j selected in step 2. The event $A(x, y)$ corresponds to the union of all cases where $y_j = y$ and $I = j$ given x. Each of these cases is independent and of equal probability as there is no reason why a particular y_j need be selected at a certain position. Thus,

$$A(x, y) = kP[(y_k = y) \cap (I = k)|x]$$

where k is the number of sample drawn. This also means that

$$\pi(x)A(x, y) = k\pi(x)P[(y_k = y) \cap (I = k)|x]$$

which is simply the overall probability of the transition from x to y occurring. The probability itself is

$$\int \cdots \int T(x,y)T(x,y_1)\cdots T(x,y_{k-1})\frac{\omega(y,x)}{\omega(y,x)+\sum_{j=1}^{k-1}\omega(y_j,x)}\min\left\{1,\frac{\omega(y,x)+\sum_{j=1}^{k-1}\omega(y_j,x)}{\omega(x,y)+\sum_{j=1}^{k-1}\omega(x_j^*,y)}\right\}\times$$

$$T(y,x_1^*)\cdots T(y,x_{k-1}^*)dy_1\cdots dy_{k-1}dx_1^*\cdots dx_{k-1}^*$$

. The terms in this integral are as follows. $T(x,y)T(x,y_1)\cdots T(x,y_{k-1})$ is simply the probability of selecting the set of y_j given our choice of x. The probability of selecting y given this set is $\frac{\omega(y,x)}{\omega(y,x)+\sum_{j=1}^{k-1}\omega(y_j,x)}$. Given this y, the probability of selecting our set of x^* is just $T(y,x_1^*)\cdots T(y,x_{k-1}^*)$. Finally, the probability of actually accepting this state, which was assumed when we said $x\neq y$, is

$$\min\left\{1,\frac{\omega(y,x)+\sum_{j=1}^{k-1}\omega(y_j,x)}{\omega(x,y)+\sum_{j=1}^{k-1}\omega(x_j^*,y)}\right\}.$$

The integrals are over all possible y_j and x^* values other than the fixed two, as they do not have to be fixed values for the state to be proposed or accepted. By moving the denominator of the acceptance probability for y into the accept-reject probability and pulling the numerator out of the integral along with $T(x,y)$ we get

$$\omega(y,x)T(x,y)\int\cdots\int\min\left\{\frac{1}{\omega(y,x)+\sum_{j=1}^{k-1}\omega(y_j,x)},\frac{1}{\omega(x,y)+\sum_{j=1}^{k-1}\omega(x_j^*,y)}\right\}\times$$

$$T(x,y_1)\cdots T(x,y_{k-1})T(y,x_1^*)\cdots T(y,x_{k-1}^*)dy_1\cdots dy_{k-1}dx_1^*\cdots dx_{k-1}^*$$

. Note that the integral is now symmetric in x and y as the y_j and x_j^* will all take on the same values. Going back to the expression $\pi(x)A(x,y)$ and ignoring the quantity inside the integral we get

$$k\pi(x)\omega(y,x)T(x,y).$$

As $\omega(y,x)=\pi(y)T(y,x)\lambda(y,x)$ our expression is

$$k\pi(x)\pi(y)T(x,y)T(y,x)\lambda(y,x).$$

This is also symmetric in x,y and λ is symmetric. Thus,

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

which is the detailed balance condition. Therefore, the sampler works.

The specific implementation of the algorithm I implemented was for

$$\lambda(x,y)=\left(\frac{T(x,y)+T(y,x)}{2}\right)^{-1}$$

If we choose a symmetric T, as is done in the normal Metropolis-Hastings sampler, this just becomes $\lambda(x,y) = \frac{1}{T(x,y)}$ This means

$$\omega(x,y) = \frac{\pi(x)T(x,y)}{T(x,y)} = \pi(x),$$

so the accept-reject probability is

$$\min\left\{1, \frac{\pi(y_1) + \cdots_{\pi}(y_k)}{\pi(x_1^* + \cdots + \pi(x_k^*))}\right\}.$$

