Report Week 2

Matthijs Neutelings, Dirren van Vlijmen, Olivier Brahma

December 2021

1 Elongated Gaussian

We are interested in sampling from the following distribution:

$$p(x_1, x_2) \propto exp(-E(x_1, x_2));$$
$$E(x) = \frac{1}{2} x' A x;$$
$$x = (x_1, x_2);$$
$$A = \begin{pmatrix} 250.25 & -249.75 \\ -249.75 & 250.25 \end{pmatrix}.$$

To summarise, We know the exact distribution up to the normalisation constant. To sample from p, we will use the Metropolis Hastings algorithm and the Hybrid Monte Carlo algorithm.

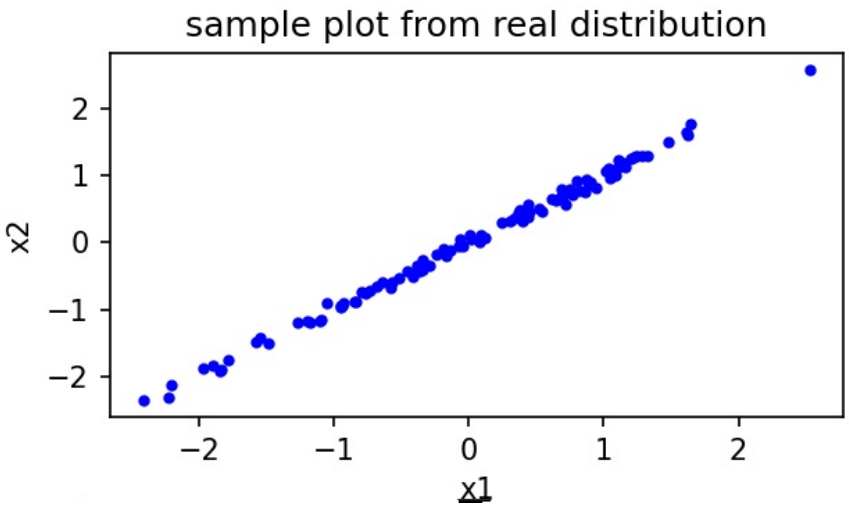

Figure 1: 100 samples drawn from the Elongated Gaussian

1.1 Metropolis Hasting algorithm

To perform Metropolis Hastings sampling, we need to have a sampling density that depends on the current sample: $q(x|x_r)$. For this particular problem, we will consider

$$q(x|x_r, \sigma) \sim \mathcal{N}(x|x_r, \sigma I_2),$$

where I_2 is a 2x2 identity matrix, and *sigma* a hyperparameter for the algorithm. We will use this q to draw our samples. For a candidate sample x' and current sample x_r, the acceptance probability a is defined as

$$a = \frac{p(x') * q(x_r|x')}{p(x_r) * q(x'|x_r)}.$$

However, because we chose q to be a normally distributed, we get that $q(x_r|x') = q(x'|x_r)$ and thus

$$a = \frac{p(x')}{p(x_r)}.$$

We accept a new sample when $a > 1$, or with probability a when $a < 1$. For different sigmas, We will generate 100 samples by applying the above described routine to initial points $(x_i^0)_{0 \leq i \leq 99}$.

1.1.1 Convergence

Before we can start sampling, we need to decide how many steps of the MH algorithm we are applying to get from our starting sample x_i^0 to our final sample x_i^N. The choice of N is important: We want to keep the number of steps minimal for time efficiency, but if we do not use enough steps x_i^N will be too dependent on x_i^0. The choice of σ influences the optimal value for N. For small σ, we will need more steps to get away from x_i^0 but the rejection rate will be low. For large σ, we will need fewer steps to get away from x_i^0 but the rejection rate will be high.

