DD2434, Assignment 1A, 2024

Zyad Haddad, Erik Wallinder-Mähler

November 20, 2024

1 Assignment 1A

1.1 Dependencies in a Directed Graphical Model

1.1.1 $w_{n,g} \perp w_{n,g+1} | s_n$?

$w_{n,g} \perp w_{n,g+1} | s_n$ is **false** ($w_{n,g} \leftarrow z_n \rightarrow w_{n,g+1}$, z_n is not observed).

1.1.2 $l_n \perp w_{n,g} | x_{n,g}$?

$l_n \perp w_{n,g} | x_{n,g}$ is **false**. ($x_{n,g}$ is a descendant of $y_{n,g}$ which means that the V-structure does not give independence.)

1.1.3 $z_n \perp x_{n,g} | w_{n,g}, h_{n,g}$?

This is **true**.
(While $w_{n,g}$ and $h_{n,g}$ are observed, for example $w_{n,g+1}$ and $h_{n,g+1}$ are not. These do not have paths to $x_{n,g}$ however. There is a path from $w_{n,g+1}$ to $x_{n,g}$ through l_n, but no descendants of l_n are observed, so there is a independence-giving V-structure there.)

1.1.4 $z_1^n \perp z_M^n | C_n, A_{1:I,1:J}^{1:K}$?

This is **false**. (There is a clear path $z_1^n \rightarrow z_2^n \rightarrow \cdots \rightarrow z_{M-1}^n \rightarrow z_M^n$ that does not cross any observed variables.)

1.1.5 $X_1^n \perp X_M^n | X_2^n, C^n$?

This is **true**.
(Because X_2^n is observed, X_1^n can no longer influence X_M^n through $e_{i,r}^k$. Likewise, the paths through z_1^n run into this same issue. It also clearly cannot influence through C^n, due to it being observed.)

1.1.6 $C^n \perp C^{n+1} | z_{1:M}^n, X_{1:M}^n$?

This is **false**. (Both C^n and C^{n+1} depend on π, which is not observed.)

1.2 CAVI

1.2.7 Implement a function that generates data points for the given model. Set $\mu = 1$, $\tau = 0.5$ and generate datasets with size $N = 10, 100, 1000$. Plot the histogram for each of 3 datasets you generated.

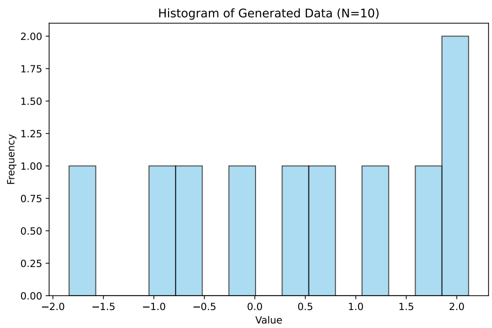


Figure 1: Histogram of data points for $N = 10$

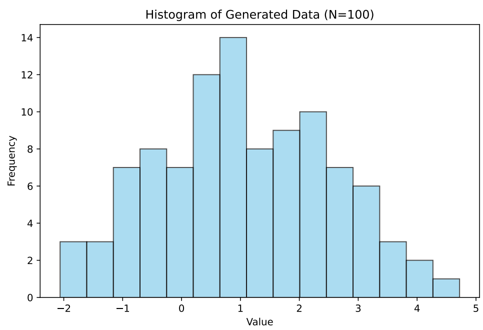


Figure 2: Histogram of data points for $N = 100$

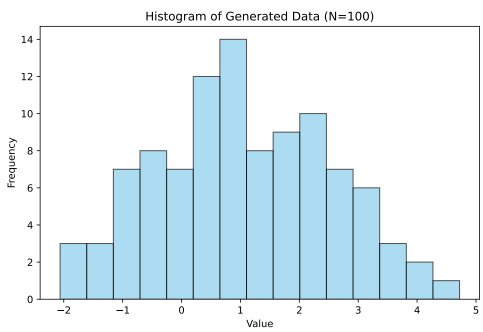


Figure 3: Histogram of data points for $N = 100$

The function is implemented as `generate_data`.

1.2.8 Find ML estimates of the variables μ and τ.

The ML estimates for μ and τ are the mean of the data, and the inverse variance of the data respectively. ML_est was implemented to return these values.

1.2.9 What is the exact posterior? (Show your derivations.)

To derive the posterior, we first begin by defining the likelihood and prior:

$$\text{Likelihood: } p(X|\mu,\tau) = (\frac{\tau}{2\pi})^{N/2} \exp\{-\frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2\} \text{ - As defined in Bishop 10.21} \tag{1}$$

With the given conjugate prior distributions for μ and τ in Bishop 10.22 and 10.23:

$$\begin{aligned} p(\mu|\tau) = \mathcal{N}(\mu|\mu_0,(\lambda_0\tau)^{-1}) &= \frac{\sqrt{\tau\lambda_0}}{\sqrt{2\pi}}\exp\{-\frac{\lambda_0\tau(x-\mu)}{2}\} \\ p(\tau) = \text{Gam}(\tau|a_0,b_0) &= \frac{b_0^{a_0}}{\Gamma(a_0)}\tau^{a_0-1}\exp\{-b_0\tau\} \end{aligned} \tag{2}$$

The posterior is defined through Bayes' theorem, as:

$$p(\mu,\tau|X) \propto p(X|\mu,\tau)p(\mu|\tau)p(\tau), \tag{3}$$

Combining the terms results in the expression:

$$p(\mu,\tau|X) = (\frac{\tau}{2\pi})^{N/2}\exp\{-\frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2\} \times \frac{\sqrt{\tau\lambda_0}}{\sqrt{2\pi}}\exp\{-\frac{\lambda_0\tau(\mu_0-\mu)^2}{2}\} \times \frac{b_0^{a_0}}{\Gamma(a_0)}\tau^{a_0-1}\exp\{-b_0\tau\} \tag{4}$$

Splitting the terms by μ and τ:

$$= (\frac{\tau}{2\pi})^{N/2}\exp\{-\tau/2\}\exp\{\sum_{n=1}^{N}(x_n-\mu)^2\}\frac{\sqrt{\tau\lambda_0}}{\sqrt{2\pi}}\exp\{\tau\}\exp\{-\frac{\lambda_0}{2}(\mu-\mu_0)^2\}\frac{b_0^{a_0}}{\Gamma(a_0)}\tau^{a_0-1}\exp\{-b_0\tau\} \tag{5}$$

The sum and the squared term can be expanded as:

$$\begin{aligned} \sum_{n=1}^{N}(x_n-\mu)^2 &= \sum_{n=1}^{N}x_n^2 - 2\mu\sum_{n=1}^{N}x_n + N\mu^2 \\ \lambda_0(\mu_0-\mu)^2 &= \lambda_0\mu_0^2 - 2\lambda_0\mu_0\mu + \lambda_0\mu^2 \end{aligned} \tag{6}$$

Together the terms create

$$\begin{aligned} \sum_{n=1}^{N}(x_n-\mu)^2 + \lambda_0(\mu_0-\mu)^2 &= (N+\lambda_0)\mu^2 - 2(\sum_{n=1}^{N}x_n + \lambda_0\mu_0)\mu + \sum_{n=1}^{N}x_n^2 + \lambda_0\mu_0^2 \\ &= (N+\lambda_0)(\mu^2 - \frac{\sum_{n=1}^{N}x_n + \lambda_0\mu_0}{N+\lambda_0} + \frac{\sum_{n=1}^{N}x_n^2 + \lambda_0\mu_0^2}{N+\lambda_0}) \end{aligned} \tag{7}$$

We want to be able to express these sums in a more suitable form. This can be done by completing the square $(\mu-\mu^*)$:

$$\begin{aligned} \text{We define } \mu^* \text{ as: } \mu^* &= \frac{\sum_{n=1}^{N}x_n + \lambda_0\mu_0}{N+\lambda_0} \\ (\text{Eq 7 with } N+\lambda_0 \text{ factored out}) = \mu^2 - 2\mu^*\mu + \frac{\sum_{n=1}^{N}x_n^2 + \lambda_0\mu_0^2}{N+\lambda_0} &= \\ \mu^2 - 2\mu^*\mu + (\mu^*)^2 - (\mu^*)^2 + \frac{\sum_{n=1}^{N}x_n^2 + \lambda_0\mu_0^2}{N+\lambda_0} &= \\ (\mu-\mu^*)^2 - (\mu^*)^2 + \frac{\sum_{n=1}^{N}x_n^2 + \lambda_0\mu_0^2}{N+\lambda_0} &= \\ (\mu-\mu^*)^2 + \frac{(N+\lambda_0)\left(\sum_{n=1}^{N}x_n^2 + \lambda_0\mu_0^2\right) - \left(\sum_{n=1}^{N}x_n + \lambda_0\mu_0\right)^2}{(N+\lambda_0)^2} \end{aligned} \tag{8}$$

