DD2434 Machine Learning, Advanced Course
Assignment 1A, 2024

Yunhao Xu, Yonghong Huang

20 november 2024

1.1 Dependencies in a Directed Graphical Model

1.1.1 Answer

No

1.1.2 Answer

No

1.1.3 Answer

Yes

1.1.4 Answer

No

1.1.5 Answer

No

1.1.6 Answer

No

1.2 CAVI

1.2.9 What is the exact posterior?

Derivation of the Exact Joint Posterior Distribution for μ and τ

To derive the exact joint posterior distribution $p(\mu, \tau \mid D)$, we use Bayes' theorem along with the given prior distributions and likelihood function. The steps are as follows:

Bayes' Theorem

Bayes' theorem gives:

$$p(\mu,\tau \mid D) \propto p(D \mid \mu,\tau)p(\mu,\tau)$$

and in this case, we have:

$$p(\mu,\tau) = p(\mu \mid \tau)p(\tau)$$

hence:

$$p(\mu,\tau \mid D) \propto p(D \mid \mu,\tau)p(\mu \mid \tau)p(\tau)$$

Likelihood and Prior

The likelihood function in this case is given by:

$$p(D \mid \mu,\tau) = \left(\frac{\tau}{2\pi}\right)^{N/2} \exp\left(-\frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right)$$

The prior distributions are:

$$p(\mu \mid \tau) = \sqrt{\frac{\lambda_0\tau}{2\pi}} \exp\left(-\frac{\lambda_0\tau}{2}(\mu-\mu_0)^2\right)$$

$$p(\tau) = \frac{b_0^{a_0}}{\Gamma(a_0)}\tau^{a_0-1}\exp(-b_0\tau)$$

Combining All Terms

$$p(\mu,\tau \mid D) \propto \left(\frac{\tau}{2\pi}\right)^{N/2} \exp\left(-\frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right) \times \sqrt{\frac{\lambda_0\tau}{2\pi}} \exp\left(-\frac{\lambda_0\tau}{2}(\mu-\mu_0)^2\right) \times \frac{b_0^{a_0}}{\Gamma(a_0)}\tau^{a_0-1}\exp(-b_0\tau)$$

Taking the logarithm of the posterior and substituting into Bayes' theorem, we have:

$$\begin{aligned}
\ln p(\mu,\tau \mid D) &\stackrel{\pm}{=} \ln p(D \mid \mu,\tau) + \ln p(\mu \mid \tau) + \ln p(\tau) \\
&= \sum_{n=1}^{N} \ln p(x_n \mid \mu,\tau) + \ln p(\mu \mid \tau) + \ln p(\tau) \\
&= \sum_{n=1}^{N}[\frac{1}{2}\ln\frac{\tau}{2\pi} - \frac{\tau}{2}(x_n-\mu)^2] + \frac{1}{2}\ln\frac{\lambda_0\tau}{2\pi} - \frac{\lambda_0\tau}{2}(\mu-\mu_0)^2 + \ln\frac{b_0^{a_0}}{\Gamma(a_0)} + (a_0-1)\ln\tau - b_0\tau \\
&\stackrel{\pm}{=} \sum_{n=1}^{N}[\frac{1}{2}\ln\tau - \frac{\tau}{2}(x_n-\mu)^2] + \frac{1}{2}\ln\tau - \frac{\lambda_0\tau}{2}(\mu-\mu_0)^2 + (a_0-1)\ln\tau - b_0\tau \\
&= \ln\tau(\frac{N+1}{2} + a_0 - 1) - \frac{\tau}{2}(\sum_{n=1}^{N}(x_n-\mu)^2 + \lambda_0(\mu-\mu_0)^2) - b_0\tau
\end{aligned}$$

The term $\sum_{n=1}^{N}(x_n-\mu)^2+\lambda_0(\mu-\mu_0)^2$ can be expanded and rewritten as:

$$\sum_{n=1}^{N}(x_n-\mu)^2+\lambda_0(\mu-\mu_0)^2 = A(\mu-\mu_N)^2+B$$

where:

$$A = N+\lambda_0$$

$$\mu_N = \frac{N\bar{x}+\lambda_0\mu_0}{N+\lambda_0}$$

$$B = \sum_{n=1}^{N} x_n^2 - \frac{(N\bar{x}+\lambda_0\mu_0)^2}{N+\lambda_0}$$

Substituting back and return to the posterior, we have:

$$p(\mu,\tau \mid D) \propto \tau^{\frac{N+1}{2}+a_0-1}\exp\left(-\frac{\tau}{2}A(\mu-\mu_N)^2-\frac{\tau}{2}B-b_0\tau\right)$$