5 Langevin Monte Carlo [6]

The Langevin Monte Carlo sampler takes the form of a generalized Hastings style sampler. Specifically, given the sate x it will propose state y with probability $q(x,y)$. This new state will then be accepted with probability

$$\alpha(x,y) = \begin{cases} \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)} & \pi(x)q(x,y) \geq 0 \\ 1 & \pi(x)q(x,y) = 0 \end{cases}$$

The proposal distribution is based upon the idea of Langevin diffusion, a process which actually converges to the stationary distribution π on its own. It requires that π is nonzero and differentiable everywhere. The process, denoted by $\mathbf{L}_t$ is defined as

$$d\mathbf{L}_t = d\mathbf{W}_t + \frac{1}{2}\nabla \log \pi(\mathbf{L}_t)dt$$

where $\mathbf{W}_t$ is Brownian motion. The paper shows that the diffusion will converge to π exponentially quickly when π has a tail which is no heavier than exponential. I will not be going into the proofs of this, as it requires a lot of measure theory and higher mathematics I am unfamiliar with.

The basic Langevin method assumes a continuous solution to the differential equation, but this is not possible in reality, so a democratization must be used. The unadjusted algorithm proposed by the paper is to use a first-order Gaussian approximation and choose state U_n given state U_{n-1} from the distribution

$$N(U_{n-1} + \frac{1}{2}h\nabla \log \pi(U_{n-1}), hI_k)$$

where h is a step size, and I_k is the k dimensional identity matrix. The problem with this implementation of the algorithm is that it does not have the same rapid convergence of the continuous version, and can converge to some distribution other than π. Thus, this basic implementation is not particularly useful.

The Metropolis-adjusted version of the algorithm introduces the Metropolis-Hastings accept-reject criteria initially discussed. This ensures that the Markov chain will in fact converge to the distribution π. This method works, although the rate of convergence is not as good as the continuous Langevin dynamics.

Specifically, tails lighter than Gaussian will make the convergence not exponential.

From actually implementing this algorithm and observing it work I noticed a particular sub optimal feature. Specifically, when the algorithm is started in an area where the derivative is much larger in one dimension than in the other, such as in the elliptical doughnut well that is 10 times wider in one dimension than the other, the algorithm will have a very hard time making its way down the probability slope. This is because The likelihood of the end location relative to the start location is not enough to compensate for the very low likelihood of reversing the chain. This can cause the algorithm to also get stuck if one of these points gets accepted even after it has reached the higher probability area. While this algorithm may work better than normal Metropolis-Hastings in roughly Gaussian distributions, it fails miserably in more complex distributions such as this one.

6 Hamiltonian Monte Carlo [7]

In Hamiltonian Monte Carlo we take the general Metropolis-Hastings setup and, in a manner similar to Langevin dynamics, replace the random step with a more complex movement. In this case, it is the simulation of the movement of a particle under Hamiltonian dynamics. Hamiltonian dynamics is a formulation for the motion of a particle based on a function of position q and momentum p called the Hamiltonian $H(q,p)$. For physical systems the Hamiltonian is the energy of the system. The set of differential equations guiding how these variables change over time is

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}.$$

A key feature of Hamiltonian dynamics is that it is reversible. That is if we go from $(q(t), p(t))$ to $(q(t+s), p(t+s))$ and then back to $(q(t), p(t))$ according the Hamiltonian dynamics we will end up at the same spot. This is because Hamiltonian dynamics do not dissipate energy, and when dealing with deterministic mechanics in this situation movement is symmetric with time. Another property is that Hamiltonian dynamics preserves the overall volume of the space. This is because the Hamiltonian has zero divergence, which can be shown to imply the preservation of volume.

