EXERCISES 7

Exercise 7.1. Consider weights

$$w_1 = 0.2993 \qquad \text{and} \qquad w_2 = 0.7007,$$

and the following matrices A_1, A_2 and vectors b_1, b_2

$$A_1 = \begin{bmatrix} 0.4 & -0.3733 \\ 0.06 & 0.6 \end{bmatrix} \qquad \text{and} \qquad b_1 = \begin{bmatrix} 0.3533 \\ 0.0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -0.8 & -0.1867 \\ 0.1371 & 0.8 \end{bmatrix} \qquad \text{and} \qquad b_2 = \begin{bmatrix} 1.1 \\ 0.1 \end{bmatrix}$$

Now we will define a Markov chain on $\mathbb{R}^2$ and we will denote our chain with $(x_k)_{k \geq 0}$. For this, consider two deterministic functions:

$$\begin{aligned} f_1(x) &= A_1 x + b_1 \\ f_2(x) &= A_2 x + b_2. \end{aligned}$$

Simulate the following Markov process

$$\begin{aligned} i_n &\sim \text{Discrete}(w_1, w_2) \\ x_{n+1} &= f_{i_n}(x_n) \end{aligned}$$

for $1 \leq n \leq N$ where $N = 10000$. You can use $x_0 = [0, 0]^\top$. Plot a scatter plot of the Markov chain. For a pretty plot, you can use the following plotting function

```
plt.scatter(x[0, 20:k], x[1, 20:k], s=0.1, color = [0.8, 0, 0])
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['bottom'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().set_xticks([])
plt.gca().set_yticks([])
plt.gca().set_xlim(0, 1.05)
plt.gca().set_ylim(0, 1)
plt.show()
```

As you can see, I chose `burnin` as 20 iterations here. Be careful about matrix products in Python: `A*x` **is not performing matrix product**, you should use `A@x` where A is a 2×2 matrix and x is a 2×1 vector. Try to simulate from this system until you see a nice picture.

Exercise 7.2. Consider the real-valued Markov kernel

$$K(x_n | x_{n-1}) = \mathcal{N}(x_n; a x_{n-1}, 1).$$

Show that this kernel satisfies the detailed balance condition w.r.t.

$$p_\star(x) = \mathcal{N}\left(x; 0, \frac{1}{1 - a^2}\right).$$

Exercise 7.3. Show that the kernel in the previous exercise satisfies

$$p_\star(x) = \lim_{n \to \infty} K^{(n)}(x | x').$$

In other words, regardless from where we start, the kernel would converge to the stationary distribution.

Hint: Start from writing the recursions $x_{n+1} = a x_n + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, 1)$ and $x_0 = x'$. Try to first derive the expression of x_{n+1} in terms of x_0. Compute the mean and the variance of x_{n+1}, then take the limit $n \to \infty$.

Exercise 7.4. Sample from the banana density of Example 5.10 using MH sampler:

$$p(x, y) \propto \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y - x^2)^2\right).$$

1. Use a symmetric random walk proposal for each dimension

$$q(x', y'|x, y) = \mathcal{N}(x'; x, \sigma_q^2)\mathcal{N}(y'; y, \sigma_q^2).$$

   Compute your acceptance ratio using $\log$ density and accept if $\log U < \log \mathsf{r}$ (for practice).

2. Use the Metropolis-adjusted Langevin algorithm proposal (MALA). Surprisingly, this may work worse than random walk for this problem.

Use the following snippet for 2D plotting:

```
x_bb = np.linspace(-4, 4, 100)
y_bb = np.linspace(-2, 6, 100)
X_bb, Y_bb = np.meshgrid(x_bb, y_bb)
Z_bb = np.exp(banana(X_bb, Y_bb)) # your banana function
plt.subplot(1, 3, 1)
plt.contourf(X_bb, Y_bb, Z_bb, 100, cmap='RdBu')
plt.subplot(1, 3, 2)
plt.hist2d(samples_RW[0, burnin:n], samples_RW[1, burnin:n], 100, cmap='RdBu', range=[[-4, 4], [-2, 6]], density=True)
plt.title('Random Walk Metropolis')
plt.subplot(1, 3, 3)
plt.hist2d(samples_Langevin[0, burnin:n], samples_Langevin[1, burnin:n], 100, cmap='RdBu', range=[[-4, 4], [-2, 6]], density=True)
plt.title('Metropolis Adjusted Langevin Algorithm')
plt.show()
```