Summering

This expression above reveals that $p(\mu,\tau \mid D)$ factorizes into:

$$p(\tau \mid D) \sim \text{Gamma}\left(\frac{N+1}{2}+a_0, \frac{1}{2}B+b_0\right)$$

$$p(\mu \mid \tau, D) \sim \mathcal{N}\left(\mu_N, \frac{1}{\tau A}\right)$$

Thus, the exact joint posterior is given by:

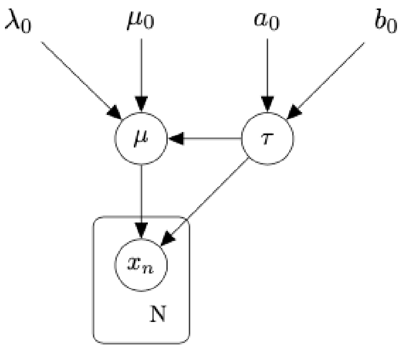
$$p(\mu,\tau \mid D) = p(\mu \mid \tau, D)\cdot p(\tau \mid D)$$

1.2.7 - 1.2.10

Other programming part will be shown in the notebook below:

Assignment 1.2 - CAVI

Consider the model defined by Equation (10.21)-(10-23) in Bishop, for which DGM is presented below:



Question 1.2.7:

Implement a function that generates data points for the given model.

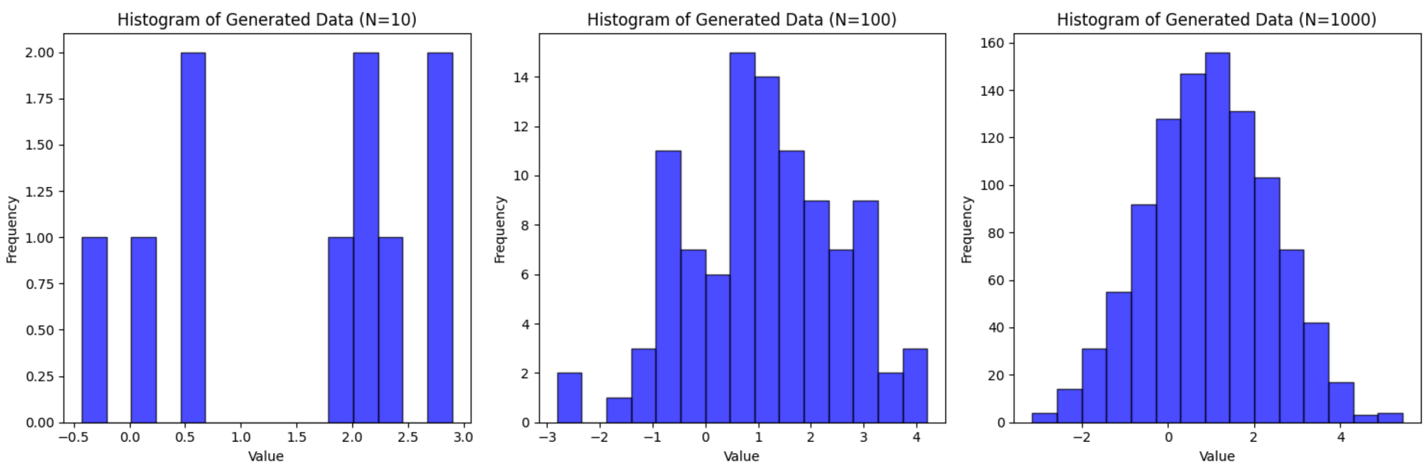
In [1]:
```
import numpy as np
def generate_data(mu, tau, N):
  # Insert your code here
  sigma = np.sqrt(1 / tau)
  D = np.random.normal(mu, sigma, N)
  return D
```

Set μ = 1, τ = 0.5 and generate datasets with size N=10,100,1000. Plot the histogram for each of 3 datasets you generated.