In order to find a suitable relationship between N and σ, we will view our sampling process as a one-dimensional random walk. We note that while our sample space is two-dimensional, there is so much covariance that we get a straight line (fig1). To minimise the distance that our samples need to travel, and thus the number of steps, we choose our initial values to be at the centre of the distribution: $x_i^0 = 0$ for all i. We note that our initial values do not need to be independent, since we use independent samples from q. We see from (fig1) that the distribution spans from approximately $(-2, 2)$ until $(2, -2)$. This means that from the centre we need to be able to bridge a distance of approximately $\sqrt{8}$. When the expected step size is equal to ϵ, we will need $\frac{8}{\epsilon^2}$ steps to have the expected distance travelled be equal to $\sqrt{8}$ (source). Furthermore, we have

$$\epsilon = \mathbb{E}(|x|) = \sqrt{\mathbb{E}(x^2)} = \sqrt{\mathbb{V}(x)} = \sqrt{\sigma}.$$

This brings us the desired relationship between N and σ:

$$N = \frac{8}{\sigma}.$$

An important consideration is that we can only really speak of a (bounded) random walk when there is an equal chance of going in either direction, e.g. a is equal to 0.5 for two points on the line in (fig1), and 0 when the candidate sample is not on the line. This is the case when q is uniformly distributed, and not normally distributed as in our case, where step direction will be biased to areas with higher probability density. We do not think that this will be a problem, because while our points will not cover as much distance, the tail of our distribution is smaller than that of a uniform one and so we only need a small number of points to reach the edge.

1.1.2 Results

Experimental results					
Sigma	N	Acceptance ratio	Mean MCMC steps	Mean	Covariance
0.005	1600	0.57	2788	0.11,0.11	0.83,0.83 0.83,0.82
0.01	800	0.46	1762	0.10,0.11	0.79,0.80 0.80,0.80
0.05	160	0.24	669	0.16,0.17	0.89,0.88 0.88,0.88
0.1	80	0.17	468	0.15,0.14	1.41,1.40 1.40,1.39
0.25	32	0.11	303	-0.05,-0.07	1.22,1.22 1.22,1.22
0.5	16	0.07	225	0.16,0.16	1.00,1.00, 1.00,1.00
1	8	0.04	180	-0.22,-0.21	1.24,1.23 1.23,1.23
2.5	4	0.02	196	-0.04,-0.04	1.17,1.17 1.17,1.18
5	2	0.01	228	-0.11,-0.11	1.06,1.05 1.05,1.05

Table 1: My table

We sampled 100 samples using different values of N and σ, all fitting $N = \frac{8}{\sigma}$ (rounded up). We note that N is the number of *accepted* samples. The number of MCMC steps depends on N and the acceptance ratio. Statistics of our experiment are presented in table 1, and plots of the generated samples in figure 2. We see that for all combinations of σ and N, we get distributions that look like the real one. Furthermore, for every combination, sample means and covariances are close to the real ones. There is some difference, but we believe that this is due to the randomness of the sampling process. We see the acceptance ratio decrease when σ increases, as expected.

