Probabilistic Modelling and Reasoning
Solutions for Tutorial 8 — Additional Practice

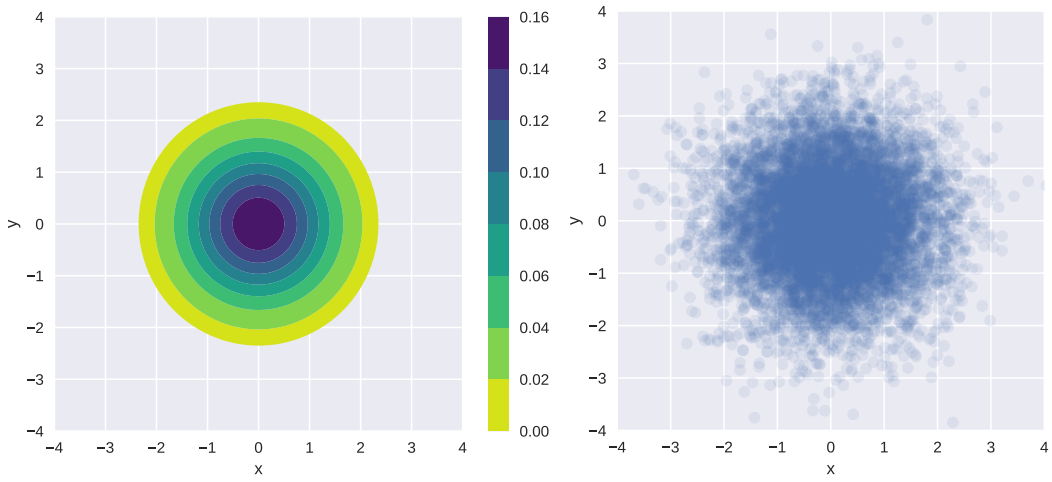
Spring 2020
Michael Gutmann

Introduction

This exercise focuses on sampling and approximate inference by Markov chain Monte Carlo (MCMC). This class of methods can be used to obtain samples from a probability distribution, e.g. a posterior distribution. The samples approximately represent the distribution, as illustrated in Figure 1, and can be used to approximate expectations. In this exercise, you will write your own MCMC algorithm, use it to perform inference, and run diagnostics to detect problems.

We denote the density of a zero mean Gaussian with variance σ^2 by $\mathcal{N}(x; \mu, \sigma^2)$, i.e.

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$



(a) True density

(b) Density represented by $10,000$ samples.

Figure 1: Density and samples from $p(x, y) = \mathcal{N}(x; 0, 1)\mathcal{N}(y; 0, 1)$.

Brief introduction to MCMC and the Metropolis-Hastings algorithm

Consider a vector of d random variables $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ and some observed data $\mathcal{D}$. In many cases, we are interested in computing expectations under the posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$, e.g.

$$\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[g(\boldsymbol{\theta})\right] = \int g(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) \mathrm{d}\boldsymbol{\theta} \tag{2}$$

for some function $g(\boldsymbol{\theta})$. If d is small, e.g. $d \leq 3$, deterministic numerical methods can be used to approximate the integral to high accuracy.[1] But for higher dimensions, these methods are generally not applicable any more. The expectation, however, can be approximated as a sample average if we have samples $\boldsymbol{\theta}^{(i)}$ from $p(\boldsymbol{\theta} \mid \mathcal{D})$:

$$\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[g(\boldsymbol{\theta})\right] \approx \frac{1}{S} \sum_{i=1}^{S} g(\boldsymbol{\theta}^{(i)}) \tag{3}$$

In MCMC methods, the samples $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(S)}$ used in the above approximation are typically not statistically independent.

[1] See e.g. https://en.wikipedia.org/wiki/Numerical_integration.

Metropolis-Hastings is an MCMC algorithm that generates samples from a distribution $p(\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ can be any distribution (prior or posterior) on the parameters. The algorithm is iterative and at iteration t, it uses:

- *a proposal distribution $q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$, parametrised by the current state of the Markov chain, i.e. $\boldsymbol{\theta}^{(t)}$;*
- *a function $p^*(\boldsymbol{\theta})$, which is proportional to $p(\boldsymbol{\theta})$. In other words, $p^*(\boldsymbol{\theta})$ is unnormalised[2] and the normalised density $p(\boldsymbol{\theta})$ is*

$$p(\boldsymbol{\theta}) = \frac{p^*(\boldsymbol{\theta})}{\int p^*(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}}. \tag{4}$$

Read Section 27.4 in Barber's book to familiarise yourself with the Metropolis-Hastings algorithm.

