Advanced Machine Learning - Week 2

Benedetta Felici, Daria Mihaila, Jasper Pieterse

November 2023

1 MK 29.15

In this exercise, we apply Gibbs sampling to infer the mean μ and precision parameter $\beta = \frac{1}{\sigma^2}$ of a one dimensional Gaussian distribution.

We recall that the joint distribution of points sampled from a one dimensional gaussian is

$$p(D|\mu, \beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{\frac{2\pi}{\beta}}} exp(\frac{-(x_i - \mu)^2 \beta}{2}) \tag{1}$$

Furthermore, we assume a broad prior normal distribution with zero mean and large variance for μ:

$$p(\mu) \sim N(\mu|o, s^2) \tag{2}$$

and a flat prior for β:

$$p(\beta) \sim \frac{1}{\beta} \tag{3}$$

The Gibbs sampling scheme involves a multi step procedure to iteratively infer parameters. At step t+1, we first sample $\mu_{t+1} \sim p(\mu_{t+1}|\beta_t)$, where p is the posterior distribution of μ_{t+1}. Using Bayes theorem, we can compute the posterior:

$$p(\mu_{t+1}|\beta_t, D) = p(D|\beta_t, \mu_{t+1}) p(\mu_{t+1}) \tag{4}$$

where $p(\mu_{t+1})$ is the prior distribution in (2) and $p(D|\beta_t, \mu_{t+1})$ the likelihood in (1). Since we have a Gaussian prior, the posterior distribution will also be a Gaussian, with parameters μ_{post} and β_{post}. Applying the results (2.141) and (2.142) in Bishop [1], we obtain that:

$$\mu_{\text{post}} = \frac{N s^2}{N s^2 + \frac{1}{\beta_t}} \bar{x} \tag{5}$$

$$\sigma_{\text{post}} = \frac{1}{s^2} + N \beta_t \tag{6}$$

where N is the number of points and $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$ the sample mean. Since we assume s^2 to be very large, we obtain the following approximation for the parameters of the posterior:

$$\mu_{\text{post}} \approx \bar{x} \tag{7}$$

$$\beta_{\text{post}} \approx N \beta_t \tag{8}$$

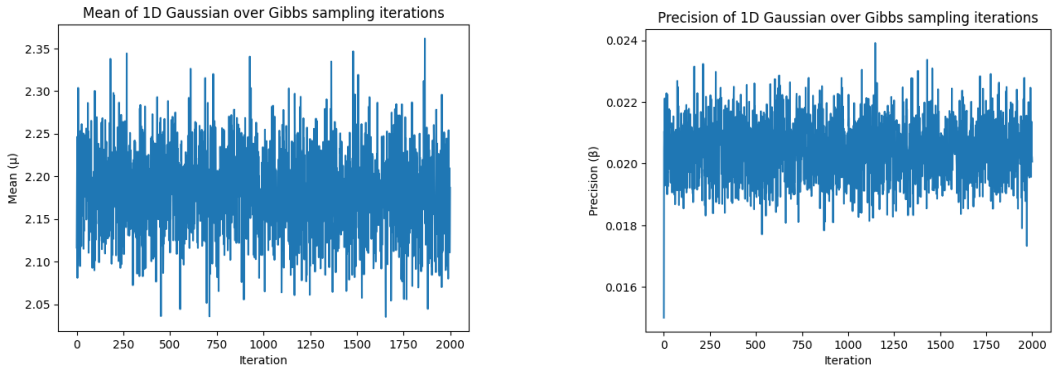
Next, we can sample β_{t+1} from the posterior distribution $p(\beta_{t+1}|\mu_{t+1})$, which for Bayes theorem is equal to:

$$\begin{aligned} p(\beta_{t+1}|\mu_{t+1}) &= p(D|\beta_{t+1}, \mu_{t+1})p(\beta_{t+1}) \\ &\propto \beta^{\frac{N}{2}-1} \exp\left(-\frac{1}{2}\beta \sum_{i=1}^{N}(x_i - \mu_{t+1})^2\right) \\ &\sim \Gamma(\alpha_{\text{post}}, b_{\text{post}}) \end{aligned} \tag{9}$$

Thus, the posterior distribution of β_{t+1} is a Gamma with parameters $\alpha_{post} = \frac{N}{2}$ and $b_{post} = -\frac{1}{2}\sum_{i=1}^{N}(x_i - \mu_{t+1})^2$. As such, we can sample β_{t+1} from the obtained posterior and this concludes a round of the Gibbs sampling scheme.

Next, we implemented the procedure in Python and ran simulations. More specifically, we first sampled $N = 1000$ points from a one dimensional Gaussian with $\mu = 2$ and $\beta = 0.2$.