In equation 5, we now have:

$$
\begin{aligned}
(\frac{\tau}{2\pi})^{N/2}\exp\{-\tau/2\}\exp\{\sum_{n=1}^{N}(x_n-\mu)^2\}\frac{\sqrt{\tau\lambda_0}}{\sqrt{2\pi}}\exp\{\tau\}\exp\{-\frac{\lambda_0}{2}(\mu-\mu_0)^2\}\frac{b_0^{a_0}}{\Gamma(a_0)}\tau^{a_0-1}\exp\{-b_0\tau\} = \\
(\frac{\tau}{2\pi})^{N/2}\exp\{-\tau/2\}\frac{\sqrt{\tau\lambda_0}}{\sqrt{2\pi}}\exp\{\tau\}\exp\{(N+\lambda_0)(\mu-\mu^*)^2\} \\
\exp\{\sum_{n=1}^{N}x_n^2+\lambda_0\mu_0^2)\}\exp\{-\frac{\left(\sum_{n=1}^{N}x_n+\lambda_0\mu_0\right)^2}{N+\lambda_0}\} \\
\frac{b_0^{a_0}}{\Gamma(a_0)}\tau^{a_0-1}\exp\{-b_0\tau\}
\end{aligned}
\tag{9}
$$

Through combining these terms, we identify:

$$
\begin{aligned}
\mu^* &= \frac{\sum_{n=1}^{N}x_n+\lambda_0\mu_0}{N+\lambda_0} \\
\lambda^* &= N+\lambda_0 \\
a_N &= a_0+\frac{N}{2} \\
b_N &= b_0+\frac{1}{2}(\sum_{n=1}^{N}x_n^2+\lambda_0\mu_0^2-\lambda^*\mu^{*2})
\end{aligned}
\tag{10}
$$

1.2.10 Implement the VI algorithm for the variational distribution in Equation (10.24) in Bishop. Run the VI algorithm on the datasets. Plot the ELBO results. Compare the inferred variational distribution with the exact posterior and the ML estimate. Visualize the results and discuss your findings.

CAVI is implemented in the notebook, based on the factorized variational approximation from Bishop. This gives the following posterior distributions:

$$\mu_N = \frac{\lambda_0\mu_0+N\bar{x}}{\lambda_0+N} \tag{11}$$

$$\lambda_N = (\lambda_0+N)\mathbb{E}[\tau],\ \text{where } \mathbb{E}[\tau] \text{ is estimated as } \frac{a_N}{b_N} \tag{12}$$

$$a_N = a_0+\frac{N}{2} \tag{13}$$

$$b_N = b_0+\frac{1}{2}\mathbb{E}_\mu\left[\sum_{n=1}^{N}(x_n-\mu)^2+\lambda_0(\mu-\mu_0)^2\right] \tag{14}$$

Iteration with CAVI and plotting the ELBO:s gave these results over the datasets:

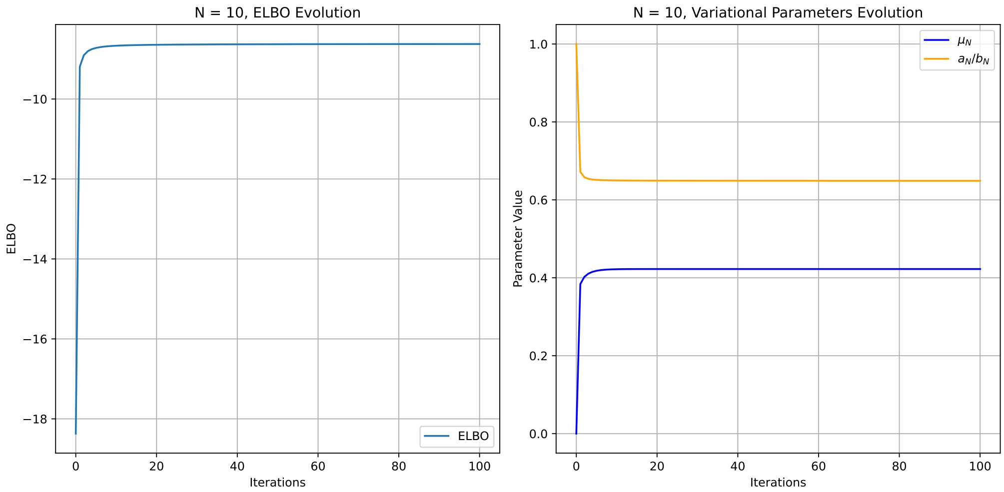


Figure 4: ELBO results and μ_N, τ_N for the dataset of size 10

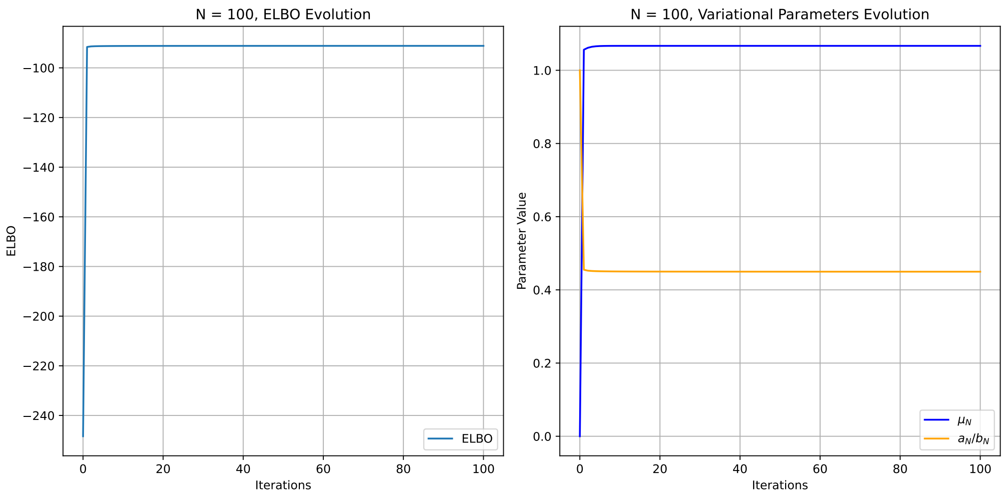


Figure 5: ELBO results and μ_N, τ_N for the dataset of size 100

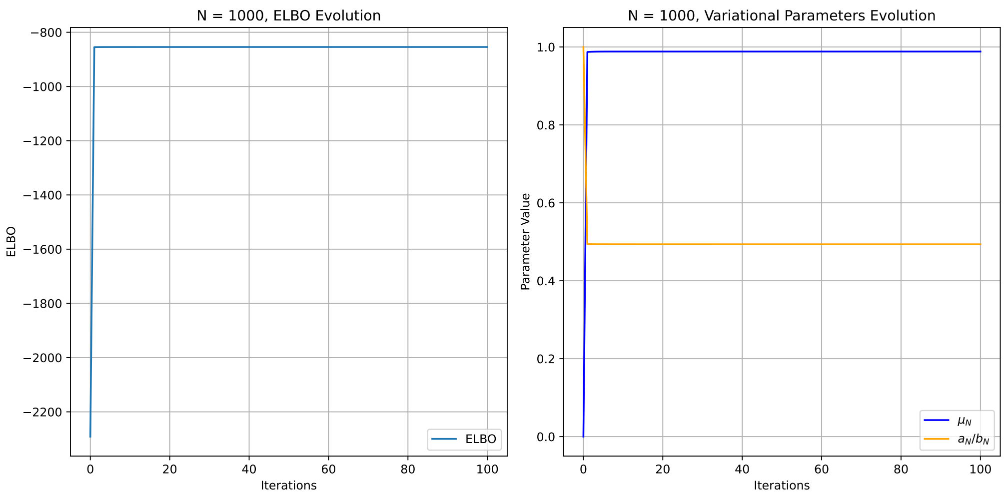


Figure 6: ELBO results and μ_N, τ_N for the dataset of size 1000

Comparing the inferred values μ, τ with the ML estimate and the exact posterior for these datasets:

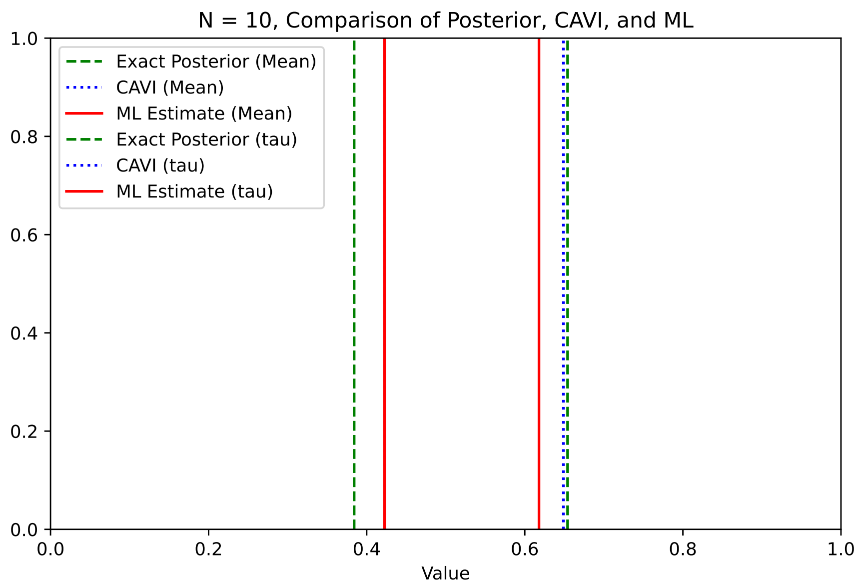


Figure 7: Comparison over the inferred parameters

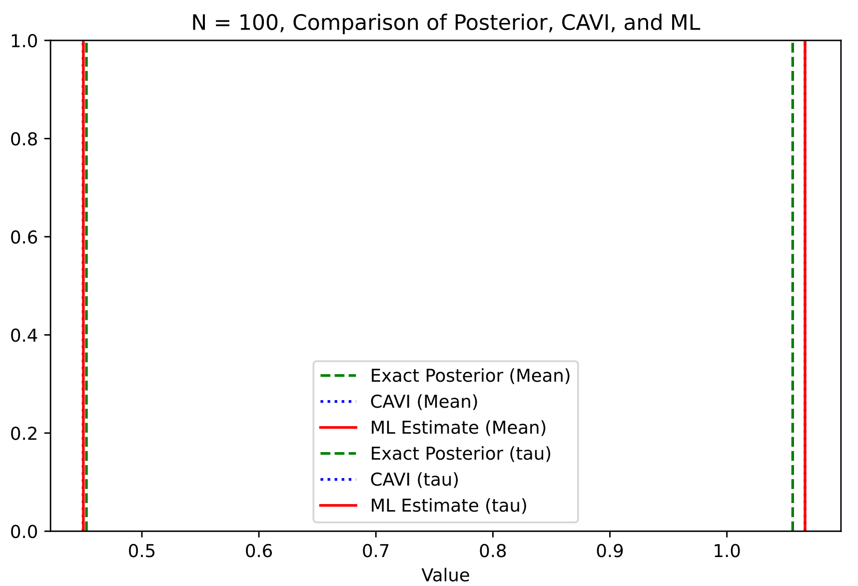


Figure 8: Comparison over the inferred parameters

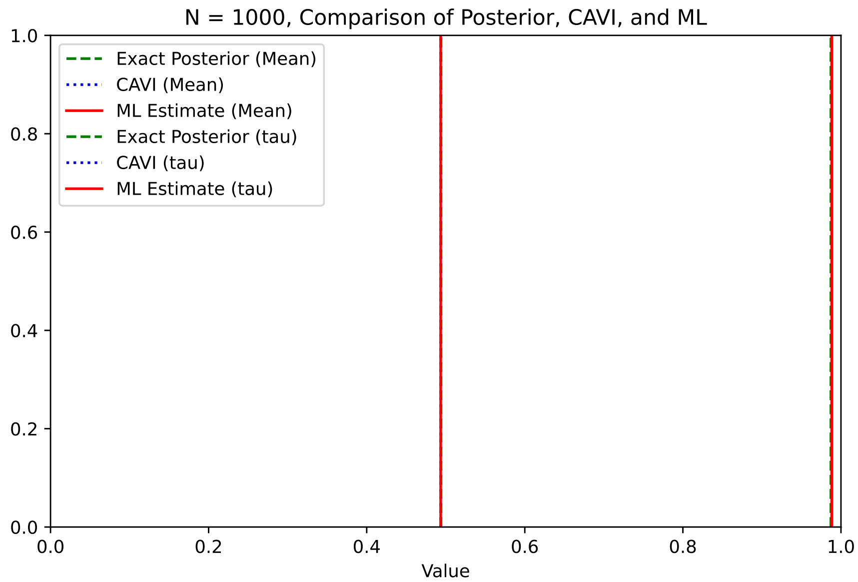


Figure 9: Comparison over the inferred parameters

Analysis

The CAVI algorithm converges very quickly for these datasets, and produces results that are comparable to or closer to the exact posterior compared with the ML estimate, even for the small datasets. Part of this is due to the simple distribution that generated the data, which is just a normal distribution. For the very small datasets, individual points have a large effect, and all methods have a large difference compared with the parameters used to generate the data $\mu = 1, \tau = 0.5$. This difference, as expected, becomes much smaller when the size of the generated data increases, and the parameters converge towards the ones used to generate the datasets.

Notebook

November 20, 2024

0.1 Run instructions:

Tested in python 3.12.7, within a .conda environment.

(Since the notebook is to be exported as html, requirements.txt is not included, so the dependencies can be installed through: `pip install numpy scikit-learn matplotlib pandas`)

Figures are configured to be stored in a folder `figures` within the same directory as the notebook. Make sure there is a folder there if you want to run the notebook.

1 *Assignment 1.2 - CAVI*

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

1.0.1 *Question 1.2.7:*

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

[21]:
```python
# Imports
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.special import psi, gammaln
```

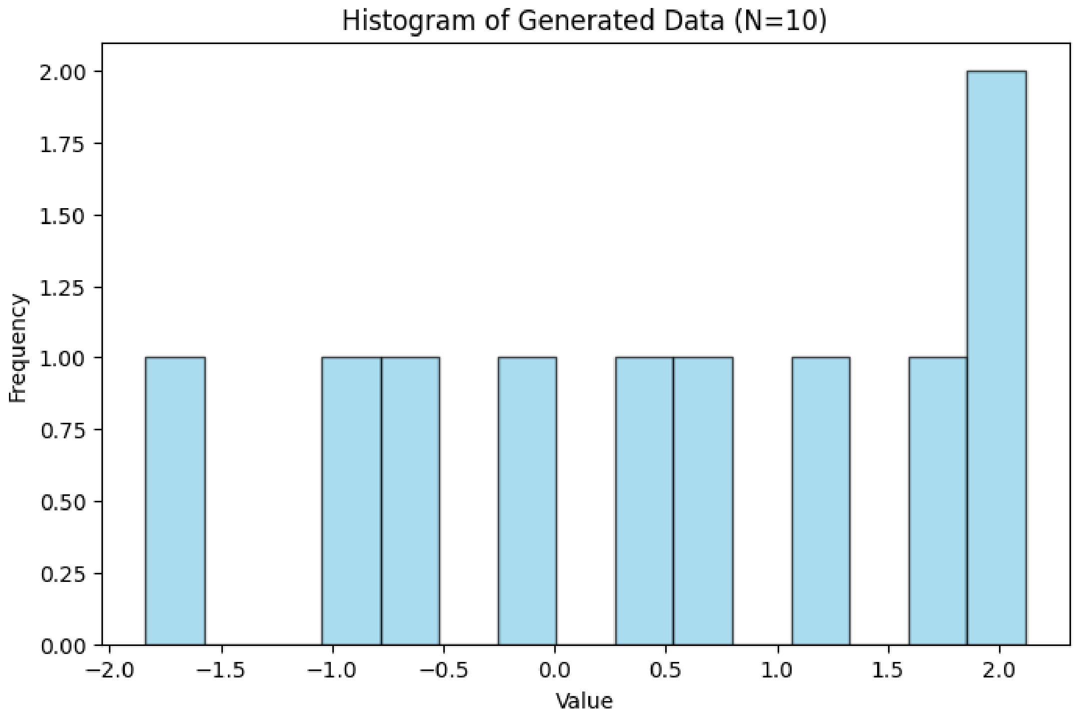
[22]:
```python
def generate_data(mu, tau, size):
    variance = 1 / tau
    return np.random.normal(mu, np.sqrt(variance), size)
```

