Homework 5
Mathematics 2

Domen Mohorčič

May 30, 2022

1 Implementations

We implemented three sampling algorithms: Metropolis-Hastings (MH), Hamiltonian Monte Carlo (HMC), and Rejection Sampling (RS) with uniform envelope. The functions are implemented in MetropolisHastings.R, HamiltonianMC.R, and RejectionSampling.R respectively. All return a single chain of m samples, and work for functions of arbitrary input dimensions.

For each algorithm we created 5 chains of length 1000. For each chain we calculated autocorrelation, effective sample size (ESS), and effective sample size per second (ESS/s).

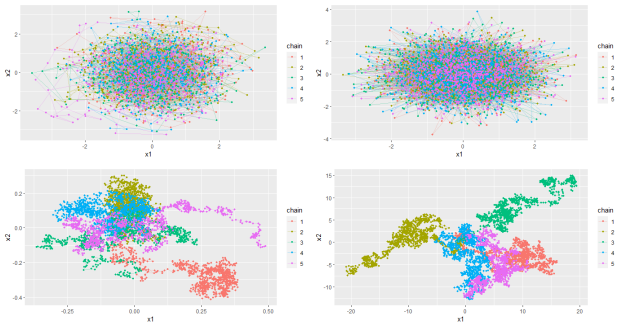

Figure 1: Traceplots for bivariate normal distribution function and each algorithm with 5 chains. Upper left is MH and upper right is RS. Bottom represents HMC, left without tuning and right with tuning.

2 Bivariate standard normal distribution

Bivariate normal distribution function is a probability density function with known mean $(0, 0)$ and covariance matrix (identity matrix). For MH no tuning was required as all parameters were known. For HMC step size was set to 0.001, and number of leapfrog steps was set to 10. For RS we sampled uniformly from area $[-5, 5] \times [-5, 5]$. Optimal M was calculated from bounds and maximum of function. Tuning of HMC was done using grid search with step size over 0.002 to 0.02 with difference of 0.002 and with number of leapfrog steps between 5 and 50 with difference of 5. We were looking for parameter combination with maximal ESS. The best parameters found were 0.012 for step size and 25 for the number of leapfrog steps with mean ESS of 9.31. The resulting plots of each run are shown on Figure 1.

From the Figure we can see that MH and RS sample from bivariate standard normal well, while HMC does not so much as it always wanders off center. Tuned HMC performed the worst as it sampled from regions that are not probable at all. That may be due to one combination of parameters randomly performing the best.

MH algorithm had low autocovariance, but does display some cyclic behaviour when we observe plots carefully. Both tuned and untuned HMC were problematic, as even with window size of 100 they still had high autocovariance. But RS has no problems, since we are sampling from uniform distribution. We can see autocovariance plots for MH and HMC on Figure 2.

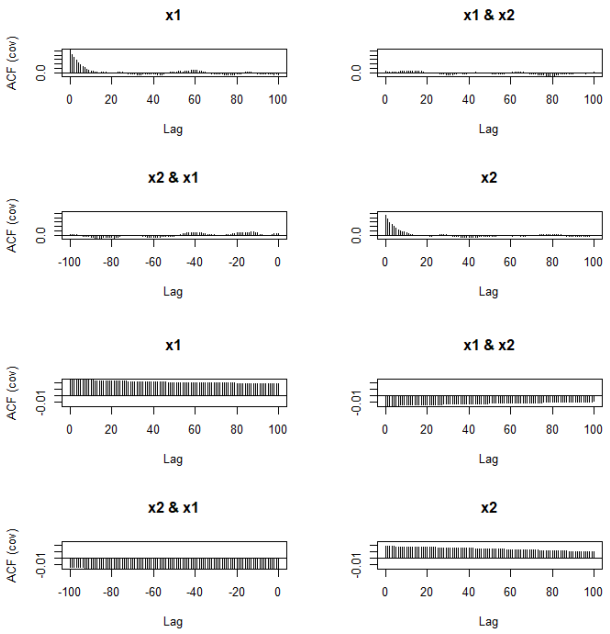

Figure 2: Autocovariance of samples from a random chain. Upper plot is from MH, and lower plot is from HMC. HMC has high autocovariance, while MH is problematic only for shorter intervals.

	ESS		ESS/s	
Method	x1	x2	x1	x2
MH	92.91	99.45	1067	1139
HMC	10.11	6.447	34.52	22.57
tuned HMC	7.266	7.638	13.43	14.03
RS	1000	1000	3721	3721

Table 1: ESS and ESS/s for MH, HMC, and RS on bivariate normal distribution. In terms of samples RS performs best.