We started the simulations with $\beta_0 = 0.15$ and ran the Gibbs sampling procedure for 2000 iterations. We report an overview of the values sampled throughout the process in Figure 1. Overall, we see that the parameters sampled during the Gibbs sampling procedure are very close to the original ones of the Gaussian.



(a) Mean *(b) Precision*

Figure 1: Simulation results of the Gibbs sampling scheme applied to the inference of the parameters μ (left side) and β (right side) of a one dimensional gaussian. The illustrations depict the evolution of the sampled parameters over the iterations of the Gibbs sampling procedure

2 Report MCMC Exercise 1: Elongated Gaussian

In this report we will look at the problem of sampling from a elongated Gaussian. We will look at two methods for sampling this Gaussian: Metropolis-Hastings (MH) and Hamiltonian Monte Carlo (HMC). We will effectively answer the following research questions

- What the most effective way is to sample this strongly correlated Gaussian?
- How do MH sampling and HMC sampling compare?
- How does the MH step size depend on the acceptance ratio?
- How does the HMC step size and leap frog step size depend on the acceptance ratio?

2.1 Theory

2.1.1 Metropolis-Hasting Sampling

The Metropolis-Hasting algorithm works by having a proposal sample distribution $q(x)$ that depends on the value of the current sample x^t. It can be shown that this algorithm satifies detailed balance, and thus that the Markov chain converges to the desired distribution [2].

The energy of this system is given by the negative log of $p(x)$:

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \tag{10}$$

The acceptance criterion is given by:

$$\alpha = \min\left(1, \frac{p(x')q(x|x')}{p(x)q(x'|x)}\right) \tag{11}$$

This acceptance criterion ensures that the new sample x' is accepted with a probability proportional to the ratio of the target distribution probability at x' and x. The term $\frac{p(x')q(x|x')}{p(x)q(x'|x)}$ represents the balance between the probability of moving from x to x' and the probability of moving back from x' to x.

2.1.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) improves upon the Metropolis-Hasting algorithm by using gradient information into the sampling process. This method reduces the random walk behavior and allows for more efficient exploration of the distribution, especially in cases where the distribution is highly correlated, as with our elongated Gaussian.

HMC introduces a Hamiltonian $H(\mathbf{x}, \mathbf{p}) = U(\mathbf{x}) + K(\mathbf{p})$, where $U(\mathbf{x})$ is the potential energy (given by equation (10)) and $K(\mathbf{p})$ is the kinetic energy (given by the momentum). The algorithm alternates between updating the position $\mathbf{x}$ and the momentum $\mathbf{p}$ by using simulated Hamiltonian dynamics. The momentum variable determines where $\mathbf{x}$ state goes, and the gradient of $E(\vec{x})$ determines how momentum $\mathbf{p}$ changes, following these equations:

$$\dot{\vec{x}} = \vec{p} \tag{12}$$

$$\dot{\vec{p}} = \frac{\partial E(\vec{x})}{\partial \vec{x}} \tag{13}$$

The acceptance criterion in HMC is based on the change in the Hamiltonian:

$$\alpha = \min\left(1, \exp\left(-\Delta H\right)\right) \tag{14}$$

where $\Delta H = H(\mathbf{x}', \mathbf{p}') - H(\mathbf{x}, \mathbf{p})$.

2.2 Methods

The bivariate Gaussian distribution $p(x)$ we will be sampling from is defined as:

$$p(\mathbf{x}) \propto e^{\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}}$$

with input vector $\mathbf{x}^T = (x_1, x_2)$ and precision matrix $\mathbf{A} = \begin{bmatrix} 250.25 & -249.75 \\ -249.75 & 250.25 \end{bmatrix}$.

2.2.1 Metropolis-Hasting Sampling

For the Metropolis-Hastings sampling, we use a specific proposal distribution to propose new states. The proposal distribution is defined as $q(\mathbf{x}) = \mathbf{x} + \sigma * \mathbf{u}$, where σ is the standard deviation and $\mathbf{u}$ is a vector containing two independent sample s drawn from a standard normal distribution.

The energy function, $E(\mathbf{s}) = -\frac{1}{2}\mathbf{s}^\top\mathbf{\Sigma}\mathbf{s}$, is used to calculate the acceptance ratio α for each proposed state:

$$\alpha = \min\left(1, \exp(E_{proposal}(s) - E_{current}(s))\right) \tag{15}$$

We then did experiments with varying values of σ to observe the impact on the acceptance ratio and the effectiveness of the sampler in covering the entire distribution.