In order to actually use Hamiltonian dynamics we need a means of approximating these differential equations. The method typically used is the leapfrog method which is time symmetric and volume preserving. The second order

version is

$$p_i(t+\epsilon/2) = p_i(t) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t)) \tag{2}$$

$$q_i(t+\epsilon) = p_i(t) - \epsilon\frac{p_i(t+\epsilon/2)}{m_i} \tag{3}$$

$$p_i(t+\epsilon) = p_i(t+\epsilon/2) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t+\epsilon)). \tag{4}$$

While not mentioned in Neal's text, two fourth order integrators which theoretically have the same properties are the Forest-Ruth algorithm and the PEFRL algorithm. Upon implementing them in the code, however, they actually performed significantly worse than the basic leapfrog algorithm, implying that something else is impacting their performance, or I have them implemented incorrectly.

Having figured out the implementation of the Hamiltonian dynamics it remains to be seen how it can be incorporated into the HMC framework. First, this is done by setting

$$H(q,p) = U(q) + K(p)$$

where the potential term U is the negative log probability of the parameters and the kinetic term is $\frac{p^2}{2m}$. From this, we form a joint distribution of $P(q,p)$ as

$$P(q,p) = \frac{1}{Z}\exp(-H(q,p)/T).$$

For each HMC step we sample a p with random direction and a magnitude draw from a Gaussian distribution. The leapfrog algorithm is simulated with a stepsize ϵ for a number of leapfrog steps L. At the end, a state (q^*, p^*) is accepted with probability

$$\min\left[1.\exp(-H(q^*,p^*) + H(q,p))\right].$$

In order to show that this process actually produces samples from the desired distribution, we must show that

$$P(A_k)T(B_k|A_k) = P(B_k)T(A_k|B_k)$$

where A_k and B_k are states, and T is the transition probability from one state to another. Rewriting this equation in terms of our probability function $P(q,p)$ we end up with

$$\frac{1}{Z}\exp(-H_{A_k})\min\left[1,\exp(-H_{B_k}+H_{A_k})\right] = \frac{1}{Z}\exp(-H_{B_k})\min\left[1,\exp(-H_{A_k}+H_{B_k})\right].$$

This can be rewritten as

$$\frac{1}{Z}\min\left[\exp(-H_{A_k}),\exp(-H_{B_k})\right] = \frac{1}{Z}\exp(-H_{B_k})\min\left[\exp(-H_{B_k}),\exp(-H_{A_k})\right]$$

which is trivially true. Thus, HMC works.

7 No-U-Turn Sampler [8]

The motivation behind this sampler is to run an HMC chain for as many leapfrog steps as possible, until the trajectory starts to double back on itself, or make a u-turn. The criteria used by the paper to determine when this u-turn has begun is

$$(\theta' - \theta) \cdot r' <= 0$$

where θ', r' are the new position and momentum. The reason for this is that if we take the derivative of the the distance squared over two between between the new states, and look for when it starts decreasing, we will have found a u-turn location. This comes down to the equation

$$\frac{d}{dt}\frac{(\theta' - \theta) \cdot (\theta' - \theta)}{2} = (\theta' - \theta) \cdot \frac{d}{dt}(\theta' - \theta) = (\theta' - \theta) \cdot r'.$$

The difficulty with this approach, is that a naive implementation where we pick a direction and let the sampler run until it starts to turn around, may not be time reversible. Thus, the algorithm needs to be more complicated. The method proposed in the paper makes use of concepts from slice sampling. Specifically, it has a slice variable u with conditional distribution

$$p(u|\theta, r) = \text{Uniform}(u; [0, \exp\{-H(\theta, r)\}]).$$

This makes the conditional distribution of θ, r to be

$$(\theta, r|u) = \text{Uniform}(\theta, r; \{\theta', r' | \exp\{-H(\theta, r) \geq u.\})$$

The basic algorithm works as follows. First, sample $u|\theta, r$. For the leapfrog integrator forward or backward 1 step. Then two steps, then 4 steps, and so on. During this doubling process, a balanced binary tree of position-momentum states is created, with the previously sampled locations being the left or right subtree, and the newly sampled points being the other. This process continues until any one of the individual subtrees starts to make a u-turn. Once this occurs, the the sampler then takes a random sample of the points which will preserve detailed balance when transferred to. The paper proposes two implementations of the algorithm, a simpler but slower and more resource intensive version, and a more complex but better algorithm.