On Table 1 we can see ESS and ESS per second for all three methods. We can see that RS has highest ESS, which is expected, as it samples from uniform distribution.

The mean of the samples for MH was $(-0.0226, -0.0768)$, for HMC was $(0.066, 0.052)$, for tuned HMC was $(-0.857, -1.997)$, and for RS was $(0.00529, 0.00119)$. The closest to true mean was RS.

From all of the results RS seems to be the best for sampling from bivariate normal distribution. Even though MH was faster, RS achieved highest possible ESS. Its mean is also the closest to true mean. HMC failed completely, as its ESS was very low, and its sample mean was furthest away from the truth. Otherwise it was realtively easy to pick parameters for HMC, and others did not need tuning.

3 Banana function

The banana function and its gradient was given by an equation. But we had to modify it so it was usable by MH and RS. Original function is convex, and we are supposed to sample the most from its valley, where it has lowest function values. For that we just flipped the sign of the function and translated it upwards to create a function where we want to sample proportional to its highest values. This is not optimal as now the function is limited by where it is positive, but a 100 shift gives large enough positive region to sample from. The true mean (extreme) for banana function is at $(0, 5)$.

For initial results unit mass matrix was used for covariance for MH and HMC methods. HMC used same parameters as in provided example, that is 0.6 step size and 27 leapfrog steps. For RS unit mass matrix was used, and sample area was $[-40, 40] \times [-60, 20]$. M was calculated using maximum function value (now 100) and area size, from which we are sampling. Tuned MH used mass matrix of $\begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix}$. Mass matrix was tuned by hand, and gave visually most promising results. For HMC we again tuned using grid search, but with fewer parameter values. The best parameters found were 0.4 for step size and 50 leapfrog steps, again with unit mass matrix. The resulting traceplots for all methods and their tuned variants are on Figure 3.

We can see that tuned MH performs better than untuned MH as it is less spread around. The same can be said for HMC, but here the effect is less visible.

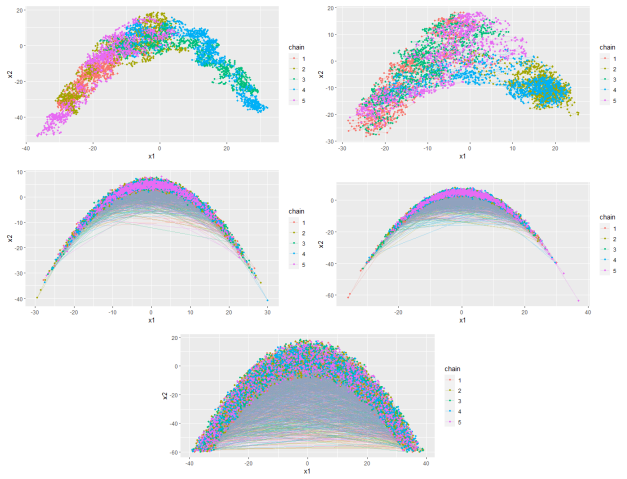

Figure 3: Traceplots for banana function. Upper row is MH, middle is HMC, and bottom is RS. Left side represents no tuning and right represents tuning.

	ESS		ESS/s	
Method	x1	x2	x1	x2
MH	6.002	5.736	59.98	57.52
tuned MH	10.93	8.527	109.9	87.98
HMC	388.2	363.0	3157	2933
tuned HMC	793.5	357.1	5294	2351
RS	993.3	1000	33694	34055

Table 2: ESS and ESS/s for MH, HMC, and RS for banana function.

No tuning was performed on RS as there is nothing to tune, and method performed ok, but its samples cover very large area, which is not ideal.

MH algorithm had surprisingly high autocovariance, whereas HMC had now low covariance. In both methods tuned version had higher covariance than untuned one. We can see autocovariance plots for tuned MH and tuned HMC on Figure 4.

On Table 2 we can see ESS and ESS per second for all three methods. RS again achieved highest ESS and ESS/s values, followed by HMC. MH performed the worse.

The average of the samples from MH was $(-5.405, -9.189)$. Tuned MH achieved average sample of $(-1.112, -5.279)$, which is better than untuned MH. HMC has sample average of $(0.192, 0.0771)$, and tuned HMC has sample average of $(-0.308, 0.126)$. RS had sample average of $(-0.424, -16.146)$.