For all tasks in this exercise, we work with a Gaussian proposal distribution $q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$, whose mean is the previous sample in the Markov chain, and whose variance is ϵ^2. That is, at iteration t of our Metropolis-Hastings algorithm,

$$q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t-1)}) = \prod_{k=1}^{d} \mathcal{N}(\theta_k; \theta_k^{(t-1)}, \epsilon^2). \tag{5}$$

When used with this proposal distribution, the algorithm is called Random Walk Metropolis-Hastings algorithm.

Exercise 1. *Basic Markov chain Monte Carlo inference*

(a) Write a function mh *implementing the Metropolis Hasting algorithm, as e.g. given in Algorithm 27.3 in Barber's book, using the Gaussian proposal distribution in (5) above. The function should take as arguments*

- *p_star: a function on $\boldsymbol{\theta}$ that is proportional to the density of interest $p^*(\boldsymbol{\theta})$;*
- *param_init: the initial sample — a value for $\boldsymbol{\theta}$ from where the Markov chain starts;*
- *num_samples: the number S of samples to generate;*
- *vari: the variance ϵ^2 for the Gaussian proposal distribution q;*

and return $[\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(S)}]$ — a list of S samples from $p(\boldsymbol{\theta}) \propto p^*(\boldsymbol{\theta})$. For example:

```
def mh(p_star, param_init, num_samples=5000, vari=1.0):
    # your code here
    return samples
```

Solution. Below is a Python implementation.

```
def mh(p_star, param_init, num_samples=5000, vari=1.0):
  x = []

  x_current = param_init
  for n in range(num_samples):

    # proposal
    x_proposed = multivariate_normal.rvs(mean=x_current, cov=vari)

    # MH step
```

[2] We used the notation $\tilde{p}$ in the lecture slides; p^* is also commonly used, e.g. in Barber's book.

```
    a = multivariate_normal.pdf(x_current, mean=x_proposed, cov=vari) *
        p_star(x_proposed)
    a = a / (multivariate_normal.pdf(x_proposed, mean=x_current, cov=vari)
        * p_star(x_current))

    # accept or not
    if a >= 1:
      x_next = np.copy(x_proposed)
    elif uniform.rvs(0, 1) < a:
      x_next = np.copy(x_proposed)
    else:
      x_next = np.copy(x_current)

    # keep record
    x.append(x_next)
    x_current = x_next

  return x
```

As we are using a symmetrical proposal distribution, $q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*) = q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$, one could simplify the algorithm by having $a = \frac{p^*(\boldsymbol{\theta}^*)}{p^*(\boldsymbol{\theta})}$, where $\boldsymbol{\theta}$ is the current sample and $\boldsymbol{\theta}^*$ is the proposed sample.

In practice, it is desirable to implement the function in the log domain, to avoid numerical problems. That is, instead of p^*, mh will accept as an argument $\log p^*$, and a will be calculated as:

$$a = (\log q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*) + \log p^*(\boldsymbol{\theta}^*)) - (\log q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) + \log p^*(\boldsymbol{\theta}))$$

(b) Test your algorithm by sampling $5,000$ *samples from* $p(x, y) = \mathcal{N}(x; 0, 1)\mathcal{N}(y; 0, 1)$. *Initialise at* $(x = 0, y = 0)$ *and use* $\epsilon^2 = 1$. *Generate a scatter plot of the obtained samples. The plot should be similar to Figure 1b. Highlight the first 20 samples only. Do these 20 samples alone adequately approximate the true density?*