In the sampler in general we have distribution $p(\theta, r) \propto \exp\{H(\theta, r)\}$ typical of HMC which is augmented with the slice variable u to give

$$p(\theta, r, u) \propto \mathbb{I}[u \in [0, \exp\{-H(\theta, r)\}]].$$

Here $\mathbb{I}[\cdot]$ is 1 when the condition inside is true and 0 otherwise. As mentioned previously, $p(u|\theta, r)$ and $p(\theta, r|u)$ are uniform over the regions they are not zero. In addition to these variables, the set $\mathcal{C}$ is the set of all points the algorithm can transition to without violating detailed balance, and $\mathcal{B}$ is the set of all traced out points. We build the set $\mathcal{B}$ according to the random binary tree procedure and determine $\mathcal{C}$ from $\mathcal{B}$. The exact process for this must satisfy

1. All elements in $\mathcal{C}$ must preserve volume
2. $p((\theta, r) \in \mathcal{C}|\theta, r, u, \epsilon) = 1$
3. $p(u \leq \exp\{-H(\theta, r)\}|(\theta', r') \in \mathcal{C} = 1)$
4. If $(\theta, r) \in \mathcal{C}, (\theta', r') \in \mathcal{C}, \forall \mathcal{B}, p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon) = p(\mathcal{B}, \mathcal{C}|\theta', r', u, \epsilon)$

Assuming we have a means of sampling $\mathcal{B}, \mathcal{C}$ is the specified manner, the sampling procedure is

1. sample $r \sim \mathcal{N}(0, I)$
2. sample $u \sim \text{Uniform}([0, \exp\{-H(q, t)\}])$
3. sample $\mathcal{B}, \mathcal{C}$ from their distribution
4. sample $\theta^{t+1}, r \sim T(\theta^t, r, \mathcal{C})$.

Here, T is the transition kernel which leaves the uniform distribution over $\mathcal{C}$ invariant.

The first three steps are all sampling variables from their conditional distributions, so are valid Gibbs updates. With the fourth step, we note that

$$p(\theta, r|u, \mathcal{B}, \mathcal{C}, \epsilon) \propto p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)p(\theta, r|\mathcal{B}, \mathcal{C}) \propto p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)p(\theta, r|u)$$

The first equality comes from some simple algebra using conditional distributions. The second equality it is because the set of points determined by $\mathcal{B}, \mathcal{C}$ should be the same as those determined by u. From here, we can note that

$$p(p(\theta, r|u)) = \mathbb{I}[u \leq \exp\{-H(\theta, r)\}]$$

as the elements in $\mathcal{C}$ preserve volume. Additionally, we are assured that

$$p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon) \propto \mathbb{I}[(\theta, r) \in \mathcal{C}]$$

from the 2nd and fourth condition. The third condition guarantees that $(\theta, r) \in \mathcal{C} -> u \leq \exp\{-H(\theta, r)\}$. Thus,

$$p(\theta, r|u, \mathcal{B}, \mathcal{C}, \epsilon) \propto \mathbb{I}[\theta, r] \in \mathcal{C}.$$

This means the final update is also a valid Gibbs update so long as the transition kernel leaves $\mathcal{C}$ invariant and uniform. The difference between the simpler and more complex variations of the algorithm is largely in the transition kernel.

The process for building a tree of high j involves j random choices in build direction. As these choices are random, a given tree is equally likely to be generated starting from any of the leaf nodes. We stop the expansion of the tree when it starts turning back on itself, or if the error in simulation becomes quite large. The second condition is triggered when any leaf node has a point which is sufficiently higher in energy than the slice value. The first condition is triggered when any of the balanced binary subtrees indicate u-turn behavior.

When choosing $\mathcal{C}$ we must satisfy the four conditions previously set out. The first condition is satisfied as the leapfrog steps are volume preserving. The second is satisfied if we include the starting location. The third is we reject any points not in the slice. The fourth is satisfied as long as we exclude any point which could not have generated $\mathcal{B}$. The only way such a point could exist is if it satisfies the stopping condition prior to the complete tree's construction.

