homework 04 solutions

Exercise 1

Let θ_1, θ_2 be real valued parameters in $[0, 1]$ and consider the following model:

- $\theta_1 \sim \theta_1 - \text{prior}$
- $\theta_2 \sim \theta_2 - \text{prior}$
- $\hat{y} = \frac{\theta + x^2}{\theta_2 \cdot x}$
- $y \sim \mathcal{N}(\hat{y}, 1)$

a. Model implementation

```
def model(theta_prior, theta2_prior, x, obs):
    theta = pyro.sample('theta', theta_prior)
    theta2 = pyro.sample('theta2', theta2_prior)
    y_hat = (theta+x**2)/(theta2*x)
    y = pyro.sample('y', dist.Normal(y_hat,1), obs= obs)
    return y
```

b.

To choose the prior distribution of the θ parameters we follow two routes: one involving Beta distributions, the other involving Gaussian ones.

After some tests, the best parameters found are:

$$\theta_1 \sim \text{Beta}(2, 2)$$
$$\theta_2 \sim \text{Beta}(2, 4)$$

and

$$\theta_1 \sim \mathcal{N}(.3, .3^2)$$
$$\theta_2 \sim \mathcal{N}(.8, .1^2)$$

```
# We code the chosen priors
#theta_prior = dist.Beta(2,2)
# #theta2_prior = dist.Beta(2,4)
theta_prior = dist.Normal(0.3,.3)
theta2_prior = dist.Normal(.8,.1)

# x and y observed
x = torch.tensor([47,87,20,16,38,5])
y = torch.tensor([58.76,108.75,25.03,20.03,47.51,6.37])
```

Given the model, the priors and the observed values we now sample from the posterior distribution via Hamiltonian Monte Carlo.

```
nuts_kernel = NUTS(model)
n_samples = 25
warmup = 100

mcmc = MCMC(nuts_kernel,n_samples,warmup)

mcmc.run(theta_prior,theta2_prior,x,y)
```

```
Sample: 100%|██████████| 125/125 [00:05, 20.93it/s, step size=3.21e-01, acc. prob=0.973]
```

c.

We can see that the two parameter's posteriors behave very differently as the prior changes:

- θ_1 seems to be heavily influenced on the prior chosen, shifting its mean value according to the prior distribution's one;
- θ_2 is more robust, no matter what prior is chosen, resulting almost always in a distribution centered on the value 0.8.

We now focus on the samples obtained with the Gaussian priors.