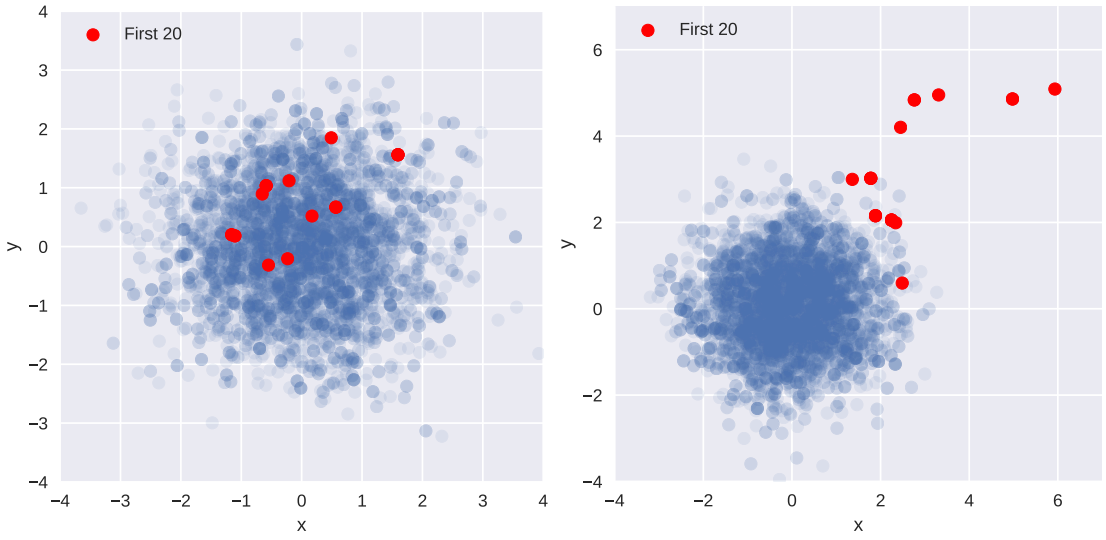
Sample another $5,000$ *points from* $p(x, y) = \mathcal{N}(x; 0, 1)\mathcal{N}(y; 0, 1)$ *using* mh *with* $\epsilon^2 = 1$, *but this time initialise at* $(x = 7, y = 7)$. *Generate a scatter plot of the drawn samples and highlight the first 20 samples. If everything went as expected, your plot probably shows a "trail" of samples, starting at* $(x = 7, y = 7)$ *and slowly approaching the region of space where most of the probability mass is.*

In practice, we don't know where the distribution we wish to sample from has high density, so we typically initialise the Markov Chain somewhat arbitrarily, or at the maximum a-posterior sample if available. The samples obtained in the beginning of the chain are typically discarded, as they are not considered to be representative of the target distribution. This initial period between initialisation and starting to collect samples is called "warm-up", or also "burn-in".

Extended your function mh *to include an additional warm-up argument* W, *which specifies the number of MCMC steps taken before starting to collect samples. Your function should still return a list of* S *samples as in (a).*

Solution. Figure 2 shows the two scatter plots of draws from $\mathcal{N}(x; 0, 1)\mathcal{N}(y; 0, 1)$:

- Figure 2a highlights the first 20 samples obtained by the chain when starting at $(x = 0, y = 0)$. They appear to be representative samples from the distribution, however, they are nearly not enough to approximate the distribution on their own.



(a) Starting the chain at $(0, 0)$. (b) Starting the chain at $(7, 7)$

Figure 2: $5,000$ samples from $\mathcal{N}(x; 0, 1)\mathcal{N}(y; 0, 1)$ (blue), with the first 20 samples highlighted (red). Drawn using Metropolis-Hastings with different starting points.

- Figure 2b highlights the first 20 samples obtained by the chain when starting at $(x = 7, y = 7)$. One can clearly see the "burn-in" tail which slowly approaches the region where most of the probability mass is.

It is straightforward to extend the `mh` function with a warm-up argument. For example, one can simply iterate for `num_samples + warmup` steps, and start recording samples only after the warm-up period:

```
def mh(p_star, param_init, num_samples=5000, vari=1.0, warmup=0):
    x = []
    x_current = param_init
    for n in range(num_samples+warmup):
        ... # body same as before

        if n >= warmup: x.append(x_next)
        x_current = x_next

    return x
```

Exercise 2. ***Bayesian Poisson regression*** **(optional, not examinable)**

Consider a Bayesian Poisson regression model, where outputs y_n are generated from a Poisson distribution of rate $\exp(\alpha x_n + \beta)$, where the x_n are the inputs (covariates), and α and β the parameters of the regression model for which we assume a broad Gaussian prior:

$$\alpha \sim \mathcal{N}(\alpha; 0, 100) \tag{6}$$
$$\beta \sim \mathcal{N}(\beta; 0, 100) \tag{7}$$
$$y_n \sim \text{Poisson}(y_n; \exp(\alpha x_n + \beta)) \quad \textit{for } n = 1, \ldots, N \tag{8}$$