This can happen if the stopping condition was satisfied by a state or subtree in the final doubling iteration, unless that state is the last state added or the subtree is the full tree. If this is the case, it is possible the tree would terminate earlier if started from one of these states. They can simply be excluded.

In the iterative version of this I wrote we first set a limit for the maximum tree depth. Then, it picks a random direction and follows the correct number of leapfrog steps. Along each the chain, the algorithm save all the states, momenta, and energy. Next, all the states are compared that are the end of subtrees. After this, we heck that no u-turn has occurred and that none of the states are so far out of the slice to warrant ending the chain. From here, we select a state randomly from those on the chain inside the slice. We compare the end state of the new chain to the other end of the total tree. Then, we select the new state with probability proportional to the number of acceptable states in the slice over the total number of acceptable states thus far. After this, add the total number of states to a counting variable and add the sum of all the Metropolis-Hastings accept probabilities to a new counting variable Finally, this is repeated until a stopping condition is reached.

Beyond the actual No-U-Turn sampler, the paper also introduced the idea of dual averaging. Suppose we want to make the expectation of some statistic H to be 0, where the expectation is

$$h(x) = \lim_{t\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[H_t|x]$$

For example, this could be the acceptance rate α_t and we want to make it δ, so

$$H_t = \delta - \alpha_t$$

Given a few conditions on h, this happens if

$$x_t - \eta_t H_t \to x_{t+1}$$

This approach works, but tends to give significant weight to the early examples, which we do not want if this is done over the burnin period, as the ideal value may changing. Dual averaging is a slightly different method which is weighted more heavily toward early examples. It is

$$\mu - \frac{\sqrt{t}}{\gamma} \frac{1}{t+t_0} \sum_{i=1}^{t} H_i \to x_{t+1}$$

$$\eta_t x_{t+1} + (t - \eta_t)\bar{x}_t \to \bar{x}_{t+1}$$

This method can be applied with a decaying update rate, or it can just be applied during burnin to allow convergence to the correct distribution.

Having run the algorithm and implemented it I have come up with a few observations. First, assuming the chain was simulated with high acceptance, we would on average expect the NUTS sampler to move no farther than half its maximum distance, as the new state must be sampled uniformly across the chain. Additionally, when using the fourth order leapfrog solver, the step sizes tend to be larger, showing that it works as a fourth order solver. In fact, with the donut shape, it gets really large step sizes and essentially becomes a normal metropolis Hastings sampler, with very few leapfrog steps It is possible that this could be fixed by measuring the acceptance rate across only the last subtree, as I think what is happening is that it travels a large distance with the first move which has decent acceptance, but that the acceptance drops dramatically afterwards.

8 Riemann Manifold Hamiltonian Monte Carlo [9][10]

The general principal behind Riemann Manifold Hamiltonian Monte Carlo (RMHMC) is that instead of simulating the motion of the particle in Euclidean space, we are simulating its motion on a Riemannian manifold, where the local metric is dependent upon the probability distribution. The most important impact is that the kinetic term becomes

$$\frac{1}{2}p^T M^{-1} p$$

where M is the metric, becoming a mass matrix. In the original paper, they motivate a couple of different possible metrics, but the version I am using is in the second paper, which is a modified Hessian matrix so that it is always positive definite and straightforward to calculate. The general goal of the metric is to untangle correlations between variables and prevent rapid changes in certain dimensions from destabilizing convergence.

In the most general form, the Hamiltonian is

$$H(\theta, p) = -\mathcal{L} + \frac{1}{2}\log\{(2\pi)^D |M|\} + \frac{1}{2}p^T M^{-1} p$$

Again, M^{-1} is treated as a mass matrix for the kinetic term. If M is constant, then the second term is a constant. If it is not, we can still write it as

$$\log\{(2\pi)^{D/2}|M|^{1/2}\} = \log\{(2\pi)^{D/2}\} + \log\{|M|^{1/2}\}$$

Clearly, the first term is always a constant, so it does not contribute to the overall dynamics. The second term is the important one, as it will have impacts if the determinant of the metric tensor changes.