2.2.2 Hamiltonian Monte Carlo

We implemented the Hamiltonian Monte Carlo method using $U(\mathbf{x})$ as given by same as the energy function (10)), and $K(\mathbf{p}) = \sum \mathbf{p}^2/2$. The initial p-value was sampled from $\mathcal{N}(0, 1)$.

The gradient of the energy function, $\nabla E(\mathbf{s}, \mathbf{\Sigma}) = \mathbf{\Sigma}\mathbf{s}$, is used to update the momentum $\mathbf{p}$ and position $\mathbf{s}$ variables through a leapfrog integration scheme. This scheme involves alternating half-step updates to the momentum and full-step updates to the position. The number of leapfrog steps L determines how many intermediate steps are taken in each iteration of HMC. The acceptance criterion in HMC is given using equation (14).

We experimented with various combinations of L and ϵ to determine their impact on the acceptance ratio and the efficiency of the sampler.

2.2.3 Evaluation Metrics

In our analysis, we computed mean and covariance of the generated samples and compared these with the true mean and covariance of the bivariate Gaussian distribution. For each run, we initialized the sampler at initial state $s_0 = (0.0, 0.0)$. The Python code for this project can be found here.

We chose to focus on the average and variability as our main measures of accuracy because the assignment required it and they are relatively straightforward to calculate. However, comparing these measures for multiple hyperparameters is not straightforward. Should we give more importance to one measure over the other? We decided to give equal weight to both, calculating the total error as $E_{tot} = E_{mean} + E_{variance}$. A solution to this could be to use the Kullback-Leibler divergence to measure the error, giving a single scalar for which each hyperparameter combination can be compared with.

To visually compare the samplers, we created a scatter plot of the samples, overlaying it with contour lines representing the true distribution. The scatter plot helps with visualizing some properties of the sampler However, the scatter plot alone can be somewhat misleading as it doesn't clearly show the central concentration of the sample points. We also computed the acceptance ratio and the computation time for these plots.

We also created graphs showing how the errors in the average and variability changed with the number of samples N for different combinations of hyperparameters. For these results, we averaged over 10 full runs to reduce the effect of randomness. Rather than plotting the computation time, we chose to plot the errors against the sample size. This assumes that the time it takes to compute the results increases linearly with the number of samples. We did this because we found some inconsistencies in how long computations took, especially with smaller sample sizes, leading to some irregular-looking graphs (these graphs are not included here, but can be found in the code).

2.3 Results

2.3.1 Metropolis-Hasting

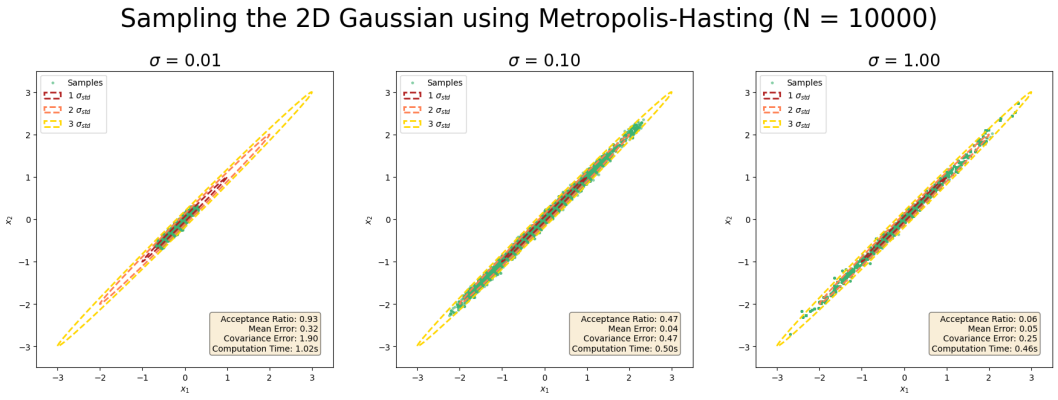


Figure 2: Visualization of Metropolis-Hasting sampling for a 2D elongated Gaussian distribution with different standard deviations. The plots display the sampled points (orange dots) and confidence ellipses representing 1, 2, and 3 standard deviations from the mean (dotted lines in red, green, and yellow, respectively). From left to right, the standard deviation σ increases from 0.01 to 1.0, with corresponding acceptance ratios (a_{ratio}) of 0.93, 0.46, and 0.05. As σ increases, the acceptance ratio decreases, indicating a less efficient sampling process. The tight clustering of samples along the distribution's elongation is clear, particularly for smaller σ.

Figure 2 displays the results of Metropolis-Hastings sampling for 10.000 data points from a two-dimensional Gaussian distribution. We show the outcomes for three different standard deviation values: $\sigma = 0.01$, $\sigma = 0.1$, and $\sigma = 1.0$. Alongside the sampled points, we have drawn ellipses to represent one, two, and three standard deviations from the distribution's mean.