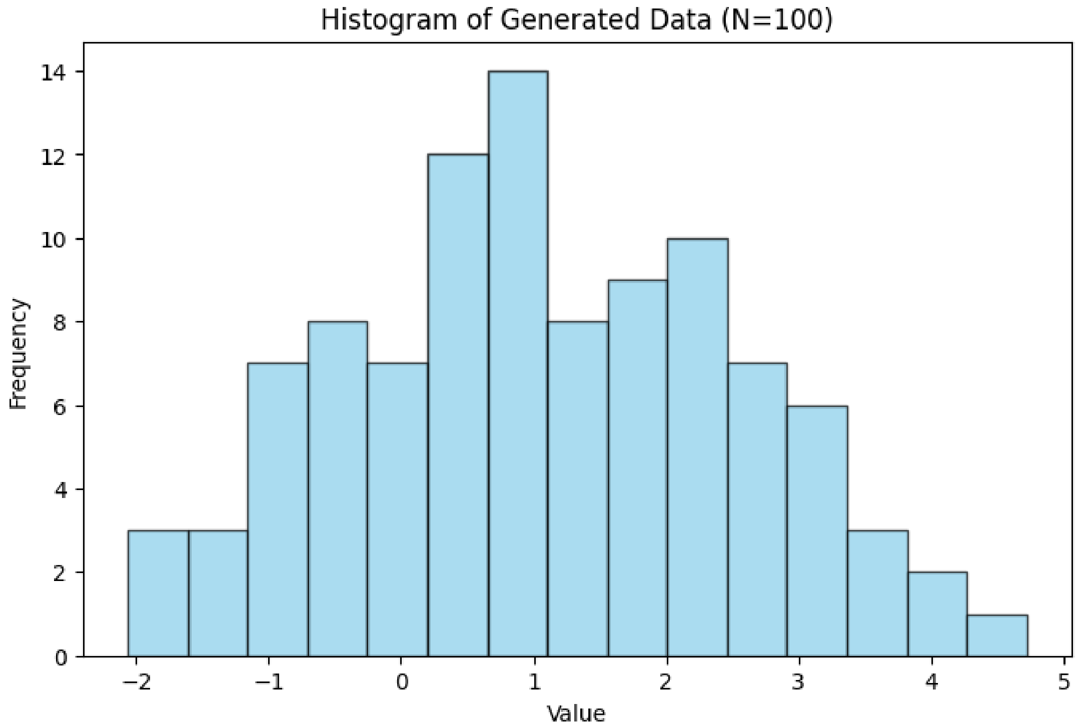
Set = 1, = 0.5 and generate datasets with size N=10,100,1000. Plot the histogram for each of 3 datasets you generated.

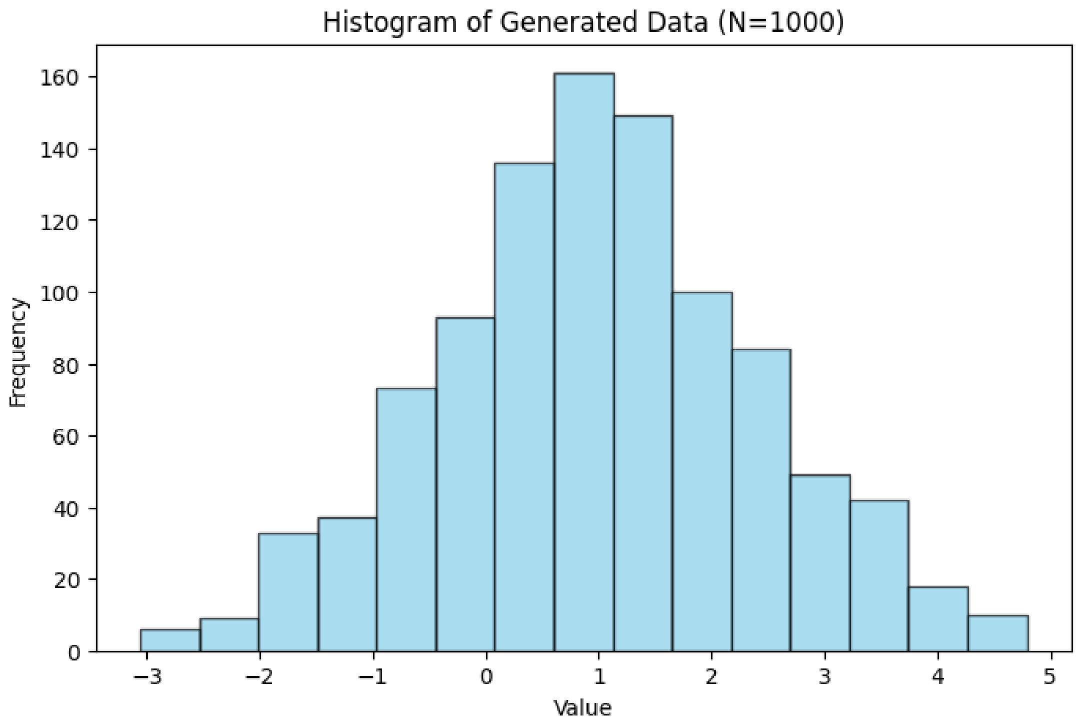
[23]:
```python
mu = 1
tau = 0.5
sizes = [10, 100, 1000]

datasets = {size: generate_data(mu, tau, size) for size in sizes}
```

```python
# Visulaize the datasets via histograms
for size, data in datasets.items():
    plt.figure(figsize=(8, 5))
    plt.hist(data, bins=15, color='skyblue', alpha=0.7, edgecolor='black')
    plt.title(f"Histogram of Generated Data (N={size})")
    plt.xlabel("Value")
    plt.ylabel("Frequency")
    plt.savefig(f"figures/{size}_histogram.pdf")
    plt.show()
```




Histogram of Generated Data (N=100)
Frequency
Value
0
2
4
6
8
10
12
14
−2
−1
0
1
2
3
4
5


Histogram of Generated Data (N=1000)
Frequency
Value
0
20
40
60
80
100
120
140
160
−3
−2
−1
0
1
2
3
4
5

1.0.2 *Question 1.2.8:*

Find ML estimates of the variables   and

[24]:
```python
def ML_est(data):
  mu_ml = np.mean(data)
  tau_ml = 1 / np.var(data)
  return mu_ml, tau_ml

ml_estimates = {size: ML_est(data) for size, data in datasets.items()}

ml_results = pd.DataFrame.from_dict(
    ml_estimates, orient='index', columns=['ML Estimate of  ', 'ML Estimate of 
 ']
)

print(ml_results)
```

```
      ML Estimate of    ML Estimate of
10           0.422523          0.617946
100          1.066792          0.449982
1000         0.988250          0.493600
```

1.0.3 *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:

[25]:
```python
def compute_exact_posterior(D, a_0, b_0, mu_0, lambda_0):
  N = len(D)
  x_mean = np.mean(D)
  lambda_N = lambda_0 + N
  a_N = a_0 + N / 2
  mu_N = (lambda_0 * mu_0 + np.sum(D)) / (lambda_0 + N)
  # Same expressions from derivation and from wikipedia - exact same value 
returned
  b_N = b_0 + 1/2 * (np.sum(np.power(D,2)) + lambda_0*mu_0**2 - 
lambda_N*mu_N**2)
  #b_N = b_0 + 1/2 * np.sum(np.power(D-x_mean,2)) + N*lambda_0/(N+lambda_0) * 
((x_mean - mu_0)**2)/2

  return {"a_N": a_N, "b_N": b_N, "mu_N": mu_N, "lambda_N": lambda_N}


print(compute_exact_posterior(datasets[1000],0,1,1,0.5))
```

```
{'a_N': 500.0, 'b_N': np.float64(1013.9656178720086), 'mu_N':
np.float64(0.988255621122432), 'lambda_N': 1000.5}
```

1.0.4 *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:

[26]:
```
# prior parameters
mu_0 = 0
lambda_0 = 1
a_0 = 1
b_0 = 1
```

Continue with a helper function that computes ELBO:

[27]:
```
def elbo(x, mu_N, lambda_N, a_N, b_N):

    N = len(x)

    # Expectations for tau
    E_log_tau = psi(a_N) - np.log(b_N)
    E_tau = a_N / b_N

    # Likelihood term
    E_log_p_x_given_mu_tau = (N / 2) * E_log_tau - 0.5 * E_tau * np.sum(x**2 -
    ↪2 * x * mu_N + mu_N**2 + 1 / lambda_N)

    # Prior on mu
    E_log_p_mu_given_tau = -0.5 * E_tau * ((mu_N - mu_0)**2 + 1 / lambda_0)

    # Prior on tau
    E_log_p_tau = (a_0 - 1) * E_log_tau - b_0 * E_tau - gammaln(a_0) + a_0 * np.
    ↪log(b_0)

    # ELBO
    elbo_val = E_log_p_x_given_mu_tau + E_log_p_mu_given_tau + E_log_p_tau

    return elbo_val
```