From the results the best method is HMC. It performed well without function modification, was fast and its ESS value was decent. It also managed to get the closest to true mean. RS also performed good, again having largest ESS values, but it did require function modification. MH also gave samples that covered function good, but its ESS values were very low. Tuning methods was difficult, as at first it did not seem like different parameter values give any better/worse samples.

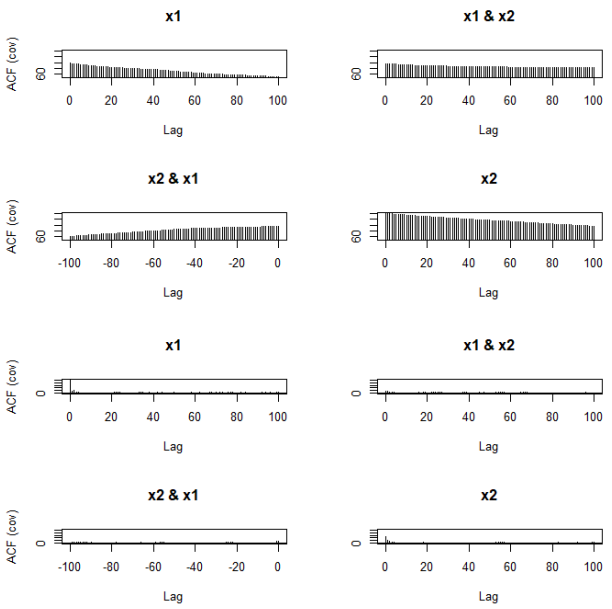

Figure 4: Autocovariance of samples from a random chain. Upper plot is from tuned MH, and lower plot is from tuned HMC. MH has very high autocovariance, while HMC has almost none.

4 Logistic regression likelihood

For final part we are given a dataset and must sample from logistic regression likelihood. For first part only two attributes are considered, and for the second part all atributes are considered. But since likelihood is numerically unstable, we will sample from log-likelihood. The log-likelihood and its gradient are presented in Equation 1. Since values are negative, we add 5000 to the function to make it positive.

$$\begin{aligned} l &= \sum_{i=1}^{N} y_i \log p_i + (1 - y_i) \log (1 - p_i) \\ \frac{dl}{d\beta_j} &= \sum_{i=1}^{N} (y_i - p_i) x_{ij} \end{aligned} \tag{1}$$

4.1 2 columns

For the initial results we used $Q = \begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$ for MH. HMC had step size of 0.0001, 10 leapfrog steps, and $Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. The M for RS was set to 640000. We tuned methods by hand, and for MH the best parameter seemed to be $Q = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}$. HMC was difficult to tune. Lots of times the chains returned only one sample. At the end we were not able to find better samples than with initial run, but the other best seem to be 0.001 step size and only 1 leapfrog step. We did not change covariance matrix as other matrices produced worse results. The resulting traceplots for all methods and their tuned variants are on Figure 5.

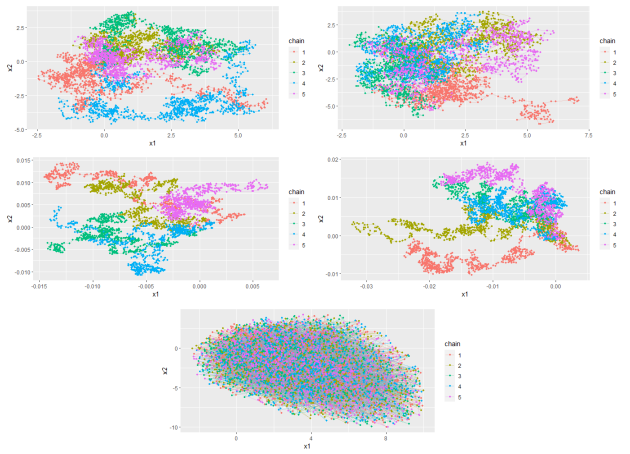

Figure 5: Traceplots for log-likelihood. Upper row is MH, middle is HMC, and bottom is RS. Left side represents no tuning and right represents tuning.

	ESS		ESS/s	
Method	x1	x2	x1	x2
MH	6.083	8.449	19.72	27.38
tuned MH	9.395	7.592	28.47	22.74
HMC	6.191	6.052	4.906	4.793
tuned HMC	5.759	6.961	11.24	13.59
RS	1000	1000	1134	1134

Table 3: ESS and ESS/s for MH, HMC, and RS for log-likelihood.

