Bayesian inference and Data assimilation

Exercise 12

Jin W. Kim (jin.won.kim@uni-potsdam.de)

Problem 1

1. Since the expectation is linear,

$$\mathbb{E}\big[\bar{g}_X\big] = \mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N} g(X_i)\Big] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\big[g(X_i)\big] = \pi[g]$$

Now the variance is

$$\begin{aligned}
\mathrm{Var}\left[\bar{g}_X\right] &= \mathbb{E}\Big[\big(\bar{g}_X - \pi[g]\big)^2\Big] \\
&= \mathbb{E}\Big[\Big(\frac{1}{N}\sum_{i=1}^{N} g(X_i) - \pi[g]\Big)^2\Big] \\
&= \frac{1}{N^2}\mathbb{E}\Big[\Big(\sum_{i=1}^{N}\big(g(X_i)-\pi[g]\big)\Big)\Big(\sum_{j=1}^{N}\big(g(X_j)-\pi[g]\big)\Big)\Big] \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\Big(\mathbb{E}\big[\big(g(X_i)-\pi[g]\big)^2\big] + \sum_{j\neq i}\mathbb{E}\big[\big(g(X_i)-\pi[g]\big)\big(g(X_j)-\pi[g]\big)\big]\Big) \\
&= \frac{1}{N}\,\mathrm{Var}\left[g(X)\right] \\
&= \frac{1}{N}\int \big(g(x)-\pi[g]\big)^2\pi(x)\,\mathrm{d}x
\end{aligned}$$

because the mean of the cross terms are all zero.

2. Note that it has to be assumed that $\pi'(x) = 0$ implies $\pi(x) = 0$, in order to make sense of the ratio. We have

$$\begin{aligned}
\pi[g] &= \int g(x)\pi(x)\,\mathrm{d}x \\
&= \int g(x)\frac{\pi(x)}{\pi'(x)}\pi'(x)\,\mathrm{d}x \\
&= \mathbb{E}\Big[\frac{\pi(Y)}{\pi'(Y)}g(Y)\Big]
\end{aligned}$$

3. It is identical to the step 1. We have

$$\begin{aligned}
\mathrm{Var}\left[\bar{g}_Y\right] &= \frac{1}{N}\,\mathrm{Var}\left[\frac{\pi(Y)}{\pi'(Y)}g(Y)\right] \\
&= \frac{1}{N}\int \Big(\frac{\pi(y)g(y)}{\pi'(y)} - \pi[g]\Big)^2\pi'(y)\,\mathrm{d}y
\end{aligned}$$

4. Note that $g(x)\pi(x) \geq 0$, and therefore we can obtain a probability density function by normalizing it. Namely, we choose

$$\pi'(x) = \frac{1}{\int g(x)\pi(x)\,\mathrm{d}x} g(x)\pi(x) = \frac{1}{\pi[g]} g(x)\pi(x)$$

Then the variance of $\bar{g}_Y$ is zero.

5. Let $\pi' = N(m, \sigma)$. Here is an example of pseudo-code (for scalar case):

Algorithm 1 Importance resampling

```
Input: π, N, m and σ
w_sum = 0
for i = 0; i < N do
  y[i] ~ Normal(m, σ)
  w[i] = π(y[i]) · exp( (1/(2σ)) (y[i] − m)^2 )
  w_sum = w_sum + w[i]
end for
for i = 0; i < N do
  w[i] = w[i] / w_sum
end for
for i = 0; i < N do
  x[i] ~ Uniform(0, 1]
  cdf = 0
  for j = 0; j < N do
    cdf = cdf + w[j]
    if x[i] ≤ cdf then
      x[i] = y[j]
      break j;
    end if
  end for
end for
```

Of course one can easily make it parallel.

Problem 2

- We initialize with $x_0^i \sim p_0$ and the weights are $w_0^i = \dfrac{1}{N}$. Each time step, we propagate the state process and compute the likelihood. The weights are updated by

$$w_t^i = w_{t-1}^i q_t(y_t \mid x_t^i)$$

- After we compute the weights, they are normalized, and effective sample size is computed.
- If the effective sample size is below the threshold, then perform resampling.