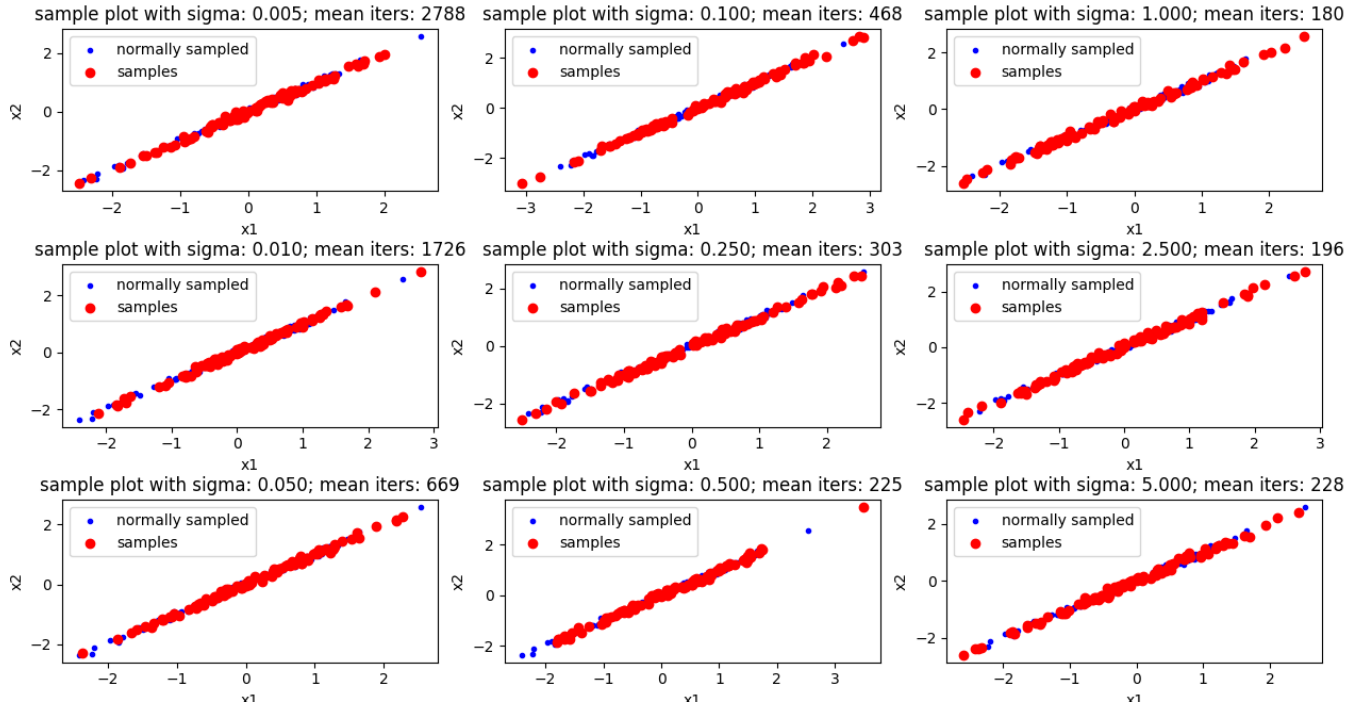

Figure 2: 100 samples created with the MCMC algorithm for different values of sigma. 'Mean iters' is the mean number of iterations to get to the number of accepted samples required for the corresponding sigma value.

To determine what the best combination of σ and N is, we look at the mean number of MCMC steps taken for each of the 100 samples. We see that the combination of (σ, N) = (1,8) gives us the least number of MCMC steps needed.

1.2 HMC algorithm

The second method we use to sample from the Elongated Gaussian is the Hybric Monte Carlo method. This method uses Hamiltonian dynamics to be able to get new samples far away from the old sample. The process is as follows: We start with an initial point. Like with the MCMC algorithm, we have 100 initial points $(x_i^0)_{0\leq i\leq 99}$ all set to 0. We draw an initial momentum value p_i^0 from a standard two-dimensional normal distribution. One step of the HMC algorithm consists of the following: for a pair of (x, p), calculate the Hamiltonian $H(x, p)$:

$$H(x,p) = E(x) + \frac{1}{2}p^2$$

Then, we find a candidate pair (x', p') by applying Hamiltonian dynamics to (x, p):

$$\frac{dp_i}{dt} = -\frac{\delta H}{\delta x_i} = -\nabla E(x)_i$$

$$\frac{dx_i}{dt} = \frac{\delta H}{\delta p_i} = p$$

We move in the direction of the gradient with the following scheme, which we call a leapfrog step:

$$p = p - \epsilon \nabla E(x)/2,$$

$$x = x + \epsilon p,$$

$$p = p - \epsilon \nabla E(x)/2.$$

Here ϵ is the step size and a hyperparameter. We make τ leapfrog steps, τ being another hyperparameter. We then have a candidate pair (x', p'). We accept this pair with probability

$$a = \frac{e^{-H(x',p')}}{e^{-H(x,p)}}.$$

After running this scheme for some number of times N, we take the last value x_i^N as our sample.

The beauty of this method is that the Hamiltonian is invariant under these dynamics. This means that a will be one, except for a small integration error that we have to take into account.

1.2.1 Convergence

To have a relationship between N, τ and ϵ, we use the same formula as for the MH algorithm:

$$N = \frac{8}{(ssd)^2},$$

where ssd is the single-step-distance. In our case, we make τ leapfrog steps of size ϵ, so we set $ssd = \epsilon\tau$. This gives us

$$N = \frac{8}{(\epsilon\tau)^2}$$

1.2.2 Results

We tested the algorithm with different values for τ and ϵ. Unlike in the MC section, we set N to be the total number of algorithm iterations, and not the number of accepted samples. The reasoning is that these numbers should not be that different since the rejection rate will be low. Statistics are reported in table 2, and plots of the generated samples in figure 3.