Now, implement the CAVI algorithm:

[28]:
```
def CAVI(x, mu_0, lambda_0, a_0, b_0):

    # Helpers for calculating updates:
    def update_mu(x, mu_0, lambda_0, E_tau):
        # Update mu, tau from Bishop (10.26, 10.27)
        N = len(x)
        lambda_N = (lambda_0 + N) * E_tau
```

```
        mu_N = (N*np.mean(x) + lambda_0 * mu_0) / (lambda_0 + N)
        return mu_N, lambda_N

    def update_tau(x, mu_N, a_0, b_0, lambda_0, lambda_N):
        # Update a_N, b_N from the approximations in Bishop (10.29, 10.30)

        N = len(x)
        a_N = a_0 + (N + 1) / 2

        # b_N:s expected mu values
        expected_x_mu_squared = np.sum(x**2 - 2 * x * mu_N + (1 / lambda_N) + mu_N**2)
        expected_mu_mu0_squared = (1 / lambda_N) + mu_N**2 - 2 * mu_0 * mu_N + mu_0**2

        b_N = b_0 + 0.5 * (expected_x_mu_squared + lambda_0 * expected_mu_mu0_squared)

        return a_N, b_N

    # Parameter priors
    mu_N = mu_0
    lambda_N = lambda_0
    a_N = a_0
    b_N = b_0

    elbo_history = [elbo(x, mu_N, lambda_N, a_N, b_N)]
    mu_history = [mu_N]
    lambdas_history = [lambda_N]
    estimated_taus_history = [a_N/b_N]

    prev_elbo = -np.inf
    for i in range(100):
        # Update mu and lambda
        E_tau = a_N/b_N

        mu_N, _ = update_mu(x, mu_N, lambda_N, E_tau)
        _, lambda_N = update_mu(x, mu_N, lambda_N, E_tau)

        a_N, _ = update_tau(x, mu_N, a_N, b_N, lambda_0, lambda_N)
        _, b_N = update_tau(x, mu_N, a_N, b_N, lambda_0, lambda_N)

        elbo_value = elbo(x, mu_N, lambda_N, a_N, b_N)

        elbo_history.append(elbo_value)
        mu_history.append(mu_N)
        lambdas_history.append(lambda_N)
```

```
        estimated_taus_history.append(a_N/b_N)

        print(f"Iteration {i+1}: ELBO = {elbo_value}")

        # Early break
        if np.abs(elbo_value - prev_elbo) < 1e-6:
            print(f"Converged after {i+1} iterations.")
            return {"a_N": a_N, "b_N": b_N, "mu_N": mu_N, "lambda_N": lambda_N,␣
↪"elbos": elbo_history, "mus": mu_history, "lambdas": lambdas_history, "taus":
↪ estimated_taus_history}

        prev_elbo = elbo_value

    return {"a_N": a_N, "b_N": b_N, "mu_N": mu_N, "lambda_N": lambda_N, "elbos":
↪ elbo_history, "mus": mu_history, "lambdas": lambdas_history, "taus":␣
↪estimated_taus_history}
```

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.

[29]:
```
def plot_results(cavi_results,dataset):
    N = len(dataset)
    elbo_history = cavi_results["elbos"]
    mu_history = cavi_results["mus"]
    taus_history = cavi_results["taus"]

    # Plot ELBO over iterations
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.plot(elbo_history, label='ELBO')
    plt.title(f'N = {N}, ELBO Evolution')
    plt.xlabel('Iterations')
    plt.ylabel('ELBO')
    plt.grid(True)
    plt.legend()

    # mu, tau over iterations
    plt.subplot(1, 2, 2)
    plt.plot(mu_history, label='$\mu_N$', color='blue')
    plt.plot(taus_history, label='$a_N/b_N$', color='orange')
    plt.title(f'N = {N}, Variational Parameters Evolution')
    plt.xlabel('Iterations')
    plt.ylabel('Parameter Value')
    plt.grid(True)
    plt.legend()

    plt.tight_layout()
```

```python
    plt.savefig(f"figures/{N}_ELBOandMuTau.pdf")
    plt.show()

    posterior_params = compute_exact_posterior(dataset, a_0, b_0, mu_0, lambda_0)
    ml_estimates = ML_est(dataset)

    plt.figure(figsize=(8, 5))
    # Exact posterior mean and precision
    plt.axvline(posterior_params["mu_N"], color="g", linestyle="--", label="Exact Posterior (Mean)")
    print(f"Posterior mu: {posterior_params["mu_N"]}")
    # CAVI mean
    plt.axvline(cavi_results["mu_N"], color="b", linestyle=":", label="CAVI (Mean)")
    print(f"CAVI mu: {cavi_results["mu_N"]}")
    # ML estimate
    plt.axvline(ml_estimates[0], color="r", linestyle="-", label="ML Estimate (Mean)")
    print(f"ML mu: {ml_estimates[0]}")
    plt.title(f"N = {N}, Comparison of Posterior, CAVI, and ML")
    plt.xlabel("Mean ( )")
    plt.legend()
    plt.grid()

    #plt.figure(figsize=(8, 5))
    # Exact posterior mean and precision
    plt.axvline(posterior_params["a_N"]/posterior_params["b_N"], color="g", linestyle="--", label="Exact Posterior (tau)")
    print(f"Posterior tau: {posterior_params["a_N"]/posterior_params["b_N"]}")
    # CAVI mean
    plt.axvline(cavi_results["taus"][-1], color="b", linestyle=":", label="CAVI (tau)")
    print(f"CAVI tau: {cavi_results["taus"][-1]}")
    # ML estimate
    plt.axvline(ml_estimates[1], color="r", linestyle="-", label="ML Estimate (tau)")
    print(f"ML tau: {ml_estimates[1]}")
    plt.title(f"N = {N}, Comparison of Posterior, CAVI, and ML")
    plt.xlabel("Value")
    plt.legend()
    plt.grid()
    plt.savefig(f"figures/{N}MLexactCAVIcomp.pdf")
    plt.show()
```

```
<>:19: SyntaxWarning: invalid escape sequence '\m'
<>:19: SyntaxWarning: invalid escape sequence '\m'
```

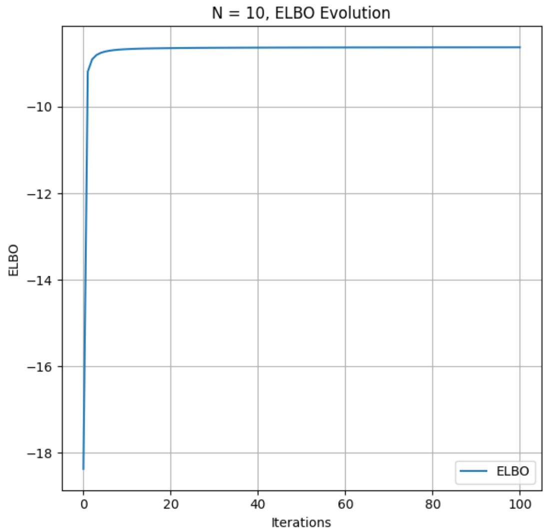
```
/var/folders/7t/7qxxs14161512g3d5tsqp_xw0000gn/T/ipykernel_56106/247895124.py:19
: SyntaxWarning: invalid escape sequence '\m'
  plt.plot(mu_history, label='$\mu_N$', color='blue')
```