At $\sigma = 0.01$, the points are tightly clustered and don't spread out enough, covering only a small part of the expected area. However, this setting does yield a high acceptance rate of 93%. With $\sigma = 0.1$, the points spread out more, suggesting a better exploration of the space, yet still not enough to match the expected spread of a true Gaussian. The acceptance rate also decreases to 46%. With $\sigma = 1.0$, the points cover a wider area, which arguably gives the optimal results, but the acceptance rate is very low at 5%, indicating that our sampling is mostly moving around without accepting new points. Additionally, there is still an under-representation of the distribution's tails.

In Figure 3 we show the acceptance rate as as a function of σ, with the x-axis plotted on a logarithmic scale The plot shows a clear sigmoidal relationship between σ and the acceptance ratio.

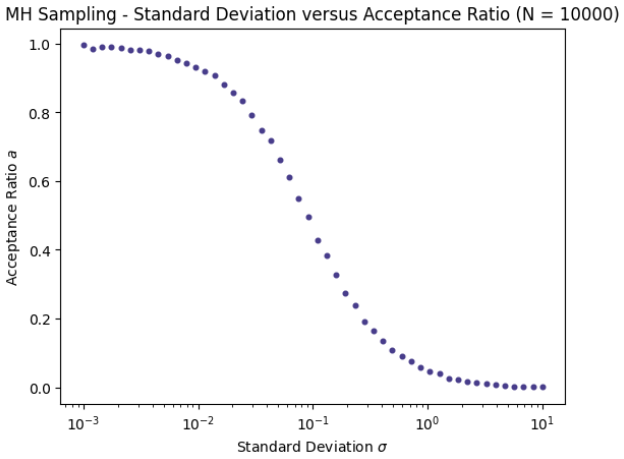


Figure 3: Acceptance ratio (a) as a function of the standard deviation (σ), using a logarithmic scale for sigma on the x-axis. Each dot is computed using 10.000 Metropolis-Hastings samples. The curve displays a sigmoidal relationship, with the acceptance ratio remaining high for smaller values of sigma and sharply decreasing as σ grows.

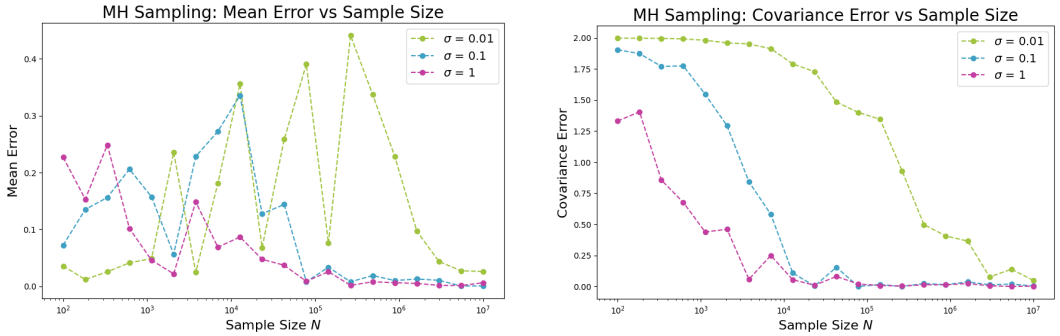


Figure 4: Mean error (left) and covariance error (right) as a function of sample size N plotted for the values $\sigma = 0.01, 0, 1, 1$. The results were averaged over 10 separate runs. For $\sigma = 0.01$ there appears to be significant random-walk behaviour, that only converges in the limit of $N = 10^7$ samples. For $\sigma = 0.1$ and $\sigma = 1$, convergence happens around the same time at $N = 10^5$

In Figure 4 we show the convergence of the mean error (left) and covariance (right) over sample size N. For $\sigma = 0.01$ we see clear oscillatory behaviour due to random-walk behaviour. This appears to converge only in the very large sample size limit at around 10 million samples, which is not useful for any practical application. Interestingly, we see that $\sigma = 1$ starts with a lower error than $\sigma = 0.1$, but they both converge to about the same error in the same time at about 10.000 samples.

From the figure we observe that convergence in mean is significantly slower than convergence in the covariance. This might be because the mean is a first-order moment, while covariance is a second-order moment. The latter can stabilize faster under certain conditions. Furthermore, the mean appears to be more sensitive to sample size. Even when we averaged it over 100 different runs, it was still fluctuating randomly.

For small sample sizes below 10.000 samples, the best sampler depends on what you want, more samples with a larger error or less samples with a smaller error. However, if our goal is to obtain as much samples with a converged error, then $\sigma = 0.1$ clearly wins.