Unlike with the MH algorithm, the acceptance ratio does not significantly decrease as N decreases. This means that the lowest value for N also gives us the fastest algorithm. From this, we get that the best values for ϵ and τ are 0.03 and 30 respectively (from the values we tested).

Experimental results					
Epsilon	Tau	N	Acceptance ratio	Mean	Covariance
0.01	10	800	0.997	0.08, 0.08	1.09,1.09 1.09, 1.10
0.02	10	200	0.984	-0.03,-0.05	0.86,0.86 0.86,0.86
0.03	10	89	0.982	-0.12,-0.11	1.06,1.06 1.06,1.07
0.01	20	200	0.996	-0.07,-0.07	0.95,0.95 0.95,0.96
0.02	20	50	0.994	-0.15,-0.15	1.08,1.09 1.09,1.09
0.03	20	23	0.97	-0.14,-0.14	1.30,1.29 1.29,1.29
0.01	30	89	0.998	0.02,0.02	0.96,0.94 0.94,0.93
0.02	30	23	0.988	-0.08,-0.08	0.89,0.89 0.89,0.90
0.03	30	10	0.968	-0.02,-0.02	1.19,1.19 1.19,1.19

Table 2: My table

1.3 Comparison between MH and HMC

We ran both the MH and MC algorithms for a number of timeframes, and report the norm of the means (figure 3). From this we can see that the HMC algorithm outperforms the MH algorithm.

2 Perceptron

We use both Metropolis Hastings and Hybrid Monte Carlo sampling to sample from the distribution $p(w|D, \alpha)$ as given in MacKay equations 41.8-41.10. For both sampling methods, we reproduce figure 41.5 from MacKay in figures 4 and 5.

We inspect the influence of the choice of hyperparameters on both the rejection rate and the burn-in time. For the Metropolis Hastings algorithm, we vary the variance of the proposal distribution σ. We measure both burn-in time and rejection rate. Burn-in time is calculated as the first time that the error M(w) becomes less than the overall mean of M(w). The results are presented in figure 6. We see that for greater σ values, the burn-in time decreases, but the rejection rate increases. This means we have to make a consideration between these two factors. From the figure, a well rounded choice for σ seems to be $\sigma = 0.1$.