The specific metric I used was based on the Hessian metric. The only difference is that the matrix was modified so that all the eigenvalues were positive. The specific function used to modify the eigenvalues is

$$\lambda' = \lambda \coth(\lambda\alpha)$$

The effect is that of an absolute value function where the area between 0 and $1/\alpha$ has been smoothed out. The method of modification is to write

$$M = Q\lambda Q^T,$$

modify the eigenvalues λ, and then reconstruct the matrix.

The other key difference with the RMHMC sampler is that the starting momentum must be sampled from a distribution with covariance matrix M. The simplest way to do this is perform a Cholskey decomposition of M into lower and upper triangular matrices and multiply the lower triangular matrix by the other normally distributed samples. This allows the sampler to converge to the desired distribution. A modification of the general method that can be made is to simply use the diagonal of the Hessian. Assuming there are not especially strong correlations between the variables, this will work well and be significantly faster.

9 l2hmc [11]

The general principal with l2hmc is to take normal HMC and modify the leapfrog integrator to improve the performance of the sampler for not ideal distributions.The method is designed to have the following properties:

1. Fast mixing
2. Fast burn-in
3. Mixing across energy levels
4. Mixing between modes

Normal HMC tends to have fast mixing spatially, but does a random walk across energy levels, and can not handle mode mixing. The new integrator must satisfy the following:

1. It must be invertible.
2. The determinant of the Jacobian must be tractable.

This is accomplished by observing the following:

1. Each sub-update only updates a subset of the variables
2. None of the sub-updates depend non-linearly on the updated variables

Each sub-update can be translated and rescaled, as long as the variables used to determine this changes are not the updated variables.

The state space is the same as in normal HMC, except there is a binary direction variable which is either -1 or 1. Additionally, for each leapfrog step, we pick a random binary vector where half the entries are 0, and the other are 1. This is used when updating the position, as it breaks it into two steps where half of the positions can be used to update the other half, preventing a non linear update of the variables.

We define two neural networks, one for the momentum update, and one for the position update. Each predicts a translation, a momentum rescaling, and a gradient or position rescaling. For the +1 direction update, we make the updates here:

$$v' = v \odot \exp(\frac{\epsilon}{2} S_v(\xi_1)) - \frac{\epsilon}{2}(\partial_x U(x) \odot \exp(\epsilon Q_v(\xi_1)) + T_v(\xi_1))$$

$$x' = x_{\bar{m}^t} + m^t \odot [x \odot \exp(\epsilon S_x(\xi_2)) + \epsilon(v' \odot \exp(\epsilon Q_x(\xi_2)) + T_x(\xi_2))]$$

$$x'' = x'_{m^t} + \bar{m}^t \odot [x \odot \exp(\epsilon S_x(\xi_3)) + \epsilon(v' \odot \exp(\epsilon Q_x(\xi_3)) + T_x(\xi_3))]$$

$$v'' = v' \odot \exp(\frac{\epsilon}{2} S_v(\xi_4)) - \frac{\epsilon}{2}(\partial_x U(x'') \odot \exp(\epsilon Q_v(\xi_4)) + T_v(\xi_4))$$

The S, V, and T functions all represent the neural network predictions and the circle with a dot represents element wise multiplication.

All of the contributions to the Jacobian determinant happen from the S terms. They are

$$\exp(\epsilon S)$$

The determinant of the Jacobian multiplies the acceptance probability in the accept-reject step/

The loss function makes use of $\delta(\xi, \xi')$, which is the squared distance between steps. The full loss function is

$$l(\xi, \xi', A(\xi'|\xi)) = \frac{\lambda^2}{\delta(\xi, \xi')A(\xi'|\xi)} - \frac{\delta(\xi, \xi')A(\xi'|\xi)}{\lambda^2}$$

Here, λ is a scale parameter. There is also another term which theoretically helps with burnin. It is the same as the above term, except it starts the leapfrog integrator at a random state. I do not see how the positive terms in these loss functions help and have removed them in my implementation. I added another term which has sometimes helped where I calculate the squared difference between the proposed state and two states ago times the acceptance probability of the new state. Sometimes this has helped spread the samples out.