$\text{Poisson}(y;\lambda)$ *denotes the probability mass function of a Poisson random variable with rate* λ*,*

$$\text{Poisson}(y;\lambda) = \frac{\lambda^y}{y!}\exp(-\lambda), \qquad y \in \{0,1,2,\ldots\}, \quad \lambda > 0 \tag{9}$$

Consider $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ *where* $N = 5$ *and*

$$(x_1,\ldots,x_5) = (-0.50519053, -0.17185719, 0.16147614, 0.49480947, 0.81509851) \tag{10}$$

$$(y_1,\ldots,y_5) = (1,0,2,1,2) \tag{11}$$

We are interested in computing the posterior density of the parameters (α,β) *given the data* $\mathcal{D}$ *above.*

(a) Derive an expression for the unnormalised posterior density of α *and* β *given* $\mathcal{D}$*, i.e. a function* p^* *of the parameters* α *and* β *that is proportional to the posterior density* $p(\alpha,\beta \mid \mathcal{D})$*, and which can thus be used as target density in the Metropolis Hastings algorithm.*

Solution. By the product rule, the joint distribution described by the model, with $\mathcal{D}$ plugged in, is proportional to the posterior and hence can be taken as p^*:

$$p^*(\alpha,\beta) = p(\alpha,\beta,\{(x_n,y_n)\}_{n=1}^N) \tag{S.1}$$

$$= \mathcal{N}(\alpha;0,100)\mathcal{N}(\beta;0,100)\prod_{n=1}^N \text{Poisson}(y_n \mid \exp(\alpha x_n + \beta)) \tag{S.2}$$

(b) Implement the derived unnormalised posterior density p^**. If your coding environment provides an implementation of the above Poisson pmf, you may use it directly rather than implementing the pmf yourself.*

Use the Metropolis Hastings algorithm from Question 1(b) to draw $5,000$ *samples from the posterior density* $p(\alpha,\beta \mid \mathcal{D})$*. Set the hyperparameters of the Metropolis-Hastings algorithm to:*

- *param_init* $= (\alpha_{\text{init}}, \beta_{\text{init}}) = (0,0)$*,*
- *vari* $= 1$*, and*
- *number of warm-up steps* $W = 1000$*.*

Plot the drawn samples with x-axis α *and y-axis* β *and report the posterior mean of* α *and* β*, as well as their correlation coefficient under the posterior.*

Solution. A Python implementation is:

```
import numpy as np
from scipy.stats import multivariate_normal, norm, poisson, uniform

xx1 = np.array([-0.5051905265552105, -0.17185719322187715,
    0.16147614011145617, 0.49480947344478954, 0.8150985069051909])
yy1 = np.array([1, 0, 2, 1, 2])
N1 = len(xx1)

def poisson_regression(params):
    a = params[0]
    b = params[1]
    # mean zero, standard deviation 10 == variance 100
    p = norm.pdf(a, loc=0, scale=10) * norm.pdf(b, loc=0, scale=10)
    for n in range(N1):
```

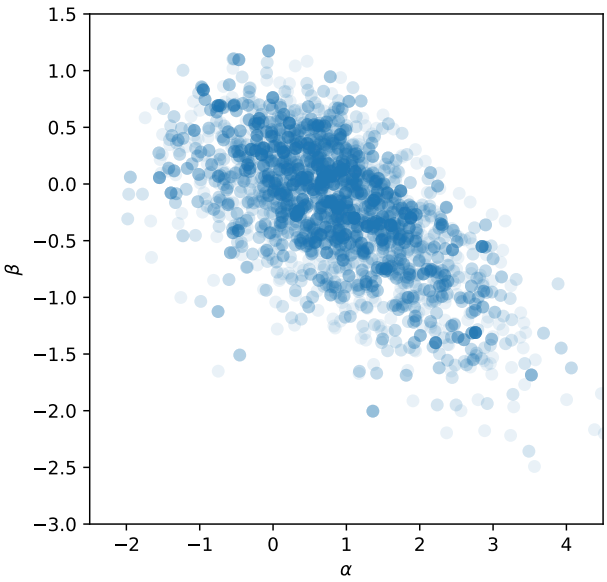


Figure 3: Posterior samples for Poisson regression problem; $\boldsymbol{\theta}_{\text{init}} = (0, 0)$.

```
        p = p * poisson.pmf(yy1[n], np.exp(a * xx1[n] + b))

    return p

# sample
S = 5000
samples = np.array(mh(poisson_regression, np.array([0, 0]), num_samples=S,
    vari=1.0, warmup=1000))
```

A scatter plot showing 5,000 samples from the posterior is shown on Figure 3. The posterior mean of α is 0.84, the posterior mean of β is -0.2, and posterior correlation coefficient is -0.63. Note that the numerical values are sample-specific.