2.3.2 Hamiltonian Monte Carlo

In Figure 5, we present nine subplots that showcase the outcomes of the Hamiltonian Monte Carlo (HMC) sampling method applied to a two-dimensional Gaussian distribution, each varying by step size (ϵ) and number of leapfrog steps (L). The plots provide a visual comparison of how different combinations of ϵ and L influence the sampler's ability to cover the distribution. The optimal results (lowest combined error) was found for the values $\epsilon = 0.05$ and $L = 5$.

The plots illustrate the trade-off between exploration and accuracy in HMC sampling. A higher number of leapfrog steps does not necessarily result in a proportional increase in the acceptance ratio, especially as the step size increases. It becomes evident that fine-tuning ϵ and L is essential for achieving a balance that maximizes both the coverage of the distribution and the efficiency of the sampling process. We see that acceptance rate between 60% to 80% appear to give the best results for this problem. From the scatter plots that HMC is able to better cover the tail of the distribution.

In Figure 6 (left), the acceptance ratio as a function of the step size (ϵ) is shown. We observe that for a broad range of ϵ values, the acceptance rate remains exceptionally high, frequently approaching 100%. However, the plot also reveals several pronounced dips which can be attributed to the stochastic nature inherent in the sampling process. These outliers are not indicative of systematic errors but rather the probabilistic variations expected from the algorithm.

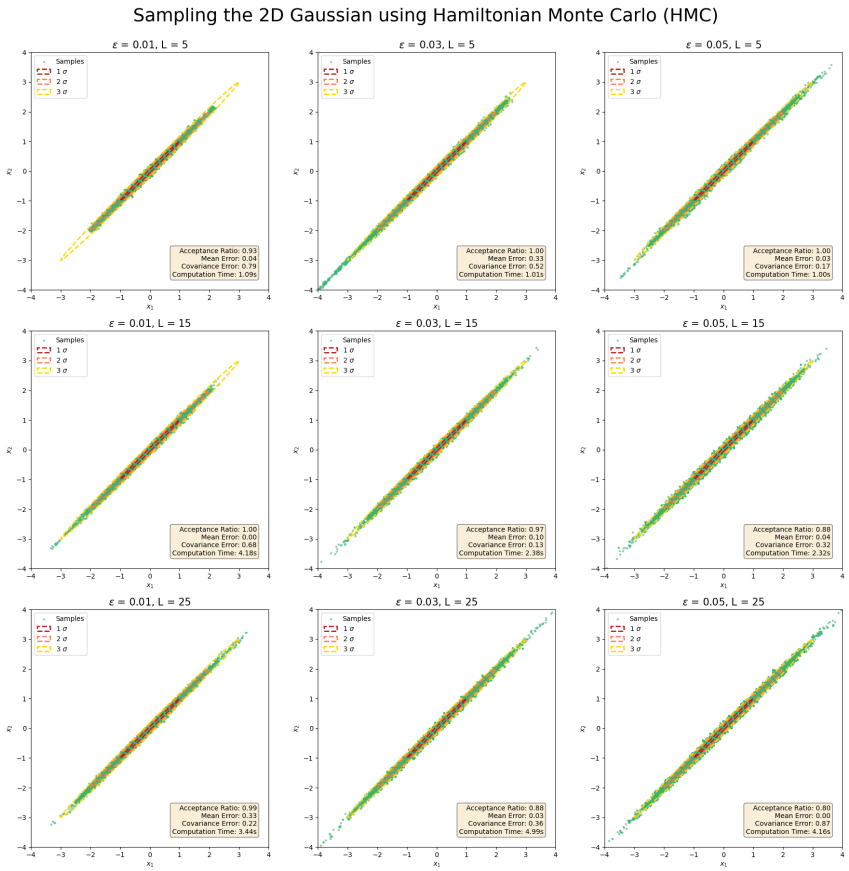


Figure 5: A series of plots illustrating the performance of Hamiltonian Monte Carlo (HMC) sampling on a 2D Gaussian distribution. Each subplot represents a combination of different step sizes (ϵ) and leapfrog steps (L), with the acceptance ratio (a_{ratio}) indicated for each scenario. The sample points are shown along with ellipses corresponding to 1, 2, and 3 standard deviations from the mean.

