Mathematics 2 - Part 3 - MCMC: Homework

Luka Žontar

Introduction

In Bayesian statistics we rely on being able to integrate the posterior distribution of parameters of our statistical model. This requires us to be able to integrate higher dimensional integrals, which is not trivial. Monte Carlo provides a simple but scalable solution for this problem by approximating integrals using random sampling. Markov Chain Monte Carlo (MCMC) methods copes with the problem of being able to extract samples from a desired distribution.

In this homework,we were tasked to implement 3 different MCMC methods and test them on 4 different scenarios. First, we will take a quick look at the implemented methods. In the following sections, we look at each scenario and check the differences between the 3 methods we implemented. For each scenario and algorithm we generate 1000 samples and run 5 chains.

To be able to reproduce the results, we first set the seed. Source code is available at https://github.com/lukazontar/DS-FRI-Math2-MCMC.

Table 1. Scenario 1: Table of hyperparameters.

Rejection sampling	Metropolis Hastings	Hamiltonian Monte Carlo
- **g**=bivariate standard normal - **M**=1.1	- **Cov**=I	- **L**=27 - **epsilon**=0.6

MCMC methods

For each of the methods, we are trying to generate **m** samples from the target density **p** or proportional density **f**.

Rejection sampling (RS)

Rejection sampling generates sample by proposing an envelope distribution from which we know how to efficiently sample from and rejecting the samples that are unlikely to be in target distribution. Thus, we only require to be able to evaluate the probability of a possible sample to belong to target distribution.

Parameters:

- **g** - proposal density.
- **M** - positive constant that is used to transform the proposal density vertically.

Metropolis-Hastings (MH)

MH does not assume detailed balance and introduces α as probability of accepting a sample from proposal distribution. With α, we adapt transition function to reach detailed balance by rejecting transitions in a smart way.

Parameters:

- x_0 - starting state ($\in S$). Starting state is $(0,0)$ by default.
- **k** - transition function. As per instructions we use multivariate normal as proposal function with 0 mean.
- **Cov** - covariance matrix - parameter to multivariate normal.

Hamiltonian Monte (HMC)

HMC is based on physics-inspired approach, proposing the next state utilizing the gradient of the target distribution. Next state depends on the current position and momentum of target density gradient.

Parameters:

- q_0 - starting state ($\in S$). Starting state is $(0,0)$ by default.
- ε - step size.
- **L** - number of leapfrog steps.
- **M** - diagonal mass matrix with diagonal elements m_i. By default, this is identity matrix.

Scenario 1: Bivariate standard normal

This scenario is trivial, because the target distribution is a distribution from which we know how to sample. In Table 1, we explain the hyperparameters that were used in the calculations. There was not much tuning needed here, since the example is more trivial.

For rejection sampling, it was obvious to use the bivariate standard normal to sample from, since we know how to sample from that density and it is also the target density. In

	Rejection sampling			Metropolis Hastings			Hamiltonian Monte Carlo		
	ESS	ESS/s	Estimate RMSE	ESS	ESS/s	Estimate RMSE	ESS	ESS/s	Estimate RMSE
X1	5000.00	1657.05	0.79 ± 0.01	919.28	1428.07	0.78 ± 0.01	31052.03	73546.21	0.78 ± 0.02
X2	5000.00	1657.05	0.78 ± 0.01	960.71	1492.44	0.79 ± 0.02	17493.10	41485.82	0.78 ± 0.02

Table 2. Scenario 1 - Table - Standard diagnostics.