Algorithm 2 Sequential Importance Resampling (SIR) filter

```
Input: N, T, {y[t]} and γ
for i = 0; i < N do
  x[i] ~ p_0
  w[i] = 1/N
end for
for t = 1; t ≤ N do
  w_sum = 0
  for i = 0; i < N do
    x[i] ~ p_t( · | x[i])
    w[i] = w[i] · q_t(y[t] | x[i])
    w_sum = w_sum + w[i]
  end for
  ess = 0
  for i = 0; i < N do
    w[i] = w[i] / w_sum
    ess = ess + (w[i])^2
  end for
  ess = 1 / ess
  if ess < γN then
    for i = 0; i < N do
      x_tmp[i] ~ Uniform(0, 1]
      cdf = 0
      for j = 0; j < N do
        cdf = cdf + w[j]
        if x_tmp[i] ≤ cdf then
          x_tmp[i] = x[j]
          break j;
        end if
      end for
      w[i] = 1/N
    end for
    x ← x_tmp
  end if
end for
```

Problem 3 We use the unnormalized pdf

$$\pi(x) = \frac{1}{2}\exp\Big(-\frac{1}{2}(x+4)^2\Big) + \frac{1}{2}\exp\Big(-\frac{1}{2}(x-4)^2\Big)$$

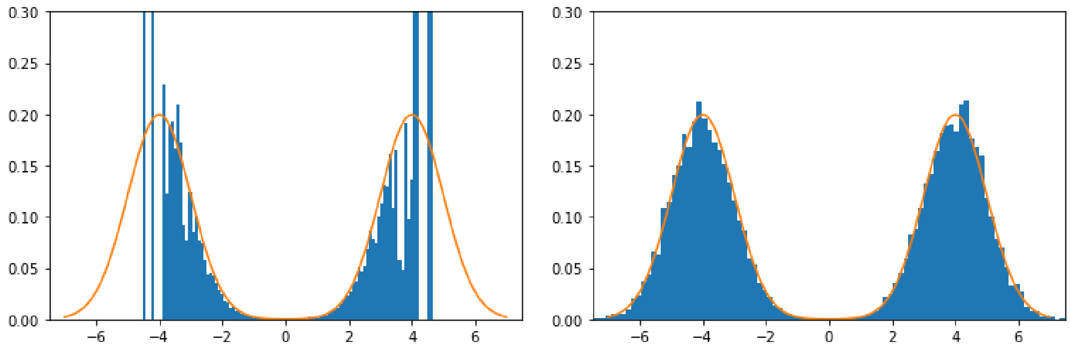
and (unnormalized) Gaussian proposal is given by

$$\pi'(x) = \exp\Big(-\frac{x^2}{2\sigma^2}\Big)$$

An example of implementation is as follows:

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

N = 10000

def pi(x):
    return 0.5*np.exp(-0.5*(x+4.)**2) + 0.5*np.exp(-0.5*(x-4.)**2)

def pi_prime(x,sigma):
    return np.exp(-0.5*(x/sigma)**2)

def ex12p3(sigma):
    y = np.random.normal(0,sigma,N)
    w = pi(y)/pi_prime(y,sigma)
    w /= w.sum()
    cdf = w.cumsum()
    omega = np.random.uniform(size=N)
    x = y[np.searchsorted(cdf, omega)]

    plt.figure()
    plt.hist(x,bins=100,density=True)
    xx = np.arange(-7,7,0.01)
    pixx = pi(xx)/2.50663
    plt.plot(xx,pixx)
    plt.xlim((-7.5,7.5))
    plt.ylim((0,0.3))

ex12p3(1)
ex12p3(2)
```

Plot with $N = 100000$ (They just look better. We can draw conclusion from $N = 10000$)



We note that there are not enough samples far away from the origin for $\sigma = 1$ case, and therefore the target distribution is not represented. We can conclude that proposal density has to cover sufficiently large area to cover the domain in order to sample from the target distribution.