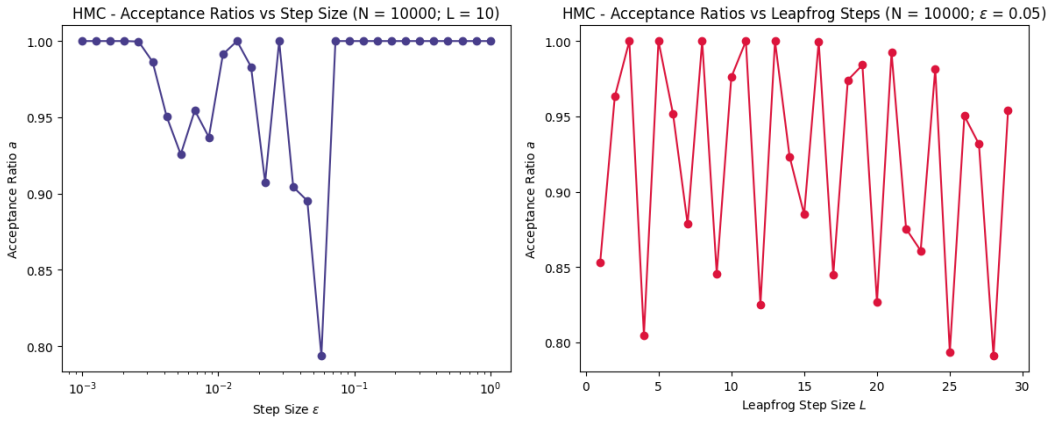


Figure 6: Acceptance ratios of Hamiltonian Monte Carlo sampling under varying parameters. On the left, acceptance ratios are plotted against the step size ϵ on a logarithmic scale, showing high consistency with a few notable dips due to stochastic variations within the sampling process. The right plot examines the acceptance ratios against the number of leapfrog steps (L), displaying an oscillatory pattern. This illustrates the complex dynamics of the HMC algorithm.

In Figure 6 (right), the acceptance ratio is plotted against the number of leapfrog steps (L). The oscillatory pattern observed appears to not stem from any anomalies in our implementation but is likely a manifestation of the stochsatic dynamics of the HMC algorithm. The leapfrog integrator's discretization of Hamiltonian dynamics can lead to complex interactions between the step size and the system's energy landscape.

As L increases, we traverse more of the phase space, which can result in varying acceptance rates depending on how the proposed points align with the target distribution. In some instances, an increase in L can cause the proposed leapfrog trajectory to overshoot or 'double back', affecting the acceptance probability.

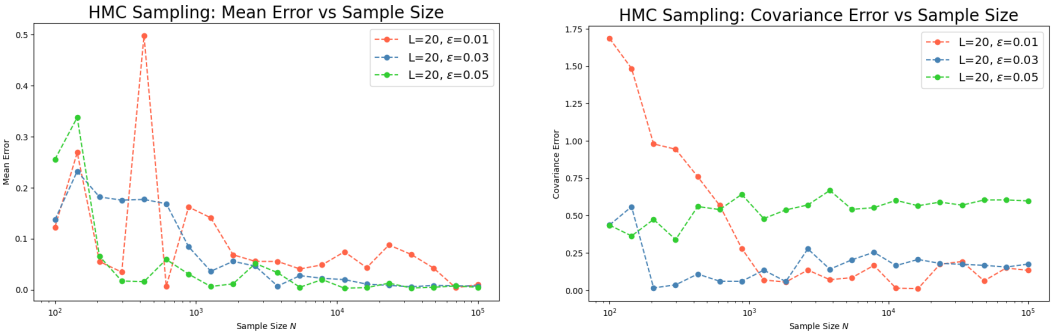


Figure 7: Mean error (left) and covariance error (right) as a function of sample size N plotted for varying learning rates $\eta = [0.01, 0.05, 0.1]$ with constant $L = 20$. The results were averaged over 10 separate runs. The mean and variance error . The learning rate appears to control the mean convergence rate and the level to which the variance can converge to, with higher learning rates corresponding to quicker convergence and larger covariance error.

In Figure 7 we show the convergence of the mean error (left) and covariance (right) over sample size N for varying learning rates $\eta = [0.01, 0.05, 0.1]$ constant $L = 20$. The results were averaged over 10 separate runs. We observe relatively quick convergence in both the mean and variance. Convergence appears to be at least an order of magnitude faster then for the MH method (10^4 samples instead of 10^5 samples for optimal parameters). Moreover,

the mean error has a lot less fluctuations, indicating a decrease in random-walk behaviour. For the mean error, higher learning rate corresponds to faster convergence. However, for the covariance error, too high a learning rate will lead to a larger covariance error in convergence.