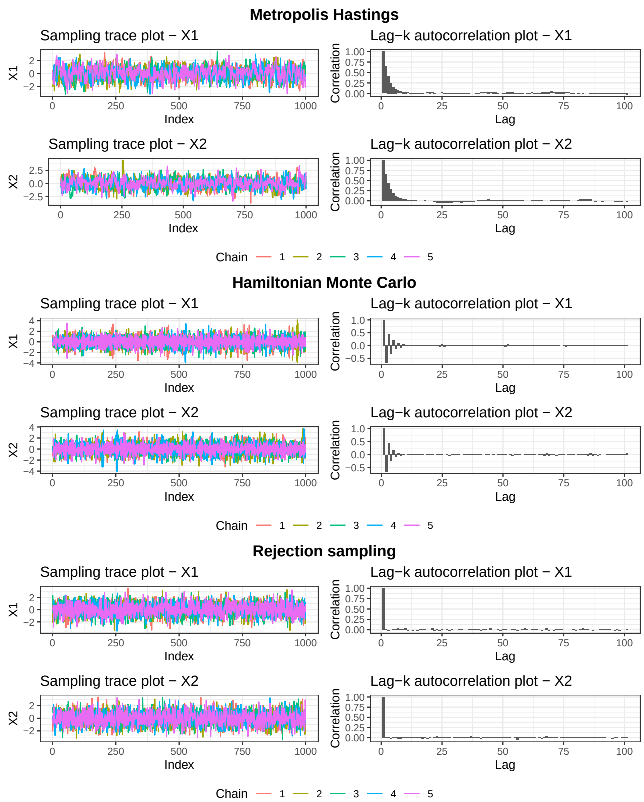

Figure 1. Scenario 1 - trace and lag-k autocorrelation plots .

Hamiltonian Monte Carlo, we proceeded with parameters as were defined in lectures and for covariance matrix in Metropolis Hastings, we simply used the identity matrix.

For each of the scenarios and algorithms we go through standard diagnostics of the sampling by checking traceplots, lag-k autocorrelation plots, ESS and ESS per second metrics. Note that for all scenarios and algorithms, we try to find problematic behaviour by evaluating standard MCMC diagnostics, however we must be aware that not finding any problematic behaviour with these standard methods does not mean that everything is OK with the generated Markov chain.

For Scenario 1, the standard diagnostics in Figure 1 and Table 1 show lag-k autocorrelation quickly declining for all the algorithms. For RS, no autocorrelation is expected, which can also be seen in the figure. Traceplots looks as they should - similar to ”hairy caterpillars”.

In Table 2 we see that RS achieves ESS equal to the number of samples that we generate. Rejection sampling is efficient due to the fact that the proposal function envelops the target density well, which makes sense since proposal distribution is equal to target distribution in this case.

HMC and MH algorithms start with a bit higher lag-k autocorrelation for small k. This could be resolved by including the thinning process by randomly excluding the generated samples. HMC achieves a very high ESS, which might be a conseuqnce of negative lag-k autocorrelations that can be seen in Figure 1.

Regarding the means comparison, we used RMSE with standard error to show the difference between the ground truth and the estimate. All three algorithms provide comparable results and none of the algorithms really stands out. On the other hand, the absolute RMSE is quite low, which shows the benefits of knowing how to sample from the target distribution and the potential of MCMC algorithms on a simpler case.

Scenario 2: The shape of the banana function

The second scenario focuses on the banana shaped function. The function and function transforms that are needed in HMC algorithm are taken from the showcase scripts from lectures.

The hyperparameters used in this scenario are shown in Table 3, where we see that M and covariance matrix are increased in comparison to the previous example. For RS, this essentially means that we transform the proposal density g and move it high above the target density making the gap between f and g bigger and thus rejecting more samples.

Table 3. Scenario 2: Table of hyperparameters.

Rejection sampling	Metropolis Hastings	Hamiltonian Monte Carlo
- g=bivariate standard normal **- M=**10	**- Cov=**$100 \cdot \mathrm{I}$	**- L=**27 **- epsilon=**0.6

The MH ESS results seem quite poor, which is most likely a consequence of poor proposal transition function since the ESS is low no matter the scenario or hyperparameters. For both MCMC approaches we see defected traceplots for the X2 parameter. Otherwise the standard diagnostic plots look good. The estimate RMSE of both MCMC algorithms is much lower than that of RS.

	Rejection sampling			Metropolis Hastings			Hamiltonian Monte Carlo		
	ESS	ESS/s	Estimate RMSE	ESS	ESS/s	Estimate RMSE	ESS	ESS/s	Estimate RMSE
X1	5000.00	2807.94	1.28 ± 0.01	430.47	292.99	0.65 ± 0.01	10583.58	2379.99	0.63 ± 0.00
X2	5000.00	2807.94	0.72 ± 0.01	561.26	382.01	0.14 ± 0.00	13543.50	3045.61	0.14 ± 0.00

Table 4. Scenario 2 - Table - Standard diagnostics.