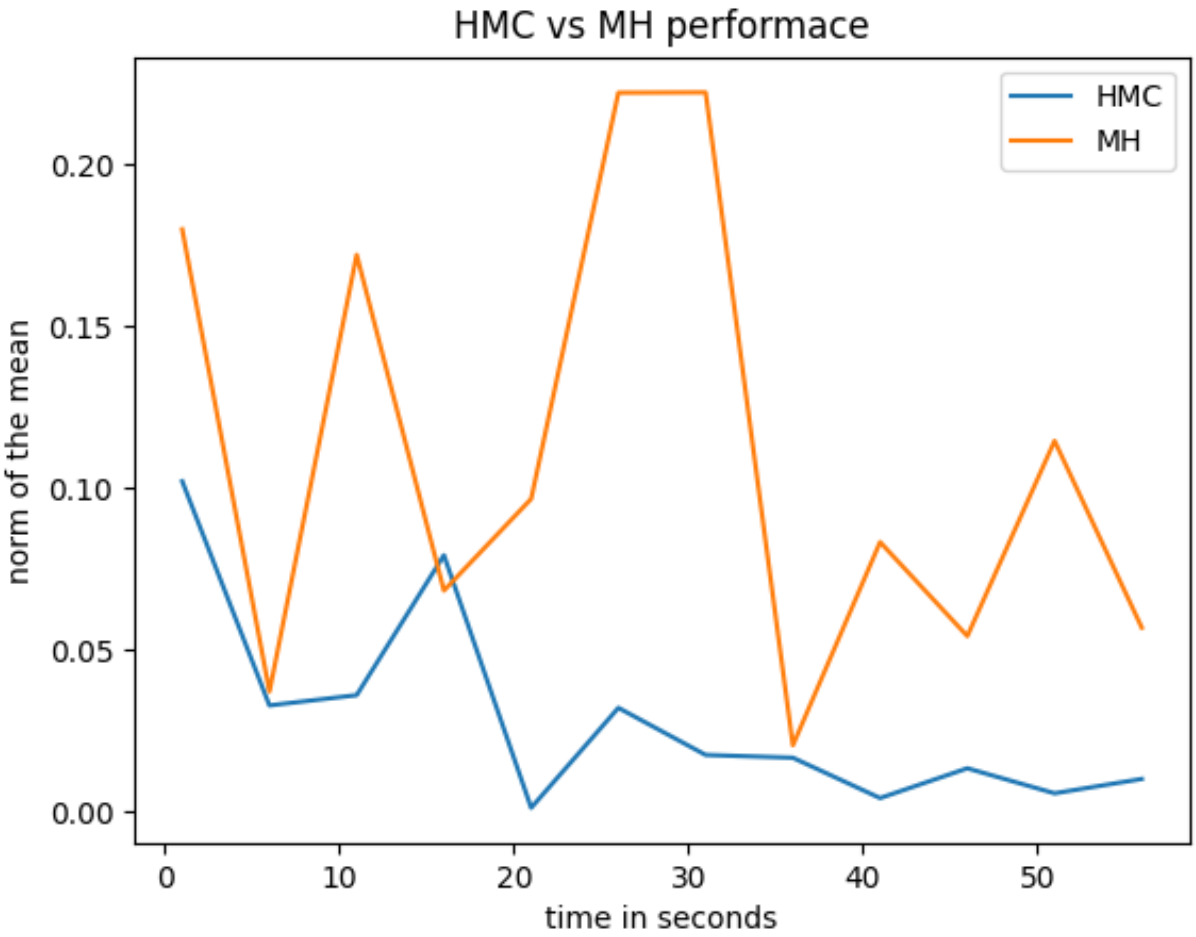

Figure 3: For HMC and MH, we measure the computation time vs the error. Since the true mean is 0, we report the norm of the mean sample values for both algorithms. We see that in general, HMC outperforms MH.

For Hybrid Monte Carlo, we vary both ϵ and τ, but keeping their product equal to 1. In other words, a small step size leads to many leapfrog steps and vice versa. The results are shown in figure 7. We see that a smaller epsilon value leads to both a lower rejection rate and a lower burn-in time. An exception however is for the largest epsilon value, where the burn-in time suddenly gets very low. Upon further inspection, the mean error value $M(w)$ was about 10 times as high as for the other ϵ values. We think that this invalidates the option for this value, and would say that the two smallest ϵ values are optimal.

3 summary

We have tested both the Metropolis Hastings and the Hybrid Monte Carlo sampling algorithm on multiple problems. We have shown what the relation is between different hyperparameters, and what their optimal choice is. Furthermore, we have compared the performance of both algorithms and found that overall, the HMC algorithm is faster than the MH algorithm.