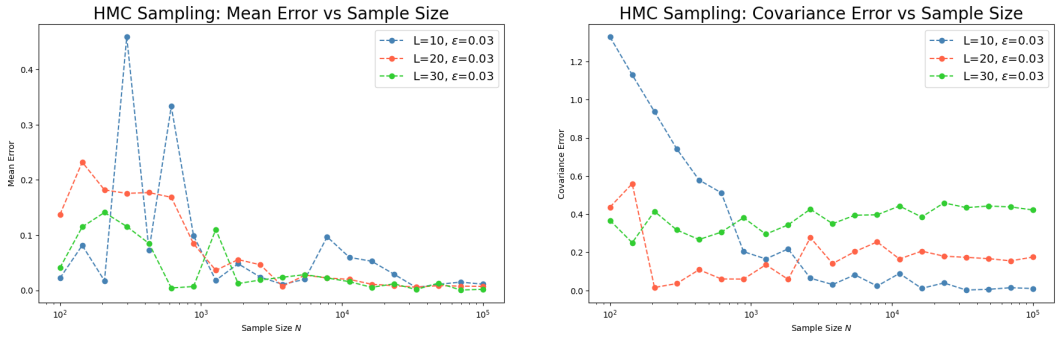


Figure 8: Mean error (left) and covariance error (right) as a function of sample size N plotted for varying leapfrog steps $L = [10, 20, 30]$ with constant $\eta = 0.03$. The results were averaged over 10 separate runs. The leapfrog step size appears to control speed of convergence in the mean, with larger values corresponding to faster convergence. Larger leapfrog step sizes also result in higher covariance error.

In Figure 7 we show the convergence of the mean error (left) and covariance (right) over sample size N for varying leapfrog steps $L = [10, 20, 30]$ with constant $\eta = 0.03$. The results were averaged over 10 separate runs. Interestingly, the leapfrog step size appears to control these errors very similar to how the learning rate does. The leapfrog step size appears to control speed of convergence in the mean, with larger values corresponding to faster convergence. Larger leapfrog step sizes also result in higher covariance error.

2.4 Conclusion

This type of Gaussian has a narrow, elongated shape along a diagonal axis in the variable space. Traditional sampling methods can struggle with such distributions due to getting stuck in narrow regions and failing to explore the entire distribution effectively. The correlation between variables means that a sample that is representative in one dimension may not be representative in the other.

Under most circumstances, the HMC method performs better than the MH method. This is reflected in the faster convergences of errors and the higher acceptance rates. The HMC method is however slightly harder to implement, has longer computation times (mostly dependening on the amount of leapfrog steps) and does not have straightforward hyperparameter tuning relations. It might be better to employ a random search to find the optimal parameters, rather then a grid search.

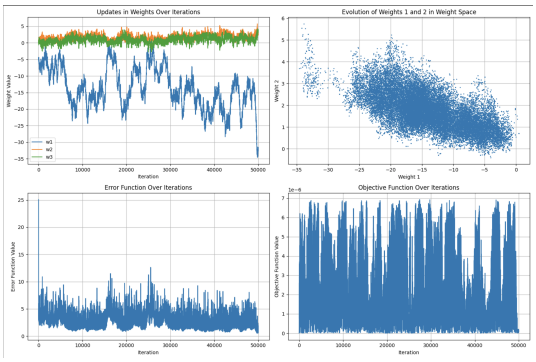


Figure 9: Figure showing the evolution of the Metropolis-Hastings sampling over 50000 iterations. Upper left: Evolution of $w1, w2, w3$ over the iterations. Upper right: evolution of weights $w1$ in $w2$ weight space. Lower left: error function $G(w)$ as a function of number of iterations. Lower right: objective function $M(x)$ as a function of number of iterations.

3 Report MCMC Exercise 2: Bayesian inference for Perceptron learning with MCMC

Very similar to the previous exercise, we will use the Metropolis-Hasting Sampling and Hamiltonian Monte Carlo, then comparing the two. We will sample from the posterior of the learning problem and use the data given in the two files: x.ext and t.ext. The distribution follows MacKay Eqs. 41.8-10 [2]:

$$P(w|D,\alpha) = \frac{P(D|w)P(w|\alpha)}{P(D|\alpha)} = \frac{e^{-G(w)}e^{-\alpha E_W(w)}/Z_W(\alpha)}{P(D|\alpha)} = \frac{1}{Z_M}\exp(-M(w)) \tag{16}$$

We reproduce plots similar to fig. 41.5 and estimate the burn in time that is required before the sampler reaches the equilibrium distribution.

We then investigate the acceptance ratio for both methods and try to optimize this by varying the proposal distribution, the step size in HMC and the number of leap frog steps , evidencing the differences via plots shown below.