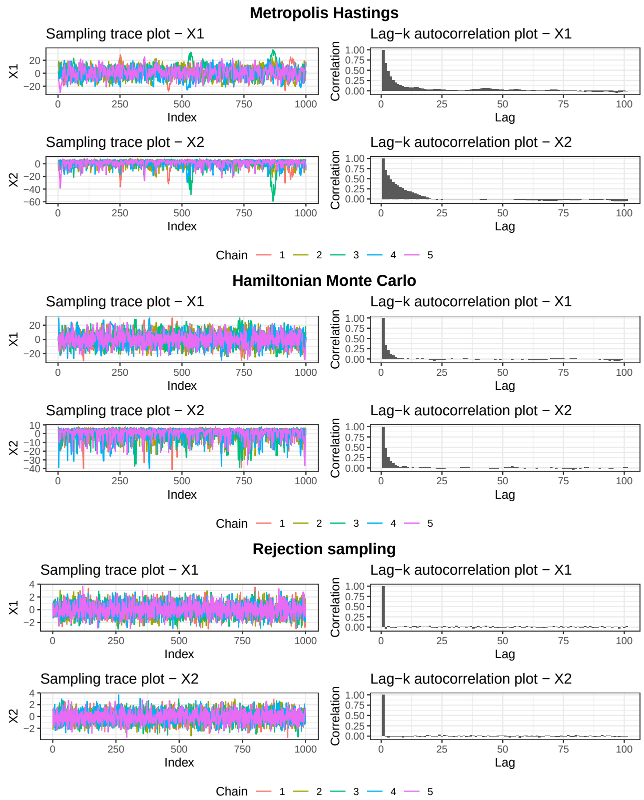

Figure 2. Scenario 2 - trace and lag-k autocorrelation plots .

Scenario 3: The shape of the logistic regression likelihood

In the this scenario we aim to sample from the logistic regression likelihood of a given dataset. In this scenario we take the easier approach by just including two of the variables. The ground truth values from the whole dataset are 2.00, -0.84.

As for hyperparameters, we lowered epsilon, thus making smaller steps in HMC. This shows in high lag-k autocorrelation in Figure 3. As can be seen in Figure 3, the MCMC algorithms produce Markov chains with huge variation in the beginning stages of the algorithm. This can be improved by including the burn-in process, thus discarding the warmup samples. Judging by the Figure 3, we would probably need to discard the first 50 samples. Also, we can see that for HMC we once again have negative lag-k autocorrelations which might be the reason behind the high ESS metric for HMC.

If we look at the estimate errors, we see that HMC and MH once again provide better results than the RS algorithm.

Table 5. Scenario 3: Table of hyperparameters.

Rejection sampling	Metropolis Hastings	Hamiltonian Monte Carlo
- g=bivariate standard normal **- M=**10	**- Cov=**$0.01 \cdot \mathrm{I}$	**- L=**27 **- epsilon=**0.01

However, once again RS algorithm provides samples that do not have problems with autocorrelation or picking the starting state of the Mrkov chain.

Low ESS generated by MH algorithm is likely a consequence of high positive autocorrelation that can be seen in Figure 3.

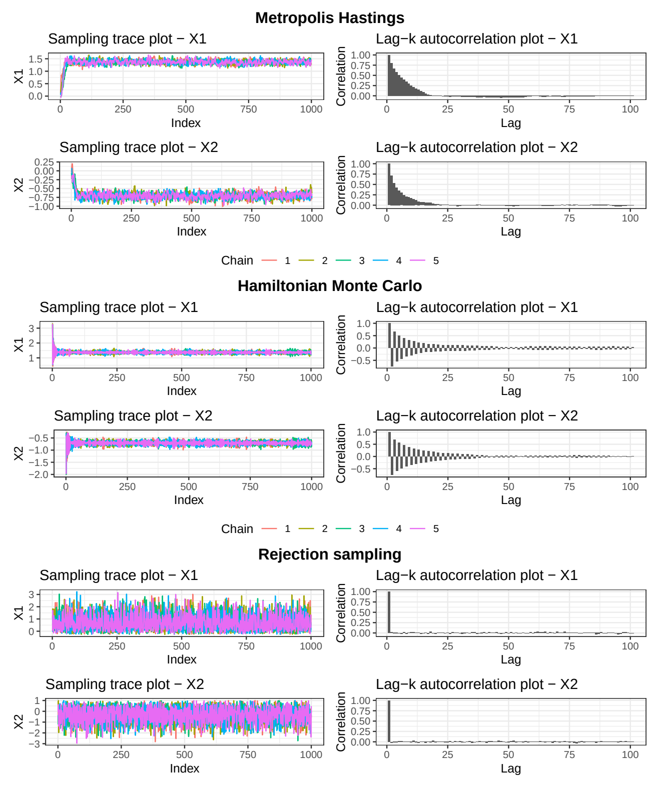

Figure 3. Scenario 3 - trace and lag-k autocorrelation plots .