Exercise 3. ***Mixing and convergence of Metropolis-Hasting MCMC***

Any MCMC algorithm is an asymptotically exact inference algorithm, meaning that if it is run forever, it will converge to the desired probability distribution. In practice, we want to run the algorithm long enough to be able to approximate the posterior adequately. How long is long enough for the chain to converge varies drastically depending on the algorithm, the hyperparameters (e.g. the variance vari*), and the target posterior distribution. It is impossible to determine exactly whether the chain has run long enough, but there exist various diagnostics that can help us determine if we can "trust" the sample-based approximation to the posterior.*

A very quick and common way of assessing convergence of the Markov chain is to visually inspect the trace plots for each parameter. A trace plot shows how the drawn samples evolve through time, i.e. they are a time-series of the samples generated by the Markov chain. Figure 4 shows examples of trace plots obtained by running the Metropolis Hastings algorithm for different values of the hyperparameters vari *and* param_init*. Ideally, the time series covers the whole domain of the target distribution and it is hard to "see" any structure in it so that predicting values of future samples from the current one is difficult. If so, the samples are likely independent from each other and the chain is said to be well "mixed".*

(a) Consider the trace plots in Figure 4: Is the variance vari *used in Figure 4b larger or smaller than*

the value of vari *used in Figure 4a? Is* vari *used in Figure 4c larger or smaller than the value used in Figure 4a?*

In both cases, explain the behaviour of the trace plots in terms of the workings of the Metropolis Hastings algorithm and the effect of the variance vari*.*

Solution. MCMC methods are sensitive to different hyperparameters, and we usually need to carefully diagnose the inference results to ensure that our algorithm adequately approximates the target posterior distribution.

(i) Figure 4b uses a *small* variance (vari was set to 0.001) . The trace plots show that the samples for β are very highly correlated and evolve very slowly through time. This is because the introduced randomness is quite small compared to the scale of the posterior, thus the proposed sample at each MCMC iteration will be very close to the current sample and hence likely accepted.

More mathematical explanation: for a symmetric proposal distribution, the acceptance ratio a becomes

$$a = \frac{p^*(\boldsymbol{\theta}^*)}{p^*(\boldsymbol{\theta})}, \tag{S.3}$$

where $\boldsymbol{\theta}$ is the current sample and $\boldsymbol{\theta}^*$ is the proposed sample. For variances that are small compared to the (squared) scale of the posterior, a is close to one and the proposed sample $\boldsymbol{\theta}^*$ gets likely accepted. This then gives rise to the slowly changing time series shown in Figure 4b.

(ii) In Figure 4c, the variance is larger than the reference (vari was set to 50) . The trace plots suggest that many iterations of the algorithm result in the proposed sample being rejected, and thus we end up copying the same sample over and over again. This is because if the random perturbations are large compared to the scale of the posterior, $p^*(\boldsymbol{\theta}^*)$ may be very different from $p^*(\boldsymbol{\theta})$ and a may be very small.

(b) In Metropolis-Hastings, and MCMC in general, any sample depends on the previously generated sample, and hence the algorithm generates samples that are generally statistically dependent. The effective sample size of a sequence of dependent samples is the number of independent samples that are, in some sense, equivalent to our number of dependent samples. A definition of the effective sample size (ESS) is

$$ESS = \frac{S}{1 + 2\sum_{k=1}^{\infty} \rho(k)} \tag{12}$$

where S is the number of dependent samples drawn and $\rho(k)$ the correlation coefficient between two samples in the Markov chain that are k time points apart. We can see that if the samples are strongly correlated, $\sum_{k=1}^{\infty} \rho(k)$ is large and the effective sample size is small. On the other hand, if $\rho(k) = 0$ for all k, the effective sample size is S.