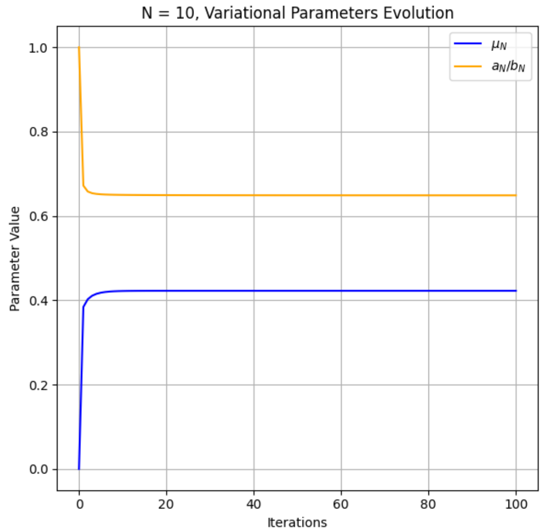
[30]:
```
for data in datasets.values():
    cavi_results = CAVI(data, mu_0, lambda_0, a_0, b_0)
    plot_results(cavi_results,data)
```

```
Iteration 1: ELBO = -9.187323262863385
Iteration 2: ELBO = -8.90285603434964
Iteration 3: ELBO = -8.801838848750094
Iteration 4: ELBO = -8.751275155939119
Iteration 5: ELBO = -8.72158059235591
Iteration 6: ELBO = -8.702394079451324
Iteration 7: ELBO = -8.68915702298984
Iteration 8: ELBO = -8.679563842516467
Iteration 9: ELBO = -8.672333954458827
Iteration 10: ELBO = -8.666706758768708
Iteration 11: ELBO = -8.66220705764017
Iteration 12: ELBO = -8.658525818553109
Iteration 13: ELBO = -8.655455269098043
Iteration 14: ELBO = -8.652851640506002
Iteration 15: ELBO = -8.650612866644842
Iteration 16: ELBO = -8.648664775429673
Iteration 17: ELBO = -8.64695229465658
Iteration 18: ELBO = -8.645433714355429
Iteration 19: ELBO = -8.64407686217685
Iteration 20: ELBO = -8.642856503479164
Iteration 21: ELBO = -8.641752541195423
Iteration 22: ELBO = -8.640748747491237
Iteration 23: ELBO = -8.639831855008577
Iteration 24: ELBO = -8.63899089516783
Iteration 25: ELBO = -8.6382167088492
Iteration 26: ELBO = -8.637501579162814
Iteration 27: ELBO = -8.636838951959753
Iteration 28: ELBO = -8.636223220300879
Iteration 29: ELBO = -8.635649556191492
Iteration 30: ELBO = -8.635113777710338
Iteration 31: ELBO = -8.634612242978932
Iteration 32: ELBO = -8.634141764728263
Iteration 33: ELBO = -8.633699540850008
Iteration 34: ELBO = -8.633283097482758
Iteration 35: ELBO = -8.63289024202373
Iteration 36: ELBO = -8.632519024070856
Iteration 37: ELBO = -8.63216770275357
Iteration 38: ELBO = -8.631834719249909
Iteration 39: ELBO = -8.63151867354342
Iteration 40: ELBO = -8.631218304668797
```

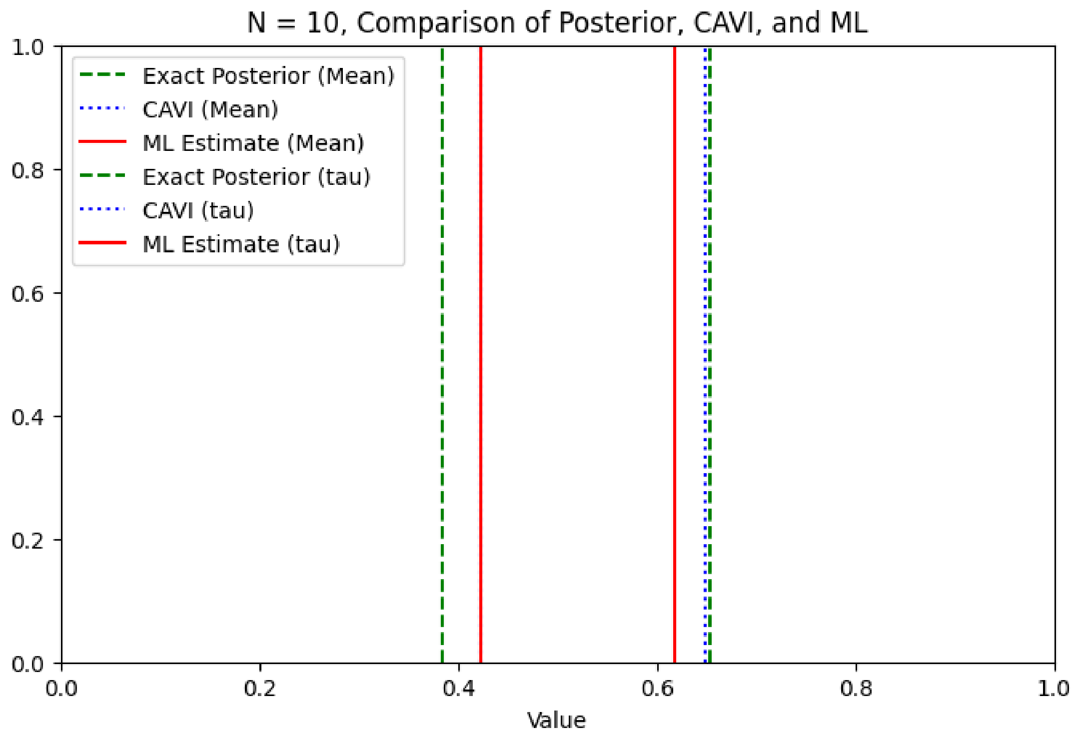
```
Iteration 41: ELBO = -8.630932473845293
Iteration 42: ELBO = -8.63066015001403
Iteration 43: ELBO = -8.630400397386522
Iteration 44: ELBO = -8.63015236468411
Iteration 45: ELBO = -8.629915275805518
Iteration 46: ELBO = -8.629688421705449
Iteration 47: ELBO = -8.629471153304461
Iteration 48: ELBO = -8.629262875280254
Iteration 49: ELBO = -8.629063040614936
Iteration 50: ELBO = -8.628871145792958
Iteration 51: ELBO = -8.62868672656077
Iteration 52: ELBO = -8.628509354173078
Iteration 53: ELBO = -8.62833863206166
Iteration 54: ELBO = -8.62817419287229
Iteration 55: ELBO = -8.628015695823196
Iteration 56: ELBO = -8.627862824344982
Iteration 57: ELBO = -8.627715283967772
Iteration 58: ELBO = -8.627572800425714
Iteration 59: ELBO = -8.627435117953338
Iteration 60: ELBO = -8.627301997751452
Iteration 61: ELBO = -8.627173216603097
Iteration 62: ELBO = -8.627048565622943
Iteration 63: ELBO = -8.626927849125147
Iteration 64: ELBO = -8.626810883596848
Iteration 65: ELBO = -8.62669749676609
Iteration 66: ELBO = -8.626587526753944
Iteration 67: ELBO = -8.626480821302396
Iteration 68: ELBO = -8.626377237069942
Iteration 69: ELBO = -8.626276638988237
Iteration 70: ELBO = -8.626178899673626
Iteration 71: ELBO = -8.626083898888252
Iteration 72: ELBO = -8.625991523045814
Iteration 73: ELBO = -8.62590166475786
Iteration 74: ELBO = -8.625814222416665
Iteration 75: ELBO = -8.625729099811384
Iteration 76: ELBO = -8.625646205774363
Iteration 77: ELBO = -8.62556545385494
Iteration 78: ELBO = -8.62548676201823
Iteration 79: ELBO = -8.625410052366663
Iteration 80: ELBO = -8.625335250882422
Iteration 81: ELBO = -8.625262287188786
Iteration 82: ELBO = -8.625191094328935
Iteration 83: ELBO = -8.625121608560702
Iteration 84: ELBO = -8.625053769165874
Iteration 85: ELBO = -8.624987518272963
Iteration 86: ELBO = -8.624922800692259
Iteration 87: ELBO = -8.624859563762266
Iteration 88: ELBO = -8.624797757206538
```

```
Iteration 89: ELBO = -8.624737333000121
Iteration 90: ELBO = -8.624678245244917
Iteration 91: ELBO = -8.624620450053225
Iteration 92: ELBO = -8.62456390543879
Iteration 93: ELBO = -8.624508571214973
Iteration 94: ELBO = -8.624454408899311
Iteration 95: ELBO = -8.624401381624098
Iteration 96: ELBO = -8.624349454052576
Iteration 97: ELBO = -8.624298592300265
Iteration 98: ELBO = -8.62424876386109
Iteration 99: ELBO = -8.624199937538
Iteration 100: ELBO = -8.624152083377703
```