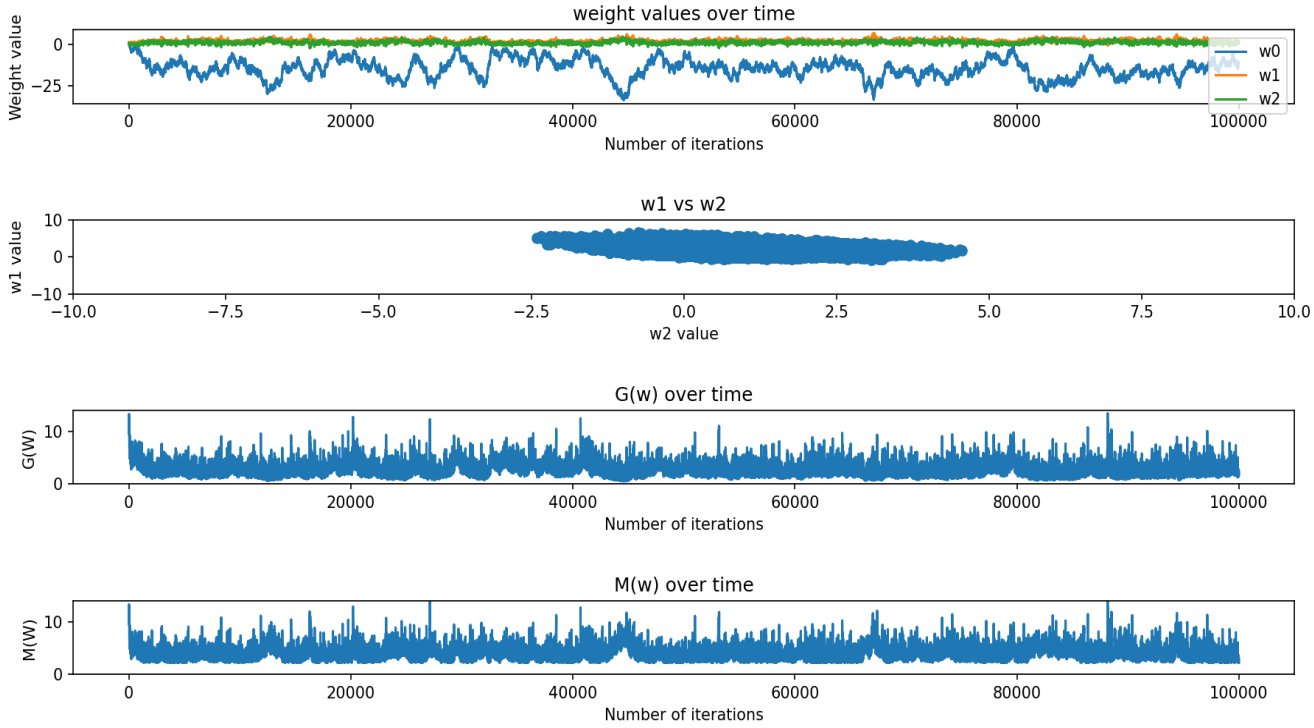

Figure 4: Recreation of Figure 41.5 from MacKay for the Metropolis Hasting algorithm, where the proposal distribution is normal with variance sigma=0.1