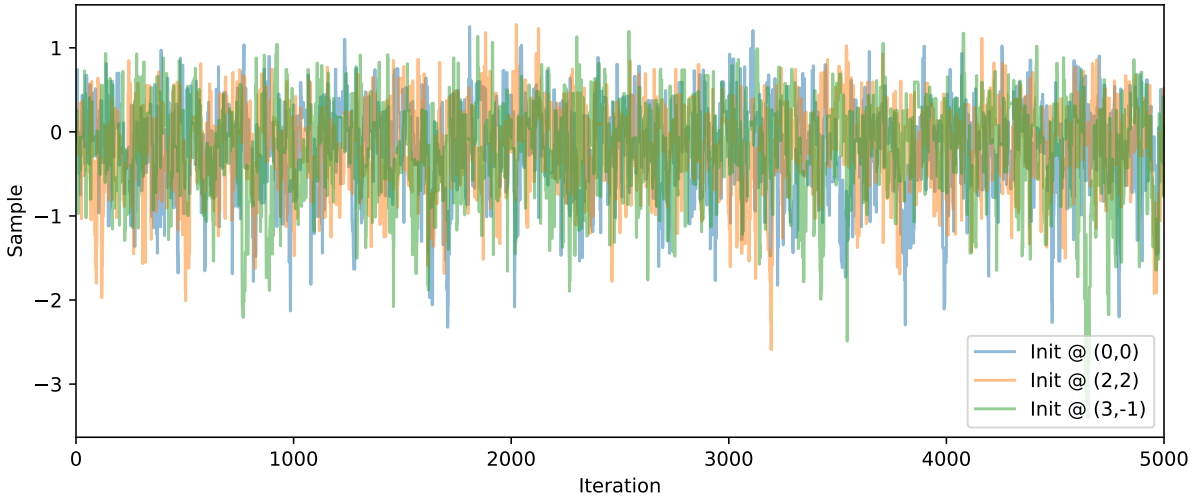
ESS, as defined above, is the number of independent samples which are needed to obtain a sample average that has the same variance as the sample average computed from correlated samples.

To illustrate how correlation between samples is related to a reduction of sample size, consider two pairs of samples (θ_1, θ_2) and (ω_1, ω_2). All variables have variance σ^2 and the same mean μ, but ω_1 and ω_1 are uncorrelated while the covariance matrix for θ_1, θ_2 is $\mathbf{C}$,

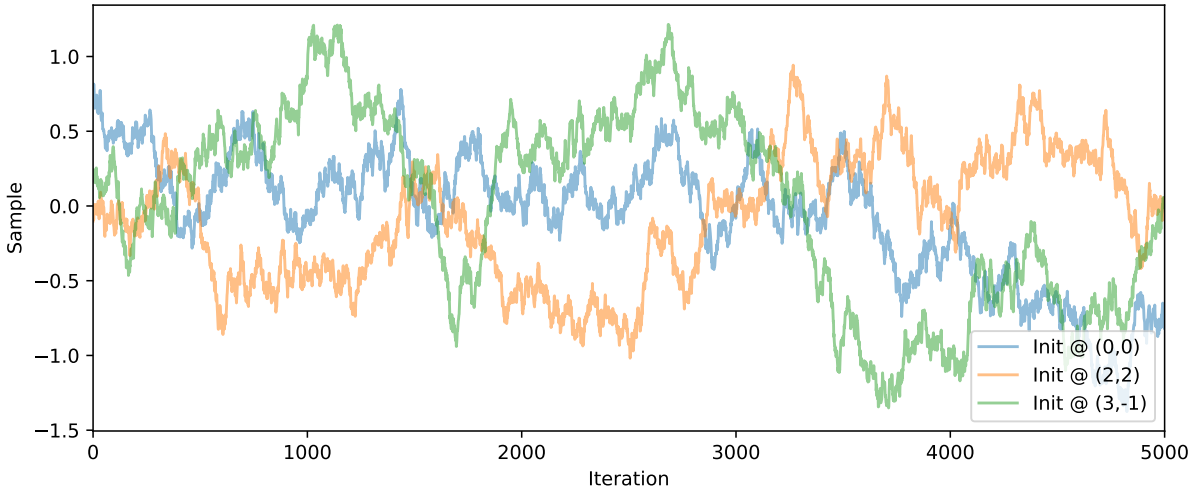
$$\mathbf{C} = \sigma^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}, \tag{13}$$

with $\rho > 0$. The variance of the average $\bar{\omega} = 0.5(\omega_1 + \omega_2)$ is