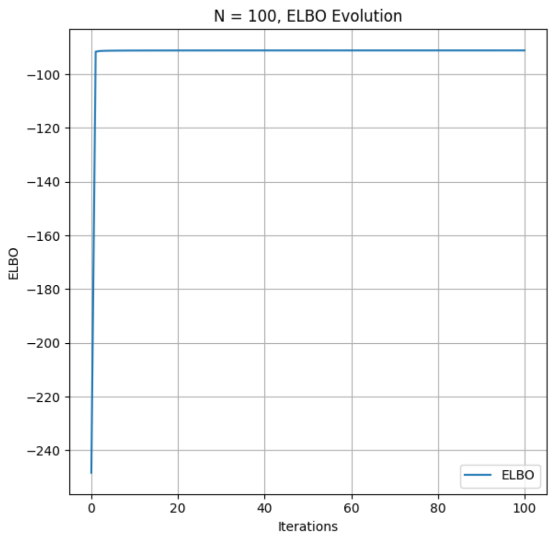

```
Posterior mu: 0.3841121169952442
CAVI mu: 0.42252332869476866
ML mu: 0.4225233286947686
Posterior tau: 0.6541310166157251
CAVI tau: 0.6488655557317343
ML tau: 0.6179456690768029
```

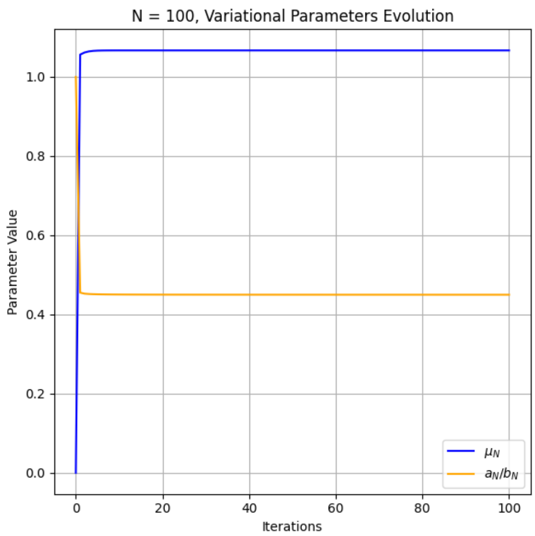



```
Iteration 1: ELBO = -91.58168991703877
Iteration 2: ELBO = -91.35537097890905
Iteration 3: ELBO = -91.28579282079392
Iteration 4: ELBO = -91.25134286406013
Iteration 5: ELBO = -91.22991085598186
Iteration 6: ELBO = -91.21495968713218
Iteration 7: ELBO = -91.2038697834088
Iteration 8: ELBO = -91.19532610918294
Iteration 9: ELBO = -91.18856242121484
Iteration 10: ELBO = -91.18309055218947
Iteration 11: ELBO = -91.17858236783403
Iteration 12: ELBO = -91.1748093922292
Iteration 13: ELBO = -91.17160830822048
Iteration 14: ELBO = -91.16885981111402
Iteration 15: ELBO = -91.16647506219832
Iteration 16: ELBO = -91.16438674773254
Iteration 17: ELBO = -91.16254304273174
Iteration 18: ELBO = -91.16090345755161
Iteration 19: ELBO = -91.15943593042796
Iteration 20: ELBO = -91.1581147591426
Iteration 21: ELBO = -91.15691910694878
Iteration 22: ELBO = -91.15583190750611
Iteration 23: ELBO = -91.15483905114054
```

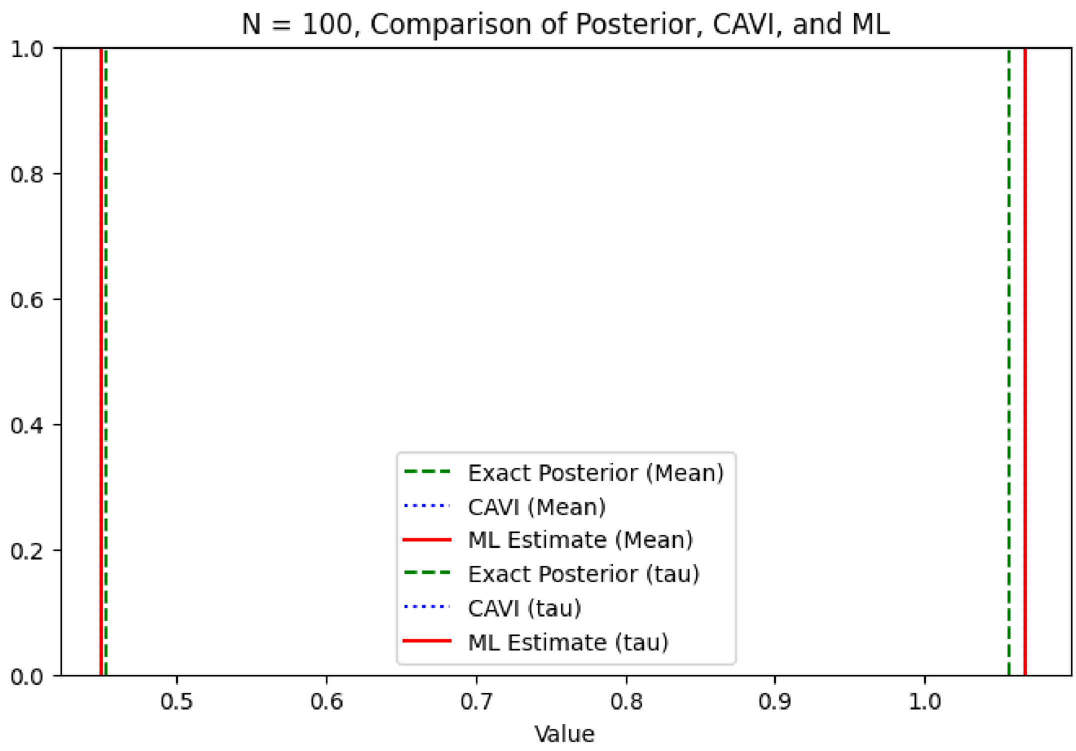
```
Iteration 24: ELBO = -91.15392877226483
Iteration 25: ELBO = -91.15309118259775
Iteration 26: ELBO = -91.15231791142237
Iteration 27: ELBO = -91.15160182539616
Iteration 28: ELBO = -91.15093680816443
Iteration 29: ELBO = -91.15031758542334
Iteration 30: ELBO = -91.14973958487208
Iteration 31: ELBO = -91.1491988232054
Iteration 32: ELBO = -91.14869181424824
Iteration 33: ELBO = -91.14821549376036
Iteration 34: ELBO = -91.14776715748563
Iteration 35: ELBO = -91.14734440980354
Iteration 36: ELBO = -91.14694512092483
Iteration 37: ELBO = -91.1465673910179
Iteration 38: ELBO = -91.14620951999079
Iteration 39: ELBO = -91.14586998191622
Iteration 40: ELBO = -91.14554740328713
Iteration 41: ELBO = -91.14524054445104
Iteration 42: ELBO = -91.14494828369244
Iteration 43: ELBO = -91.14466960353461
Iteration 44: ELBO = -91.14440357890678
Iteration 45: ELBO = -91.14414936688715
Iteration 46: ELBO = -91.14390619778253
Iteration 47: ELBO = -91.14367336734573
Iteration 48: ELBO = -91.14345022996437
Iteration 49: ELBO = -91.14323619268362
Iteration 50: ELBO = -91.14303070994475
Iteration 51: ELBO = -91.14283327894252
Iteration 52: ELBO = -91.14264343551768
Iteration 53: ELBO = -91.14246075051457
Iteration 54: ELBO = -91.14228482654202
Iteration 55: ELBO = -91.14211529508825
Iteration 56: ELBO = -91.14195181394435
Iteration 57: ELBO = -91.14179406489878
Iteration 58: ELBO = -91.14164175166975
Iteration 59: ELBO = -91.14149459804861
Iteration 60: ELBO = -91.14135234622808
Iteration 61: ELBO = -91.14121475529495
Iteration 62: ELBO = -91.14108159986824
Iteration 63: ELBO = -91.14095266886827
Iteration 64: ELBO = -91.14082776439895
Iteration 65: ELBO = -91.14070670073497
Iteration 66: ELBO = -91.14058930340026
Iteration 67: ELBO = -91.14047540832877
Iteration 68: ELBO = -91.14036486109993
Iteration 69: ELBO = -91.14025751623988
Iteration 70: ELBO = -91.14015323658221
Iteration 71: ELBO = -91.14005189268474
```

```
Iteration 72: ELBO = -91.13995336229168
Iteration 73: ELBO = -91.13985752984318
Iteration 74: ELBO = -91.13976428602189
Iteration 75: ELBO = -91.13967352733792
Iteration 76: ELBO = -91.13958515574554
Iteration 77: ELBO = -91.13949907828916
Iteration 78: ELBO = -91.13941520677808
Iteration 79: ELBO = -91.13933345748467
Iteration 80: ELBO = -91.13925375086552
Iteration 81: ELBO = -91.13917601130349
Iteration 82: ELBO = -91.139100166868
Iteration 83: ELBO = -91.13902614909352
Iteration 84: ELBO = -91.13895389277323
Iteration 85: ELBO = -91.1388833357677
Iteration 86: ELBO = -91.13881441882585
Iteration 87: ELBO = -91.13874708541972
Iteration 88: ELBO = -91.13868128159005
Iteration 89: ELBO = -91.13861695580108
Iteration 90: ELBO = -91.1385540588069
Iteration 91: ELBO = -91.13849254352493
Iteration 92: ELBO = -91.13843236491937
Iteration 93: ELBO = -91.13837347989
Iteration 94: ELBO = -91.13831584717009
Iteration 95: ELBO = -91.13825942723008
Iteration 96: ELBO = -91.13820418218624
Iteration 97: ELBO = -91.13815007571665
Iteration 98: ELBO = -91.13809707298147
Iteration 99: ELBO = -91.13804514054773
Iteration 100: ELBO = -91.13799424631947
```