In [2]:
```
mu = 1
tau = 0.5

dataset_1 = generate_data(mu, tau, 10)
dataset_2 = generate_data(mu, tau, 100)
dataset_3 = generate_data(mu, tau, 1000)

# Visulaize the datasets via histograms
# Insert your code here
from matplotlib import pyplot as plt
N_values = [10, 100, 1000]
datasets = [dataset_1, dataset_2, dataset_3]
plt.figure(figsize=(15, 5))
for i in range(3):
    plt.subplot(1, 3, i + 1)
    plt.hist(datasets[i], bins=15, alpha=0.7, color='blue', edgecolor='black')
    plt.title(f'Histogram of Generated Data (N={N_values[i]})')
    plt.xlabel('Value')
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



Question 1.2.8:

Find ML estimates of the variables μ and τ

In [3]:
```
def ML_est(data):
  # insert your code
  N = len(data)
  mu_ml = np.mean(data)
```

```
    tau_ml = N / np.sum((data - mu_ml) ** 2)
    return mu_ml, tau_ml
```

Question 1.2.9:

What is the exact posterior? First derive it in closed form, and then implement a function that computes it for the given parameters:

In [4]:
```python
def compute_exact_posterior(D, a_0, b_0, mu_0, lambda_0):
    N = len(D)
    x_mean = np.mean(D)
    x_var = np.sum((D - x_mean)**2)

    mu_N = (lambda_0 * mu_0 + N * x_mean) / (lambda_0 + N)
    a_N = a_0 + N / 2
    b_N = b_0 + 0.5 * x_var + 0.5 * lambda_0 * (mu_N - mu_0)**2
    lambda_N = lambda_0 + N * (a_N / b_N)

    exact_post_dist_parameters = {'a_N': a_N, 'b_N': b_N, 'mu_N': mu_N, 'lambda_N': lambda_N}
    return exact_post_dist_parameters
```

Question 1.2.10:

You will implement the VI algorithm for the variational distribution in Equation (10.24) in Bishop. Start with introducing the prior parameters:

In [5]:
```python
# prior parameters
mu_0 = 0
lambda_0 = 1
a_0 = 2
b_0 = 1
```

Continue with a helper function that computes ELBO:

In [6]:
```python
from scipy.special import psi, gammaln
def compute_elbo(D, a_0, b_0, mu_0, lambda_0, a_N, b_N, mu_N, lambda_N):
    """
    ELBO = E_q[log p(D | mu, tau)] + E_q[log p(mu)] + E_q[log p(tau)] - E_q[log q(mu)] - E_q[log q(tau)]
    """
    N = len(D)

    E_log_tau = psi(a_N) - np.log(b_N)
    E_tau = a_N / b_N
    E_mu = mu_N
    Var_mu = 1 / lambda_N

    # E_q[log p(D | mu, tau)]
    sq_diff = (D - E_mu) ** 2
    E_xn_mu_sq = Var_mu + sq_diff
    sum_E_xn_mu_sq = np.sum(E_xn_mu_sq)
    E_log_p_D_given_mu_tau = (
        N * 0.5 * (E_log_tau - np.log(2 * np.pi))
        - 0.5 * E_tau * sum_E_xn_mu_sq
    )

    # E_q[log p(mu)]
    E_mu_mu0_sq = Var_mu + (E_mu - mu_0) ** 2
    E_log_p_mu = (
        -0.5 * np.log(2 * np.pi / lambda_0)
        - 0.5 * lambda_0 * E_mu_mu0_sq
    )

    # E_q[log p(tau)]
    E_log_p_tau = (
        (a_0 - 1) * E_log_tau
        - b_0 * E_tau
        - gammaln(a_0)
        + a_0 * np.log(b_0)
    )

    # E_q[log q(mu)]
    E_log_q_mu = (
        -0.5 * np.log(2 * np.pi / lambda_N)
        - 0.5
    )

    # E_q[log q(tau)]
    E_log_q_tau = (
        (a_N - 1) * E_log_tau
        - b_N * E_tau
        - gammaln(a_N)
        + a_N * np.log(b_N)
    )

    # ELBO
    elbo = (
        E_log_p_D_given_mu_tau
        + E_log_p_mu
        + E_log_p_tau
        - E_log_q_mu
        - E_log_q_tau
```

```
    )

    return elbo
```

Now, implement the CAVI algorithm:

In [7]:
```
def CAVI(D, a_0, b_0, mu_0, lambda_0, max_iter=100):
    # make an initial guess for the expected value of tau
    initial_guess_exp_tau = a_0 / b_0