The best looking ones are MH and RS. HMC seems to not be working as it is very sensitive to parameters. RS has again no problems.

Autocovariance is very low for RS chains. MH has very highly autocorrelated samples, and HMC has somewhat autocorrelated samples. We will not show autocorrelation here, as plots are very much like all previous ones.

On Table 3 we can see ESS and ESS per second for all three methods. Unsurprisingly RS achieved highest ESS and ESS/s values, followed by MH. HMC has the lowers ESS value.

Finally we can look at average sample value. We calculated the expected parameter values using Bayesian logistic regression, and got mean of $(1.366, -0.713)$. MH had average sample value of $(1.383, -0.259)$, and tuned MH had $(1.305, -0.979)$. HMC had $(-0.00499, 0.0028)$ and tuned HMC had $(-0.00858, 0.00481)$. RS had average sample value of $(3.538, -2.225)$. From this we can see that closest to truth is MH method. HMC did not move much from initial point, and RS overestimated mean by around a factor of 3.

From this results we can see that MH performed the best. Even thou it did not have high ESS, its sample average was very close to true mean, compared to other methods. RS takes second place, as its average was at least proportional to true mean. HMC performed the worst, and often returned a single sample as a chain. It was also the most difficult to tune.

Method	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11
Estimate	2.075	-0.888	-0.505	0.726	-0.085	-1.033	-0.830	0.240	-0.605	-0.410	0.485
MH	21.968	-7.510	-5.504	7.105	-0.894	-11.615	-8.520	1.408	-5.679	-3.420	7.092
tuned MH	0.751	-0.842	0.291	-0.969	0.092	-1.773	0.063	0.612	0.214	0.798	-0.762
HMC	1.169	-0.595	-0.093	0.348	-0.105	-0.460	-0.132	0.136	-0.294	-0.003	0.180
tuned HMC	1.867	-0.813	-0.436	0.703	0.042	-1.060	-0.788	0.224	-0.584	-0.352	0.619

Table 4: Average parameter values for MH and HMC. Estimate parameter values are provided for easier comparison.

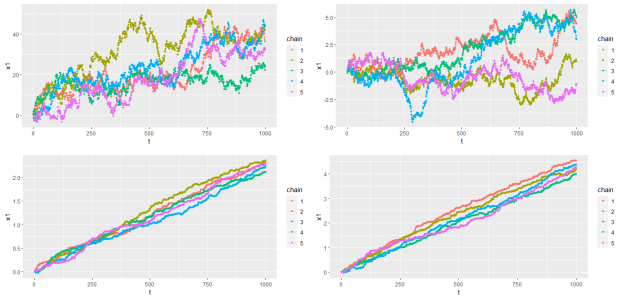

Figure 6: Traceplots for the first parameter (intercept). Upper row is MH, bottom is HMC. Left side represents no tuning and right represents tuning.

Method	ESS	ESS/s
MH	8.249	17.61
tuned MH	6.993	18.10
HMC	6.048	6.399
tuned HMC	6.019	5.943

Table 5: ESS and ESS/s for MH, HMC, and RS for log-likelihood.

4.2 11 columns

The final test was 11 dimensional function, log-likelihood of a logistic regression on the whole dataset. For MH the initial covariance matrix used was identity matrix. The tunning was done by dividing covariance matrix. The best divisor found was 50. For HMC we found a stable parameter set with step size of 0.001 and 4 leapfrog steps. The covariance matrix used was 10 times the identity matrix. Tunning was done by hand, and best parameters found were 0.001 step size, 5 leapfrog steps, and 20 times identity matrix for covariace. Traceplots for the first parameter are on Figure 6

We can immediately see that HMC gives very autocorrelated samples. Samples from MH are much less autocorrelated, but their average is further from the true mean. This may also be because sample size of 1000 is just right for the averages of HMC to be very close to estimated parameter values.

The resulting ESS and ESS/s values can be seen on Table 5. Both methods have low ESS values, but that was expected from their high autocovariances.

In Table 4 we can see average sample values for each method, including parameter values, sampled with Bayesian logistic regression. The closest is tuned HMC, while untuned MH seems to miss the mean by a factor of 10, but that is consistent for all dimensions.

The best performing method was HMC. Tuned HMC got closest to true average, with untuned version being ok as well. MH method overestimated average by a factor of 10. But both methods had low ESS values, and HMC had very high sample autocovariance. Tuning was also not that hard as for only two parameters, which is interesting, as we were expecting the most problems for this task.