	Rejection sampling			Metropolis Hastings			Hamiltonian Monte Carlo		
	ESS	ESS/s	Estimate RMSE	ESS	ESS/s	Estimate RMSE	ESS	ESS/s	Estimate RMSE
X1	5000.00	2807.94	1.28 ± 0.01	430.47	292.99	0.65 ± 0.01	10583.58	2379.99	0.63 ± 0.00
X2	5000.00	2807.94	0.72 ± 0.01	561.26	382.01	0.14 ± 0.00	13543.50	3045.61	0.14 ± 0.00

Table 6. Scenario 3 - Table - Standard diagnostics.

Scenario 4: The shape of the logistic regression likelihood - Full dataset

Due to dimensionality issues, we only perform the MH and HMC algorithms. The last scenario takes the same dataset as the third scenario but includes all the variables instead of just first two. Hyperparameters used in Scenario 4 are same as in Scenario 3 but with higher dimension since we have 11 variables.

Because this scenario produces lots of combinations of traceplots and lag-k autocorrelation plots, we decided not to show those in the report but rather direct an interested viewer to the plot image that is versioned on GitHub (see link). There we see that for lots of estimates a problematic behaviour such as high lag-k autocorrelation and varying starting samples of the traceplots occur.

Interestingly, despite the problematic behaviours, we see extremely low estimate RMSE for all of the variables. These results are shown in Table 8.

Table 7. Scenario 4: Table of hyperparameters.

Metropolis Hastings	Hamiltonian Monte Carlo
- Cov=$0.01 \cdot \mathrm{I}$	**- L=**27 **- epsilon=**0.01

Discussion on hyperparameters and most efficient algorithm

To conclude, we discuss how we adjusted hyperparameters for different algorithms. Essentially we used the trial and error approach by starting with some initial parameters and making simple parameter transformations to see how the the diagnostics metric behaved. The main metric on which we tried to optimize the results was the Estimate RMSE. A better approach here would be to make a more organized grid search on the parameter space by minimizing a specific diagnostics metric.

Judging on the results above, we can argue that Hamiltonian Monte Carlo is the most efficient algorithm. This makes sense since it is a state-of-the-art algorithm for generating samples from a target distribution and is most widely used in applications nowadays. However, this statement cannot be said without emphasizing several considerations. High-negative autocorrelation and weird behaviour of traceplots in the beginning are two issues that should be adressed when trying to generate samples using the HMC approach.

	Metropolis Hastings			Hamiltonian Monte Carlo		
	ESS	ESS/s	Estimate RMSE	ESS	ESS/s	Estimate RMSE
X1	157.20	25.59	0.16 ± 0.01	5000.00	320.14	0.13 ± 0.00
X2	208.90	34.01	0.11 ± 0.01	9950.74	637.13	0.10 ± 0.00
X3	296.48	48.27	0.15 ± 0.01	14442.93	924.75	0.14 ± 0.00
X4	427.42	69.59	0.10 ± 0.00	7912.91	506.65	0.08 ± 0.00
X5	902.78	146.98	0.08 ± 0.00	183760.20	11765.79	0.08 ± 0.00
X6	229.08	37.30	0.22 ± 0.01	7039.23	450.71	0.20 ± 0.00
X7	360.55	58.70	0.16 ± 0.01	8538.53	546.70	0.15 ± 0.00
X8	408.10	66.44	0.10 ± 0.00	33783.36	2163.08	0.08 ± 0.00
X9	466.73	75.99	0.11 ± 0.01	11118.10	711.87	0.10 ± 0.00
X10	376.14	61.24	0.09 ± 0.00	17708.80	1133.86	0.08 ± 0.00
X11	317.12	51.63	0.09 ± 0.00	18450.02	1181.32	0.08 ± 0.00

Table 8. Scenario 4 - Table - Standard diagnostics.