EXERCISES 6

Exercise 6.1 (Bayesian Inference with SNIS)**.** In this exercise, we will use our toy Gaussian model to perform Bayesian inference. We will use the same model Example 3.6. Recall that this model is given as

$$p(x) = \mathcal{N}(x; \mu_0, \sigma_0^2),$$
$$p(y_i|x) = \mathcal{N}(y_i; x, \sigma^2),$$

where $i = 1, \ldots, M$, i.e., y_i are conditionally independent given x. In the first implementation of this exercise, you can set $\mu_0 = 0$, $\sigma^2 = 1$ and $\sigma_0^2 = 1$, but you should also explore other parameters to gain intuition.

1. First simulate $M = 1000$ data points from this model. In particular, you should perform $X \sim p(x)$ and fixing the sample $X = x$, and then $y_i \sim p(y_i|x)$ for $i = 1, \ldots, M$.

2. For these particular datapoints, compute the *true* posterior mean estimate using the expression μ_p in Example 3.6. This is the true posterior mean estimate for this model.

3. Now we will use SNIS to estimate the posterior mean. For this, we will use the following proposal

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

   In the first implementation, you can set $\mu_q = 0$ and $\sigma_q^2 = 1$, but you should also explore other parameters to gain intuition. Implement the SNIS estimator for this model. Note that your unnormalized posterior is

   $$\bar{p}(x|y_{1:n}) = p(y_{1:n}|x)p(x) = \prod_{i=1}^{M} p(y_i|x)p(x),$$

   as given in Exercise 3.6. You should compute the weights in the log-domain. Compare the posterior mean estimate when you compute the weights without the log-trick and with the log-trick. You should observe that the log-trick is necessary for this model.

4. Finally compute effective sample size (ESS) and vary your N to see how it affects the ESS. Plot the ESS w.r.t. N.

Exercise 6.2. In this example, we will work on $\mathbb{R}^2$. Let us assume the following "banana" prior

$$p(x) \propto \exp(-x_1^2/10 - x_2^2/10 - 2(x_2 - x_1^2)^2),$$

where $x \in \mathbb{R}^2$. We would like to then use the following likelihood:

$$p(y|x) = \mathcal{N}(y; Hx, \sigma^2 I),$$

where $H = [1, 0]$ and $y \in \mathbb{R}$. Now assume that we are given $y = 1$ and $\sigma = 0.05$ (note that this is standard deviation). Our goal is to sample from $p(x|y)$.

1. First visualise the prior $p(x)$ using the following code:

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

def bar_p(x):
    return np.exp(-x[0]**2/10 - x[1]**2/10 - 2 * (x[1] - x[0]**2)
                                            **2)

x_bb = np.linspace(-4, 4, 100)
y_bb = np.linspace(-2, 6, 100)
X_bb, Y_bb = np.meshgrid(x_bb, y_bb)
# evaluate barp on this grid
Z_bb = np.zeros((100, 100))
for i in range(100):
    for j in range(100):
        Z_bb[i, j] = bar_p([X_bb[i, j], Y_bb[i, j]])
# plot barp
plt.contourf(X_bb, Y_bb, Z_bb, 100, cmap='RdBu')
plt.show()
```

You will see that this gives you a 2D (unnormalised) density. Now try to interpret our task. We assume that $y = 1$ with $\sigma = 0.05$. By just looking at the prior and the likelihood, try to guess how the posterior should look like (before proceeding).

2. Now we will use SNIS to estimate the posterior mean. For this, we will use the following proposal

$$q(x) = \mathcal{N}(x; \mu_q, \sigma_q^2 I),$$

Note that this is also defined on $\mathbb{R}^2$ and you can set $\mu_q = [0, 0]$ and $\sigma_q^2 = 1$. Compute the posterior mean using SNIS (do not forget using log-prior and log-likelihood and performing the log-trick) and compute the ESS (choose varying N. Start with $N = 1000$).

3. Now using the idea of importance sampling resampling (Section 4.6.2) in lecture notes, resample N samples from this weighted sample. Recall that SNIS will result in a set of weights and samples $\{\bar{w}_i, x_i\}_{i=1}^N$. You should resample from this set to obtain a new set of *resampled* samples $\{\bar{x}_i\}_{i=1}^N$. Then scatter these samples to plot against the prior using the following code:

```python
# plot resampled samples
# x_res are the resampled samples (note that samples are 2D)
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.zeros((100, 100))
for i in range(100):
    for j in range(100):
        Z_bb[i, j] = bar_p([X_bb[i, j], Y_bb[i, j]])
plt.contourf(X_bb, Y_bb, Z_bb, 100, cmap='RdBu')
plt.scatter(x_res[0, :], x_resampled[1, :], s=10, c='white')
plt.show()
```

Now interpret the result and compare it to your educated guess you have done in Part 1.