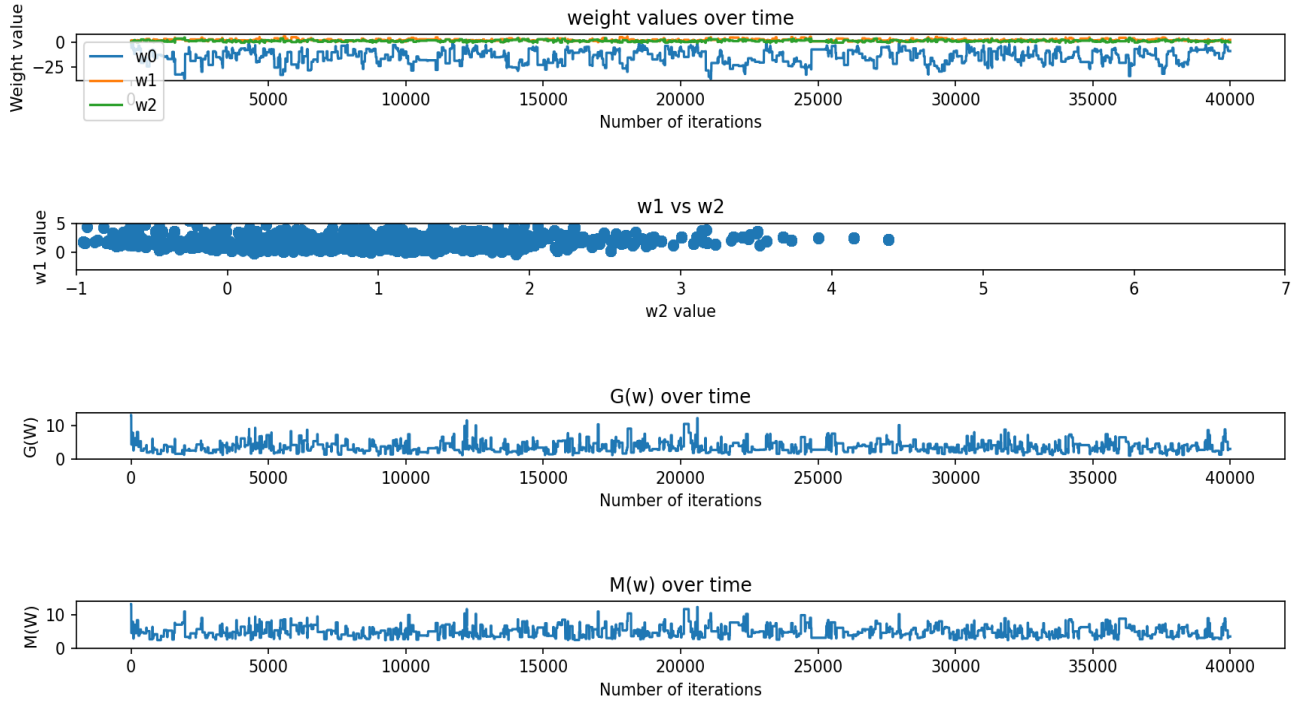

Figure 5: Recreation of Figure 41.5 from MacKay for the Hybrid Monte Carlo algorithm, with epsilon = sqrt(0.01) and tau = 1/epsilon