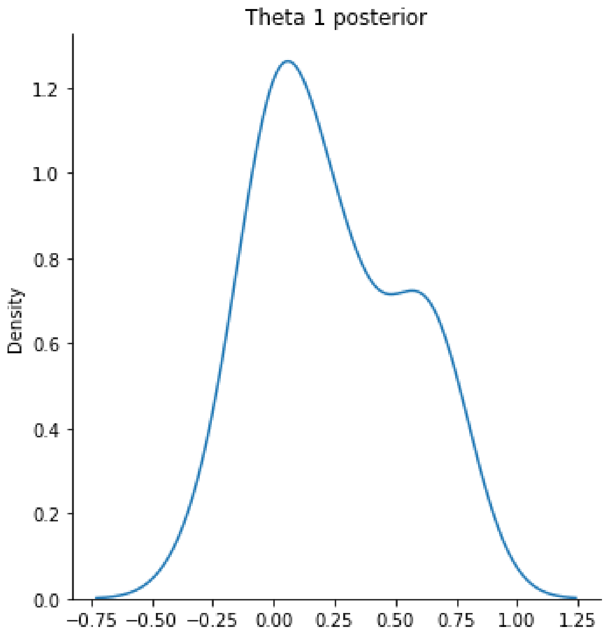
```
mcmc.summary()
```

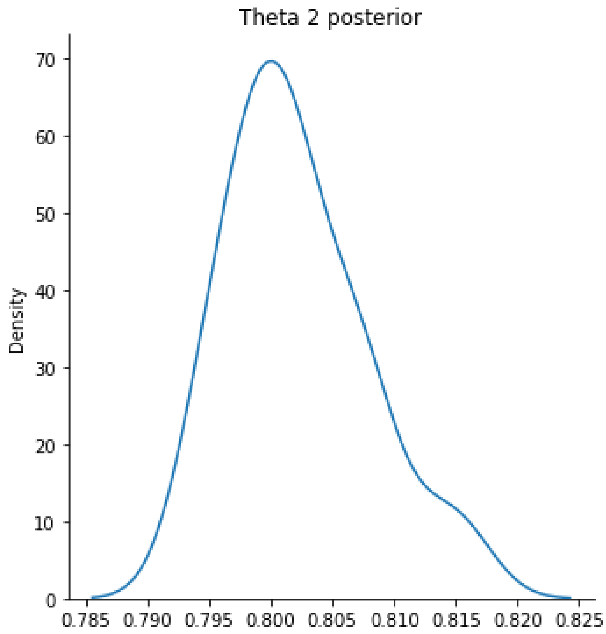
```
                mean       std    median      5.0%     95.0%     n_eff     r_hat
     theta      0.23      0.29      0.15     -0.06      0.78      7.26      1.07
    theta2      0.80      0.01      0.80      0.79      0.81     13.82      0.97

Number of divergences: 0
```

Looking at the summary of the MCMC samples and focusing on the convergence checks we can conclude that both parameters have reached the stationary distribution, given the relatively high `n_eff` and the `r_hat` index which is very close to 1 in both cases.

```
mcmc_samples = mcmc.get_samples()
sns.displot(x = mcmc_samples["theta"], kind = "kde")
plt.title("Theta 1 posterior")
sns.displot(x = mcmc_samples["theta2"], kind = "kde")
plt.title("Theta 2 posterior")
```

```
Text(0.5, 1.0, 'Theta 2 posterior')
```





Exercise 2

the property for multivariate Normal distribution are

$$x_1|x_2, x \sim N(\rho x_2, 1-\rho^2) \sim \rho x_2 + \sqrt{1-\rho^2 N(0,1)}$$

and

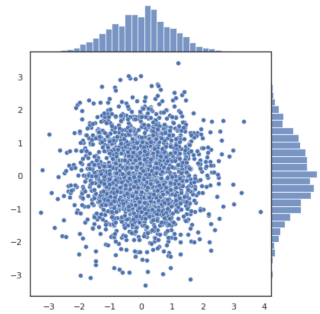
$$x_2|x_1, x \sim N(\rho x_1, 1-\rho^2) \sim \rho x_1 + \sqrt{1-\rho^2 N(0,1)}$$

then we implement the gibbs function as follow:

```python
import torch
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns
import pyro
import pyro.distributions as dist
from pyro.infer.mcmc import MCMC, HMC, NUTS
pyro.set_rng_seed(0)
```

```python
def gibbs(rho, iters, warmup):
    x1 = torch.zeros(warmup + iters, 1)
    x2 = torch.zeros(warmup + iters, 1)
    x2[0] = pyro.sample("x2", dist.Normal(0,1))
    x1[0] = pyro.sample("x1", dist.Normal(rho*x2[0].item(), np.sqrt(1-rho**2)))

    for i in range(1, warmup+iters-1):
         x2[i] = pyro.sample("x2", dist.Normal(rho*x1[i-1].item(), np.sqrt(1-rho**2)))
         x1[i] = pyro.sample("x1", dist.Normal(rho*x2[i-1].item(), np.sqrt(1-rho**2)))

    x1 = x1[warmup:iters]
    x2 = x2[warmup:iters]

    return x1, x2
```

```python
plt.figure(figsize=(12,5))
# we consider rho=0.5, iters=3000, warmup=1000
x1, x2 = gibbs(0.5, 3000, 1000)

sns.set(style="white")
sns.jointplot(x=x1.flatten(), y=x2.flatten(), space=0, color="b")

plt.xlabel("x1", fontsize=12)
plt.ylabel("x2", fontsize=12)
plt.show()
```



```python
#plot with different rho
plt.figure(figsize=(12,5))
x1, x2 = gibbs(0.99, 3000, 1000)

sns.set(style="white")
sns.jointplot(x=x1.flatten(), y=x2.flatten(), space=0, color="b")

plt.xlabel("x1", fontsize=12)
plt.ylabel("x2", fontsize=12)
plt.show()
```