    # CAVI iterations ...
    # save ELBO for each iteration, plot them afterwards to show convergence

    N = len(D)
    elbos = []
    exp_tau = initial_guess_exp_tau
    a_N, b_N, mu_N, lambda_N = a_0, b_0, mu_0, lambda_0

    for _ in range(max_iter):
        lambda_N = lambda_0 + N * exp_tau
        mu_N = (lambda_0 * mu_0 + exp_tau * np.sum(D)) / lambda_N

        a_N = a_0 + N / 2
        b_N = b_0 + 0.5 * (np.sum((D - mu_N) ** 2) + N / lambda_N)

        exp_tau = a_N / b_N

        elbo = compute_elbo(D, a_0, b_0, mu_0, lambda_0, a_N, b_N, mu_N, lambda_N)
        elbos.append(elbo)

    return a_N, b_N, mu_N, lambda_N, elbos
```

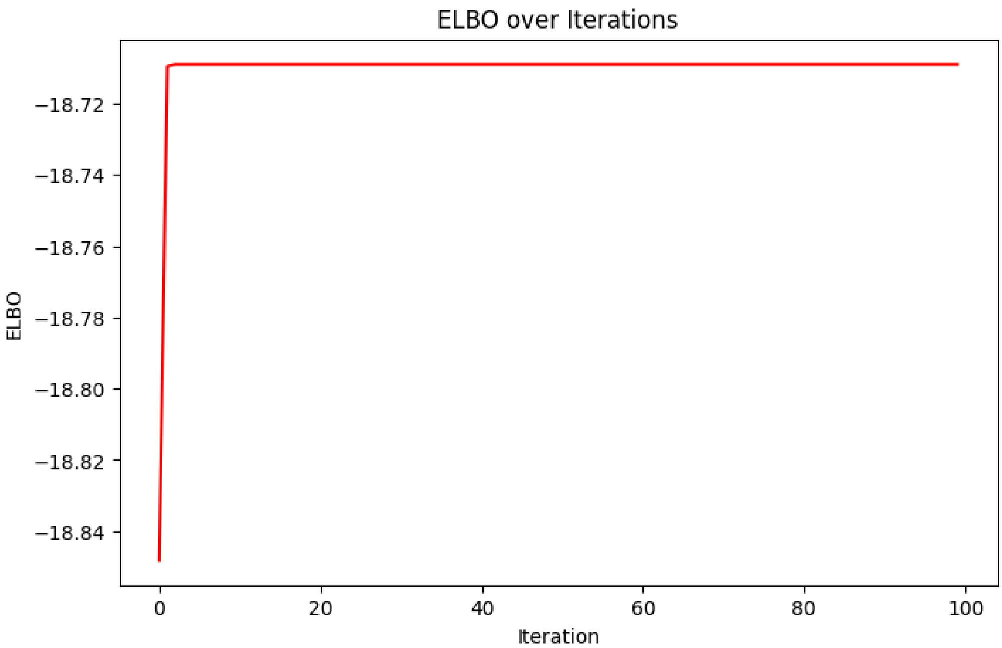
In [8]:
```
def plot_elbo(elbo_values):
    plt.figure(figsize=(8, 5))
    plt.plot(elbo_values, color='red')
    plt.title('ELBO over Iterations')
    plt.xlabel('Iteration')
    plt.ylabel('ELBO')
    plt.show()
```

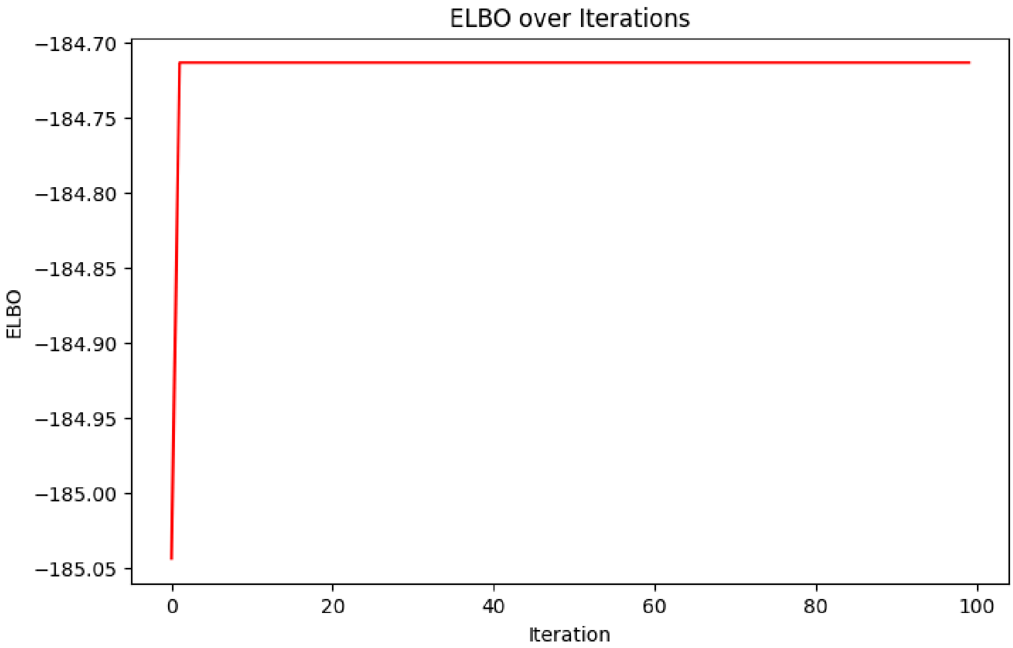
Run the VI algorithm on the datasets. Compare the inferred variational distribution with the exact posterior and the ML estimate. Visualize the results and discuss your findings.

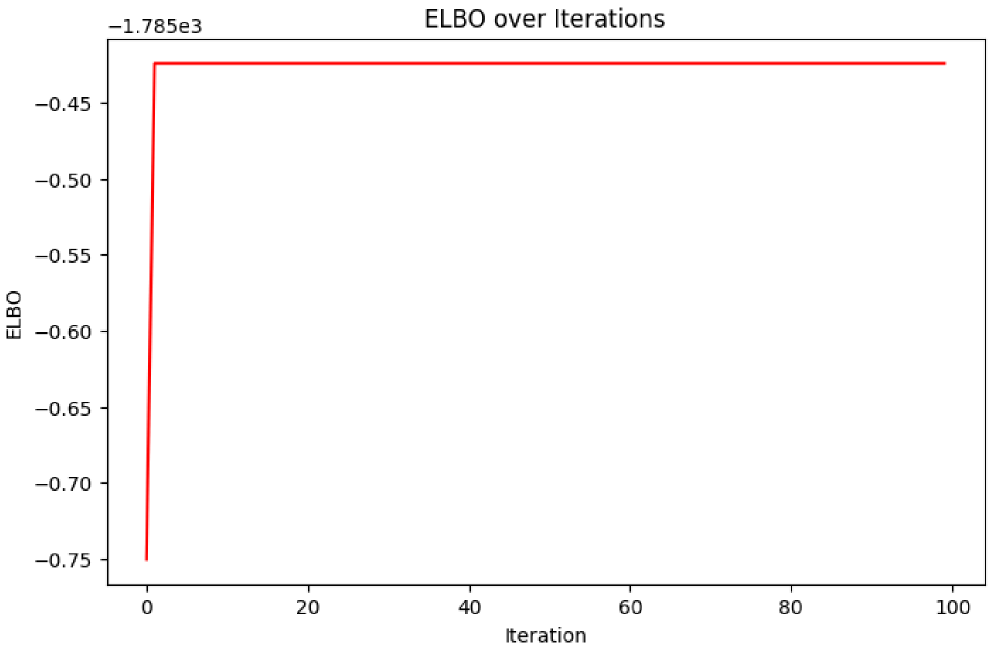
In [9]:
```
# Insert your main code here