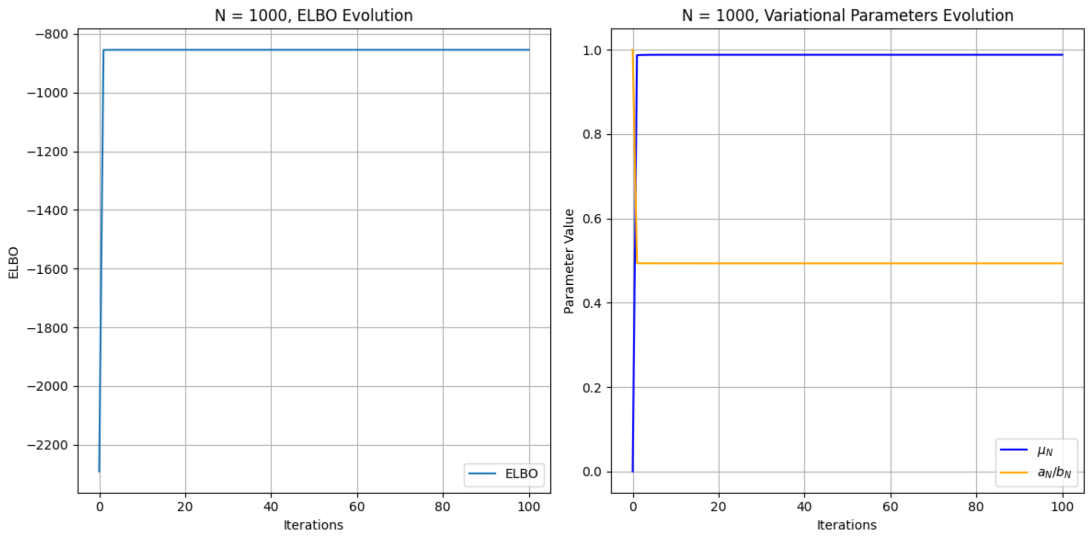
```
Posterior mu: 1.0562298179442813
CAVI mu: 1.0667921161237242
ML mu: 1.0667921161237242
Posterior tau: 0.45261338017085945
CAVI tau: 0.4497396428472921
ML tau: 0.4499819818276406
```




```
Iteration 1: ELBO = -854.7426719244954
Iteration 2: ELBO = -854.4958466164718
Iteration 3: ELBO = -854.4142098448644
Iteration 4: ELBO = -854.3734310868642
Iteration 5: ELBO = -854.348888327698
Iteration 6: ELBO = -854.3324555490261
Iteration 7: ELBO = -854.3206702591896
Iteration 8: ELBO = -854.3118028228499
Iteration 9: ELBO = -854.3048897783746
Iteration 10: ELBO = -854.2993504256987
Iteration 11: ELBO = -854.2948133669037
Iteration 12: ELBO = -854.2910298427438
Iteration 13: ELBO = -854.2878269563269
Iteration 14: ELBO = -854.2850808267947
Iteration 15: ELBO = -854.2827003966447
Iteration 16: ELBO = -854.2806172574069
```

```
Iteration 17: ELBO = -854.2787790342159
Iteration 18: ELBO = -854.2771449572036
Iteration 19: ELBO = -854.2756828212139
Iteration 20: ELBO = -854.2743668519979
Iteration 21: ELBO = -854.2731761790324
Iteration 22: ELBO = -854.2720937232095
Iteration 23: ELBO = -854.2711053737985
Iteration 24: ELBO = -854.2701993706999
Iteration 25: ELBO = -854.269365834697
Iteration 26: ELBO = -854.2685964059673
Iteration 27: ELBO = -854.2678839628488
Iteration 28: ELBO = -854.2672224008238
Iteration 29: ELBO = -854.2666064572
Iteration 30: ELBO = -854.2660315708464
Iteration 31: ELBO = -854.2654937690621
Iteration 32: ELBO = -854.264989575656
Iteration 33: ELBO = -854.2645159357284
Iteration 34: ELBO = -854.2640701537274
Iteration 35: ELBO = -854.2636498421228
Iteration 36: ELBO = -854.2632528786322
Iteration 37: ELBO = -854.26287737038
Iteration 38: ELBO = -854.2625216237141
Iteration 39: ELBO = -854.2621841186767
Iteration 40: ELBO = -854.2618634872815
Iteration 41: ELBO = -854.2615584949957
Iteration 42: ELBO = -854.2612680248468
Iteration 43: ELBO = -854.2609910637569
Iteration 44: ELBO = -854.2607266907356
Iteration 45: ELBO = -854.2604740666519
Iteration 46: ELBO = -854.2602324253395
Iteration 47: ELBO = -854.2600010658238
Iteration 48: ELBO = -854.2597793455479
Iteration 49: ELBO = -854.2595666743993
Iteration 50: ELBO = -854.2593625094711
Iteration 51: ELBO = -854.2591663504397
Iteration 52: ELBO = -854.2589777354544
Iteration 53: ELBO = -854.2587962375286
Iteration 54: ELBO = -854.258621461295
Iteration 55: ELBO = -854.2584530401425
Iteration 56: ELBO = -854.2582906336439
Iteration 57: ELBO = -854.2581339252574
Iteration 58: ELBO = -854.2579826202726
Iteration 59: ELBO = -854.2578364439589
Iteration 60: ELBO = -854.2576951398977
Iteration 61: ELBO = -854.257558468484
Iteration 62: ELBO = -854.2574262055725
Iteration 63: ELBO = -854.2572981412446
Iteration 64: ELBO = -854.2571740787047
```

```
Iteration 65: ELBO = -854.2570538332645
Iteration 66: ELBO = -854.2569372314301
Iteration 67: ELBO = -854.2568241100623
Iteration 68: ELBO = -854.2567143156213
Iteration 69: ELBO = -854.2566077034637
Iteration 70: ELBO = -854.2565041372154
Iteration 71: ELBO = -854.2564034881793
Iteration 72: ELBO = -854.2563056348149
Iteration 73: ELBO = -854.2562104622336
Iteration 74: ELBO = -854.2561178617626
Iteration 75: ELBO = -854.2560277305213
Iteration 76: ELBO = -854.2559399710456
Iteration 77: ELBO = -854.2558544909334
Iteration 78: ELBO = -854.2557712025186
Iteration 79: ELBO = -854.2556900225808
Iteration 80: ELBO = -854.2556108720499
Iteration 81: ELBO = -854.2555336757708
Iteration 82: ELBO = -854.2554583622453
Iteration 83: ELBO = -854.2553848634292
Iteration 84: ELBO = -854.255313114511
Iteration 85: ELBO = -854.2552430537344
Iteration 86: ELBO = -854.2551746222118
Iteration 87: ELBO = -854.2551077637647
Iteration 88: ELBO = -854.2550424247679
Iteration 89: ELBO = -854.2549785540059
Iteration 90: ELBO = -854.2549161025383
Iteration 91: ELBO = -854.2548550235791
Iteration 92: ELBO = -854.2547952723742
Iteration 93: ELBO = -854.254736806091
Iteration 94: ELBO = -854.2546795837278
Iteration 95: ELBO = -854.2546235660013
Iteration 96: ELBO = -854.2545687152679
Iteration 97: ELBO = -854.2545149954377
Iteration 98: ELBO = -854.2544623718908
Iteration 99: ELBO = -854.2544108114105
Iteration 100: ELBO = -854.2543602821039
```




```
Posterior mu: 0.9872624864465467
CAVI mu: 0.9882497489329933
ML mu: 0.9882497489329932
Posterior tau: 0.4938620077824699
CAVI tau: 0.49361078527716995
ML tau: 0.49360018564056446
```