CODING/PROBLEM SESSION EXERCISE SHEET – 22 NOV 2022

For those of you who understood the solutions of the exercises posted earlier, here are a few more exercises to be worked on during the coding/problem session.

Q1: SAMPLING FROM THE MIXTURE OF GAUSSIANS

Sample from a mixture of Gaussians using MH:

$$p_\star(x) = w_1 \mathcal{N}(x; \mu_1, \sigma_1^2) + w_2 \mathcal{N}(x; \mu_2, \sigma_2^2),$$

using a symmetric proposal

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

See Example 5.5 from lecture notes for the acceptance ratio. Try and produce results with different σ_q choices.

Q2: SAMPLING FROM THE BANANA DENSITY

Sample from the banana density of Example 5.11 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:

```python
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()
```

Q3: Gibbs sampling for image denoising

We will replicate the result in Example 5.9 in this exercise.

Consider a set of random variables X_{ij} for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. This is a matrix modeling an $m \times n$ image. We assume that we have an image that takes values $X_{ij} \in \{-1, 1\}$ – note that this is an "unusual" image, as the images usually take values between $[0, 255]$ (or $[0, 1]$). We assume that the image is corrupted by noise, i.e., we have a noisy image

$$Y_{ij} = X_{ij} + \sigma \epsilon_{ij},$$

where $\epsilon_{ij} \sim \mathcal{N}(0, 1)$ and σ is the standard deviation of the noise. We assume that the noise is independent of the image. We want to recover the image X_{ij} from the noisy image Y_{ij} and utilise Gibbs sampler for this purpose.

Our aim is to obtain (conceptually) $p(X|Y)$, i.e., samples from $p(X|Y)$ given Y. For this, we need to specify a prior $p(X)$. We take this from the literature and place as a prior a smooth Markov random field (MRF) assumption. This is formalised as

$$p(X_{ij}|X_{-ij}) = \frac{1}{Z} \exp(J X_{ij} W_{ij}),$$

where W_{ij} is the sum of the X_{ij}'s in the neighbourhood of X_{ij}, i.e.,

$$W_{ij} = \sum_{kl:\text{neighbourhood of } (i,j)} X_{kl} = X_{i-1,j} + X_{i+1,j} + X_{i,j-1} + X_{i,j+1}.$$

This is an intuitive model of the image, making the current value of the pixel depend on the values of its neighbours.

We aim at using a Gibbs sampler approach from sampling the posterior $p(X|Y)$. Note that now we need to sample from full conditionals, e.g., for each (i, j), we need to sample from $X_{ij} \sim p(X_{ij}|X_{-ij}, Y_{ij})$. We derive the full conditional as

$$p(X_{ij} = k|X_{-ij}, Y_{ij}) = \frac{p(Y_{ij}|X_{ij} = k) p(X_{ij} = k|X_{-ij})}{\sum_{k \in \{-1,1\}} p(Y_{ij}|X_{ij} = k) p(X_{ij} = k|X_{-ij})},$$

where $p(Y_{ij}|X_{ij} = k) = \mathcal{N}(Y_{ij}; k, \sigma^2)$ is the likelihood of the noisy image given the value of the pixel. We can easily compute these probabilities since each term in the Bayes rule is computable (and $1/Z$ cancels). Therefore, we can get explicit expressions for $q = p(X_{ij} = 1|X_{-ij}, Y_{ij})$ and $1 - q = p(X_{ij} = -1|X_{-ij}, Y_{ij})$. We can then sample from the full conditional as

$$X_{ij} \sim \begin{cases} 1 & \text{with probability } q, \\ -1 & \text{with probability } 1 - q. \end{cases}$$

We can now loop over (i, j) (to sample from each full conditional) and sample from the full conditionals.

1. Download the noisy image from https://akyildiz.me/images/image.png
2. Read it and convert it to $\{-1, 1\}$-valued image using

```
import numpy as np
import matplotlib.pyplot as plt


img = plt.imread('riceImage.png')
img = img[:, :, 0]

img[img < 0.5] = -1
img[img >= 0.5] = 1
```

Next generate your noisy image

```
sig = 1
img_noisy = img + sig * np.random.randn(*img.shape)

fig = plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img, cmap='gray', vmin=-1, vmax=1)
plt.subplot(1, 2, 2)
plt.imshow(img_noisy, cmap='gray', vmin=-1, vmax=1)
```

At this point, you are ready to define the Gibbs sampler. Choose $J = 4$ and implement the Gibbs sampler to denoise this image.