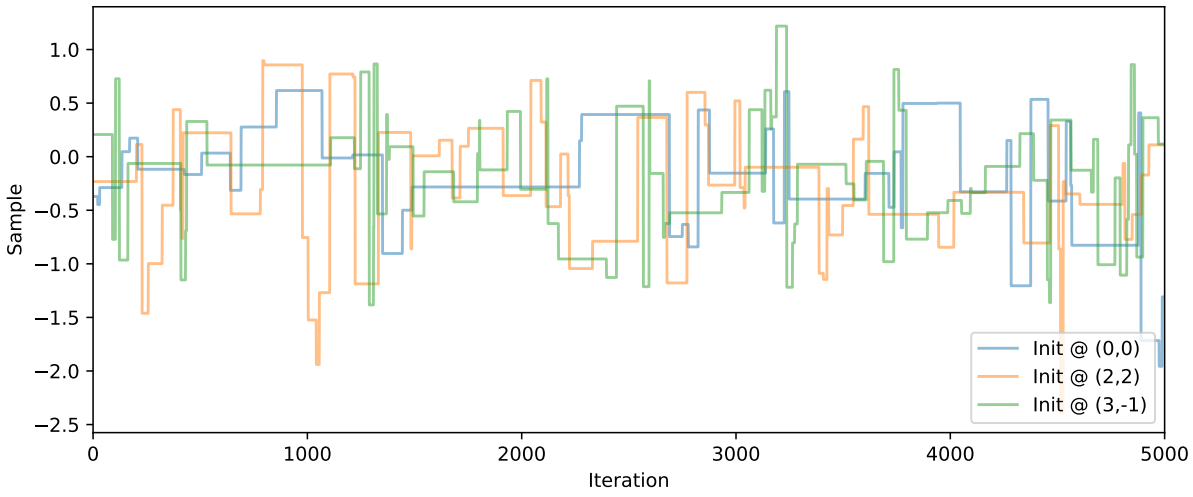
$$\mathbb{V}(\bar{\omega}) = \frac{\sigma^2}{2}, \tag{14}$$



(a) variance vari: 1



(b) Alternative value of vari



(c) Alternative value of vari

Figure 4: For Question 3(a): Trace plots of the parameter β from Question 2 drawn using Metropolis-Hastings with different variances of the proposal distribution.

where the 2 in the denominator is the sample size.

Derive an equation for the variance of $\bar{\theta} = 0.5(\theta_1 + \theta_2)$ and compute the reduction α of the sample size when working with the correlated (θ_1, θ_2). In other words, derive an equation of α in

$$\mathbb{V}\left(\bar{\theta}\right) = \frac{\sigma^2}{2/\alpha}. \tag{15}$$

What is the effective sample size $2/\alpha$ as $\rho \to 1$?

Solution. Note that $\mathbb{E}(\bar{\theta}) = \mu$. From the definition of variance, we then have

$$\begin{aligned}
\mathbb{V}(\bar{\theta}) &= \mathbb{E}\left((\bar{\theta} - \mu)^2\right) && \text{(S.4)}\\
&= \mathbb{E}\left(\left(\frac{1}{2}(\theta_1 + \theta_2) - \mu\right)^2\right) && \text{(S.5)}\\
&= \mathbb{E}\left(\left(\frac{1}{2}(\theta_1 - \mu + \theta_2 - \mu\right)^2\right) && \text{(S.6)}\\
&= \frac{1}{4}\mathbb{E}\left((\theta_1 - \mu)^2 + (\theta_2 - \mu)^2 + 2(\theta_1 - \mu)(\theta_2 - \mu)\right) && \text{(S.7)}\\
&= \frac{1}{4}(\sigma^2 + \sigma^2 + 2\sigma^2\rho) && \text{(S.8)}\\
&= \frac{1}{4}(2\sigma^2 + 2\sigma^2\rho) && \text{(S.9)}\\
&= \frac{\sigma^2}{2}(1 + \rho) && \text{(S.10)}\\
&= \frac{\sigma^2}{2/(1 + \rho)} && \text{(S.11)}
\end{aligned}$$

Hence: $\alpha = (1 + \rho)$, and for $\rho \to 1$, $2/\alpha \to 1$.

Because of the strong correlation, we effectively only have one sample if $\rho \to 1$.