3.1 Metropolis Hastings Sampling

Following the theory described in subsection 2.1.1, from the sample distribution $q(x)$ and accept based on the acceptance criterion from Eq. 10, updating the to a new sample. We are here replicating figure 41.5 from MacKay's book, obtaining the following results evidenced in Figure 9: From this figure, we can observe that only the w1 is significantly changing over the 50000 iterations, however, without any clear convergence and stabilisation. This can be linked to the error function plot as well, which has no clear convergence either to very low values (stays above at least >1 at all times. The same behaviour can be also seen in the model figure from the textbook. The lack of very clear convergence in the error graph leads to choosing a burner time cut after a high number of iterations. Choosing the burner period to be after 30000 iterations, the average error is 2.2016, as opposed to an average over all iterations of 2.6370. This does indicate to somewhat of a convergence, but also is hinting to the faults of the algorithm. I would propose for a more thorough analysis; a more comprehensive test might be necessary. But I understood the task at hand is be 'estimate' the burn time, thus judged by eye looking at the error function.

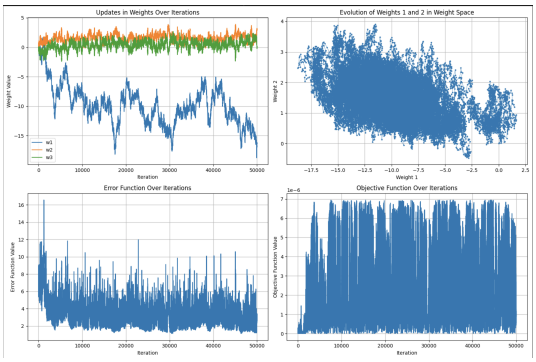


Figure 10: Figure showing the evolution of the Hamiltonian Monte Carlo sampling over 50000 iterations. Upper left: Evolution of $w1, w2, w3$ over the iterations. Upper right: evolution of weights $w1$ in $w2$ weight space. Lower left: error function $G(w)$ as a function of number of iterations. Lower right: objective function $M(x)$ as a function of number of iterations.

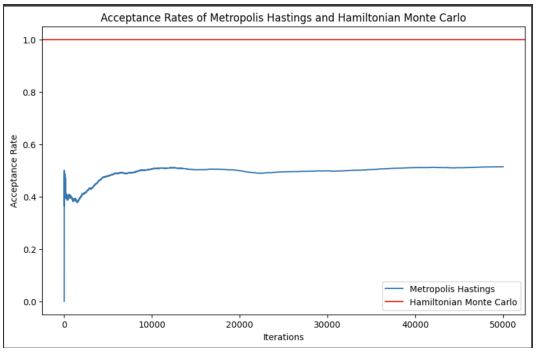


Figure 11: Acceptance Ratios of of Metropolis Hastings and Hamiltonian Monte Carlo

3.2 Hamiltonian Monte Carlo Sampling

We now implement the HMC sampling method to solve the same problem. This method improves the MH one by using the gradient information as well for updating to new samples, as described in section 2.1.2. We show our results in Figure 10, using the same model of figure as for the Metropolis Hastings sampling.

The distributions and graphs are very similar to the HM ones, however, we can notice that the update of the w1 over the 50000 iterations does look more clear, possibly converging more distinctly after the 50000 iterations, reaching stability. The same can be hypothesised also about the error function. However, the average of the error function after the slicing at the burner time = 45000 iterations is 2.4570, compared to an overall average of 2.9562 over all iterations. Thus, we notice that the error of the HMC sampling is higher than that of MH. This could perhaps be due to the nature of the data and the distribution as well. Maybe with more data or more iterations, the difference between these two sampling techniques would be more distinctive or there could be a more comprehensive analysis made of the performance of the two by using other tests as well.

3.3 Analysis

By investigating the acceptance ratio of the two methods, it is immediately clear that HMC is accepting all samples - showing that the chosen parameters are a good fit of the model - and the MH converges after about 10000 iterations to a 50% acceptance rate.

For a more in depth analysis of the acceptance ratio of HMC, we vary the proposal distribu-

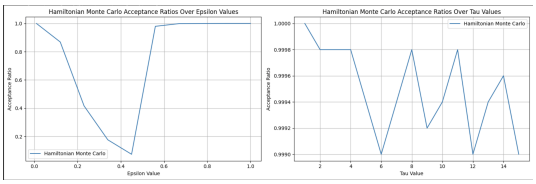


Figure 12: Investigating the acceptance ratios of the HMC sampling method with left: varying step sizes, right: varying number of leap frog steps size.

tion, the step size ϵ and the number of leap frog steps τ and plot the changes in acceptance ratio as shown in Figure 12. From this we can observe that perhaps a value of ϵ that is closer to 1 is more suited for optimising the Hamiltonian Monte Carlo, and the leap frog step τ value should be very low (as close to 0 as possible while still efficient computing).

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738.

[2] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.