When setting S, Q, and T to 0, I recover normal HMC as expected. As sometimes NaNs creep into the gradients the code only runs the optimization

step if there are non nan gradients. I've also observed that the neural networks are super sensitive to their initialization values. The default behavior is that S and Q have a tanh applied to them after the network and an optional multiplicative constant applied to them. I also did this for T and made the parameter trainable, and initialized to a small number, to help start the sampler close to traditional HMC. The step size was also initialized to a small trainable value, and the square root was trained to avoid negative step sizes.

Some test cases proposed by the paper where a diagonal Gaussian where one dimension is significantly smaller than the other,a multivariate Gaussian with sd between 0.01 and 100 linearly spaced with respect to their log values, a 2d multivariate Gaussian with an added term in the potential with $\eta = 0.01$

$$\eta \sum_i \cos(x_i/\eta),$$

and a multi modal Gaussian. I've got all of these working except the multimodal case. The only one where I saw a substantial increase in performance compared to HMC was the rough well. In order to get the multi modal case this to working I need some sort of decreasing temperature for the acceptance, and I have not found an optimal schedule yet.

10 Coupling From the Past [12]

This is an interesting method which can be used to exactly sample from a distribution with a finite space of possible states. In this method, instead of simulating a Markov Chain forward, you simulate one backward. Specifically, start at state 0. We don't know what state proceeded this one, so we consider all the possible states which may have, and step a normal Markov Chain forward from this state -1.

Next, we check and see if this map from all possible states maps to only one possible state. If this is the case, this state is the sample, otherwise create another map from state -2 to -1, check the combined map, and repeat until a constant map is obtained. Once this map is constant, it will remain constant, because any further mappings included will simply change the input to the constant map.

The algorithm is guaranteed to produce this constant map, as there is a nonzero probability of going between any two states, so any map has a nonzero chance of being constant. Thus, one eventually will be. According to the paper, this sample is pulled from the stationary distribution because it's probability is not changed from an additional Markov Chain step at the beginning of the chains.

This method is fairly computationally expensive and can be sped up. Suppose the space being sampled from has some sort of partial ordering, and the Markov chain steps "almost surely" respects the ordering such that

$$A \leq B \rightarrow M(A) \leq M(B)$$

. Then the paper claims we need only run the algorithm with starting states 0,1, where these are the minimal and maximal elements of the partially ordered space. Intuitively, this makes sense as the Markov chain should preserve 0 and 1 being minimal and maximal, so by the time they are equal, all other states will also be equal This algorithm has the potential for use in applications such as spin systems.

11 Reversible Jump [13]

This algorithm presents a way of extending MCMC sampling to a problem where there are different subspaces with different dimensionalities. The first example used in the paper is modeling coal mining disasters using multiple Poisson distributions, where the parameters for the distribution change at unspecified times an unspecified number of times. In general, this method is useful for Bayesian model selection. The actual algorithm isn't super interesting, it just uses a modified Hastings accept-reject step for adding or removing a dimension. The actual accept-reject step is fairly complicated, but it's just to ensure detailed balance works out

References

[1] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21, 1953.

[2] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[3] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[4] Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3), 2003.

[5] Jun S. Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[6] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341, 1996.

[7] Radford Neal. Mcmc using hamiltonian dynamics. *Chapman & Hall/CRC Handbooks of Modern Statistical Methods Handbook of Markov Chain Monte Carlo*, 2011.

[8] Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo, 2011.

[9] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[10] Michael Betancourt. A general metric for riemannian manifold hamiltonian monte carlo. *Lecture Notes in Computer Science Geometric Science of Information*, page 327–334, 2013.

[11] Daniel Levy, Matthew D. Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks, 2017.

[12] James Gary Propp and David Bruce Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1-2):223–252, 1996.

[13] Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, Dec 1995.