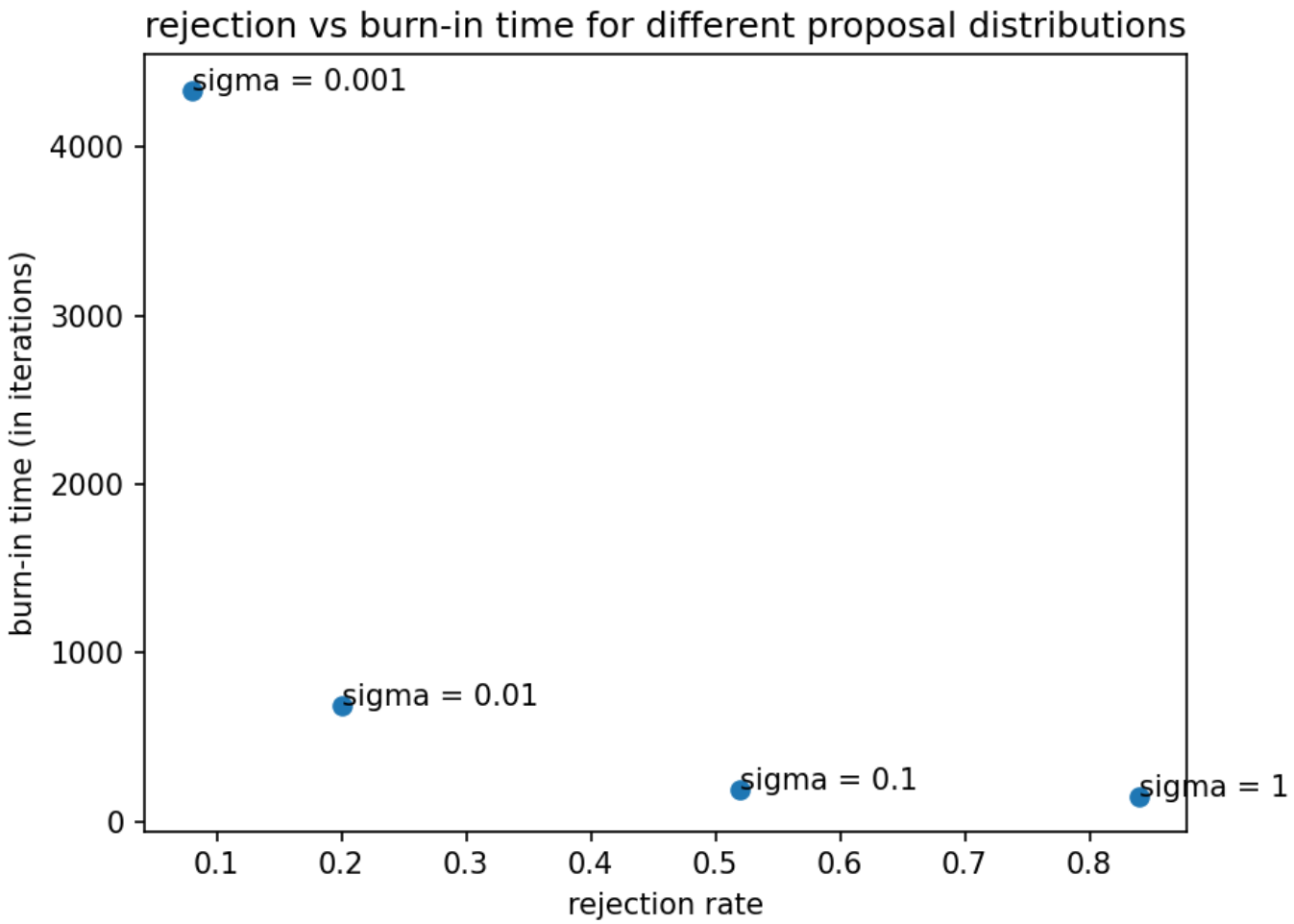

Figure 6: burn-in time vs rejection rate for different sigma values. We see that a lower rejection rate comes at the cost of a longer burn-in time

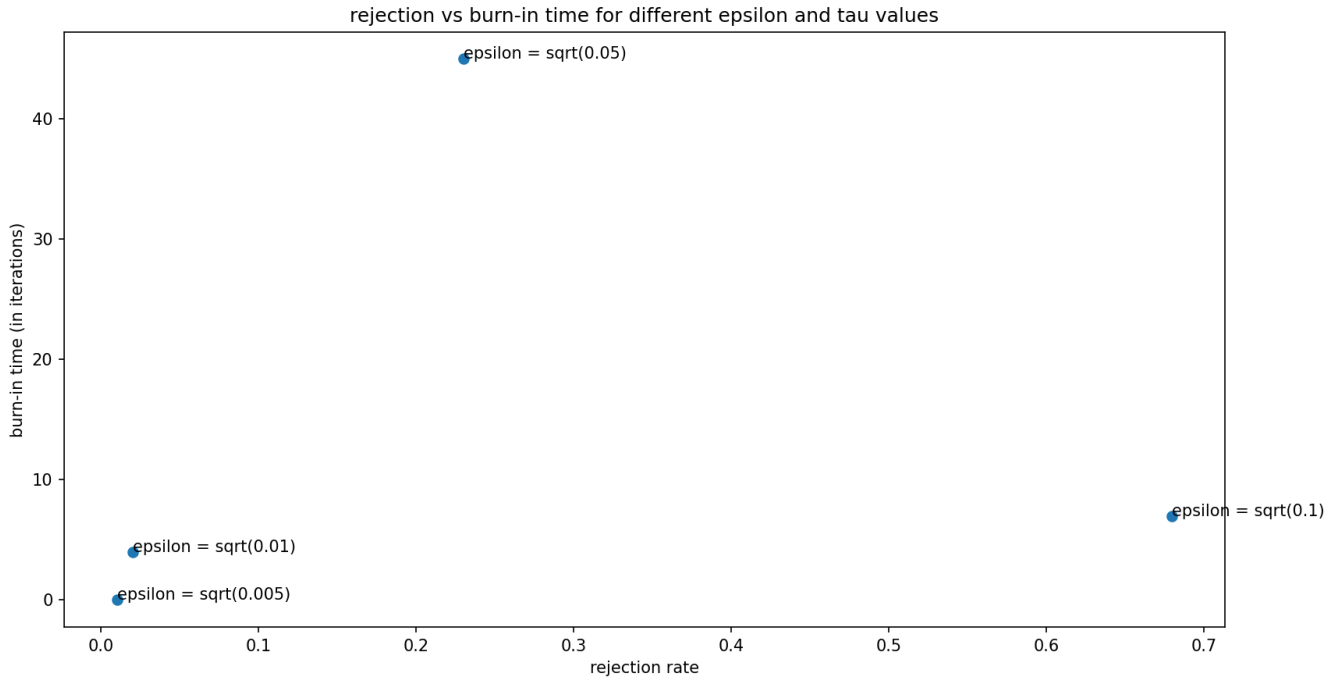

Figure 7: burn-in time vs rejection rate for different epsilon values, with tau = 1/epsilon. We see that a lower epsilon leads to both a lower rejection rate and a shorter burn-in time