def test(dataset):
    mu_ml, tau_ml = ML_est(dataset)
    a_N, b_N, mu_N, lambda_N, elbos = CAVI(dataset, a_0, b_0, mu_0, lambda_0)
    plot_elbo(elbos)
    exact_post_dist_param = compute_exact_posterior(dataset, a_0, b_0, mu_0, lambda_0)
    cavi_post_dist_param = {'a_N': a_N, 'b_N': b_N, 'mu_N': mu_N, 'lambda_N': lambda_N}
    ml_e_dist_param = {'mu_ml': mu_ml, 'tau_ml': tau_ml}
    return exact_post_dist_param, cavi_post_dist_param, ml_e_dist_param

test_results = [(test(dataset), dataset) for dataset in [dataset_1, dataset_2, dataset_3]]
```





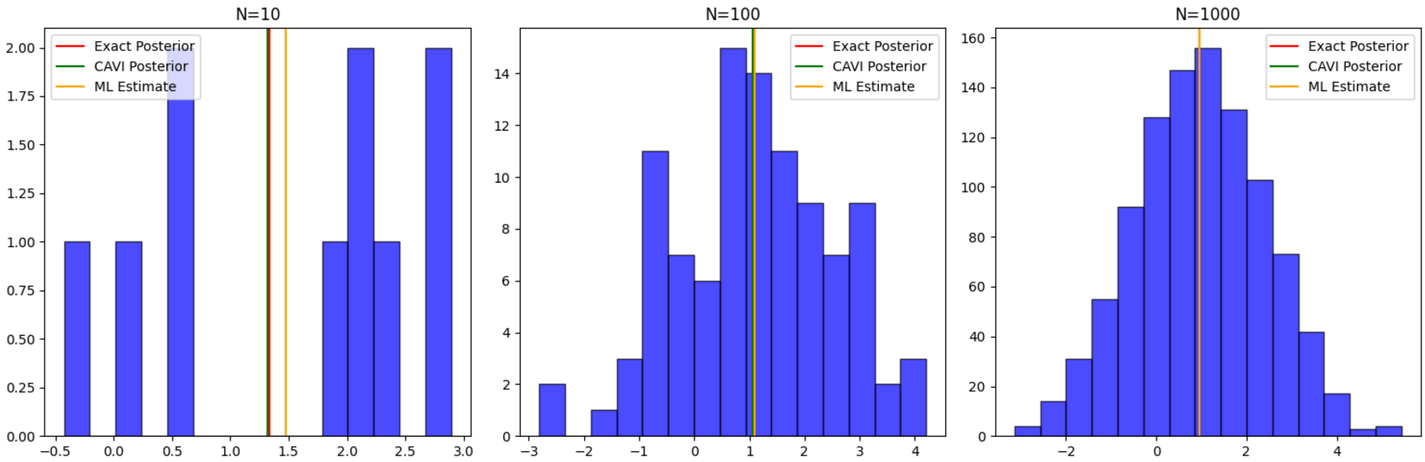


In [10]:
```
def plot_results(test_results):
    plt.figure(figsize=(15, 5))
    for i, ((exact_post_dist_param, cavi_post_dist_param, ml_e_dist_param), dataset) in enumerate(test_results):
        plt.subplot(1, 3, i + 1)
        plt.hist(dataset, bins=15, alpha=0.7, color='blue', edgecolor='black')
        plt.axvline(exact_post_dist_param['mu_N'], color='red', label='Exact Posterior')
        plt.axvline(cavi_post_dist_param['mu_N'], color='green', label='CAVI Posterior')
        plt.axvline(ml_e_dist_param['mu_ml'], color='orange', label='ML Estimate')
        plt.title(f'N={len(dataset)}')
        plt.legend()
    plt.tight_layout()
    plt.show()

plot_results(test_results)

for result in test_results:
    print(f"Results for dataset with N = {len(result[1])}")
```

```python
    print(f"Exact Posterior: {result[0][0]}")
    print(f"CAVI Posterior: {result[0][1]}")
    print(f"ML Estimate: {result[0][2]}")
    print("\n")
```




```
Results for dataset with N = 10
Exact Posterior: {'a_N': 7.0, 'b_N': 8.246056803736828, 'mu_N': 1.3380064318896074, 'lambda_N': 9.488905869321496}
CAVI Posterior: {'a_N': 7.0, 'b_N': 7.975661001057302, 'mu_N': 1.3212647886896758, 'lambda_N': 9.776702017640968}
ML Estimate: {'mu_ml': 1.4718070750785681, 'tau_ml': 0.7872867427894785}


Results for dataset with N = 100
Exact Posterior: {'a_N': 52.0, 'b_N': 106.98437029344821, 'mu_N': 1.0780353751321907, 'lambda_N': 49.60523070554027}
CAVI Posterior: {'a_N': 52.0, 'b_N': 107.43974245493843, 'mu_N': 1.0667745777506275, 'lambda_N': 49.39922249609771}
ML Estimate: {'mu_ml': 1.0888157288835125, 'tau_ml': 0.4743684938567458}


Results for dataset with N = 1000
Exact Posterior: {'a_N': 502.0, 'b_N': 1026.2445821127876, 'mu_N': 0.9636591312475273, 'lambda_N': 490.16214394672295}
CAVI Posterior: {'a_N': 502.0, 'b_N': 1026.802825569814, 'mu_N': 0.962653755326952, 'lambda_N': 489.89620041843966}
ML Estimate: {'mu_ml': 0.9646227903787749, 'tau_ml': 0.48790947505759086}
```