Simulation Based Inference – Comparing Mixture Density Network & Normalizing Flow

University of Nottingham
UK | CHINA | MALAYSIA

Department of Mathematical Sciences
Ying Zhang, Supervised by Theodore Kypraios

Background

Simulation-Based Inference (SBI) addresses the challenge of posterior estimation in complex probabilistic models, particularly when likelihood functions are intractable or computationally expensive to evaluate. Traditional Markov Chain Monte Carlo (MCMC) methods often become inefficient or infeasible in high-dimensional parameter spaces or with computationally intensive simulators.

Aim of this project

This work focuses on two neural posterior approximation techniques – **Mixture Density of Gaussian** and **Normalizing Flow,** emphasizing the benefits of amortized inference, where a one-time training phase enables rapid posterior estimation across different observations, significantly reducing computational costs.

By simulating a large amount of data, the neural network is able to learn **a mapping from observed data $\boldsymbol{y}$ to posterior information about parameters $\boldsymbol{\theta}$.**

$$\boldsymbol{y} \rightarrow \boldsymbol{p}(\boldsymbol{\theta}|\boldsymbol{y})$$

That is, the network learns to approximate the posterior distribution directly from simulated data, without needing an explicit likelihood function.

Model 1: Mixture Density Network

A Mixture Density Gaussian Network learns to output the parameters of a Gaussian mixture distribution conditioned on observed data. This allows it to represent **complex, possibly multi-modal posterior** distributions, making it much more expressive than a single Gaussian approximation.
So instead of predicting a single Gaussian, the network predicts the entire mixture:

$$q_\phi(\theta|y) = \sum_{k=1}^{\mathrm{K}} \pi_k(y) N(\theta|\mu_k(y),\ \sigma_k^2(y))$$

Training Loss:
Maximize the likelihood of the observed data

$$\mathcal{L}_{Mixture}(y) = -\frac{1}{N}\sum_{i=1}^{N} \log\left[\sum_{k=1}^{K} \pi_k(y) \cdot \mathrm{N}\left(\theta_i|\mu_k(y), \sigma_k^2(y)\right)\right]$$

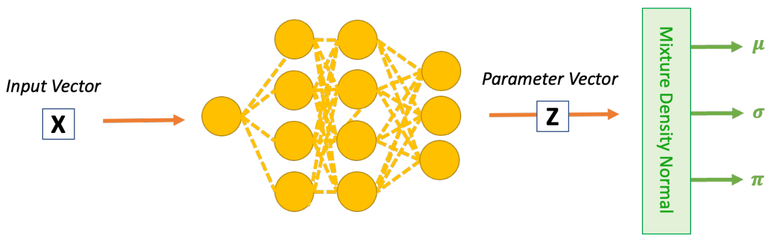

Figure 1: Flow Chart of Mixture Density Network

Model 2: Normalizing Flow

A Normalizing Flow learns a sequence of invertible transformations that map a simple base distribution (Gaussian in this project) into a complex posterior distribution. This makes it extremely flexible, able to represent multi-modal, skewed, or high-dimensional posteriors while still allowing exact density evaluation.

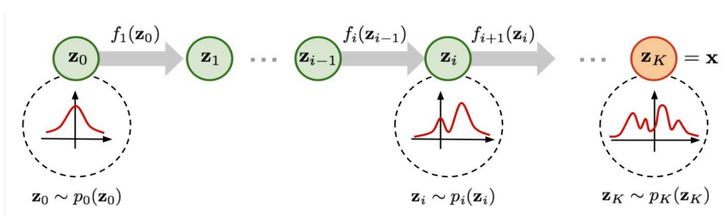

Figure 2: A sequence of invertible transform

Since the flow network is conditioned on data y, so it learns:

$$q_\phi(\theta|y) = f_\phi(z; y), z \sim \mathrm{N}(0, I)$$

Each transformation f_i is **invertible and differentiable**, we can track how the probability density changes:

$$p(\theta) = p(z_0) \cdot \prod_{i=1}^{K} \left|\det \frac{\partial \mathrm{f}_i^{-1}}{\partial z_i}\right|$$

Model 2: Normalizing Flow - Continued

KL-Divergence
We assume our normalizing flow gives us the transformed distribution $q(\theta|y)$, we use KL-divergence to represent the distance between the true posterior and our estimated posterior:

$$KL(q_\phi(\theta|\boldsymbol{y})||\pi(\theta|\boldsymbol{y})) = E_{q_\phi(\theta|\boldsymbol{y})}\left[log\frac{q_\phi(\theta|\boldsymbol{y})}{\pi(\theta|\boldsymbol{y})}\right]$$

And based on Bayesian theorem, we have:

$$\pi(\theta|y) \propto \pi(y|\theta)\pi(\theta)$$

Plugging into the KL expression:

$$KL(q_\phi||\pi) = E_{q_\phi}\left[q_\phi(\theta|\boldsymbol{y}) - \pi(\boldsymbol{y}|\theta) - \pi(\theta) + \log\pi(\boldsymbol{y})\right]$$

Since $\pi(\mathrm{y})$ is constant with respect to θ, it can be dropped when optimizing. Thus, we can define our loss function to be loss function to be:

$$\mathcal{L}(\phi) = \log q_\phi(\theta|\boldsymbol{y}) - \log\pi(y|\theta) - \log\pi(\theta)$$

Where $q_\phi(\theta|\boldsymbol{y})$ is defined in equation 1.

Experiment 1 Gamma Example

1. Sample $\theta_i \sim Gamma(\alpha, \beta)$
2. For each θ, simulate N independent Poisson trials with rate θ_i, and compute their sum

$$y^i = \sum_{j=1}^{N} Poisson\,(\theta_i)$$

3. Store raw training triples $(\theta_i,\ y^i,\ N)$ across different values of N.
4. Group by observations: For each unique observation pair (y^i, N) gather all corresponding θ_i samples. These grouped samples approximate draws from the posterior distribution $p(\theta|y, N)$
5. Feature construction: Encode each observation with numerical features $[y/N, \log(1+y), \log(1+N)]$ to be used as inputs for the neural network.
6. Train MDN and NF that take observation features as input and output parameters of a Gaussian mixture or conditional flows.
7. Evaluation: Compare the MDN approximation of $p(\theta|y, N)$ against the analytical Gamma posterior

$$\theta|y, N \sim Gamma(\alpha + y\,, \beta + N)$$

Comparison between MDN & NF

MDG Training Process:
For each unique y' and N, train through the neural network, output the parameters of our MDG, then for all the $\theta_{\mathrm{y'}}$ substitute them into the likelihood function and find the maximum of the likelihood.
We train our model with N in [10,50,100,150,200,300]
And test our unseen pairs for $y_{sum} = 10, n = 5\ and\ y_{sum} = 50, n = 50$

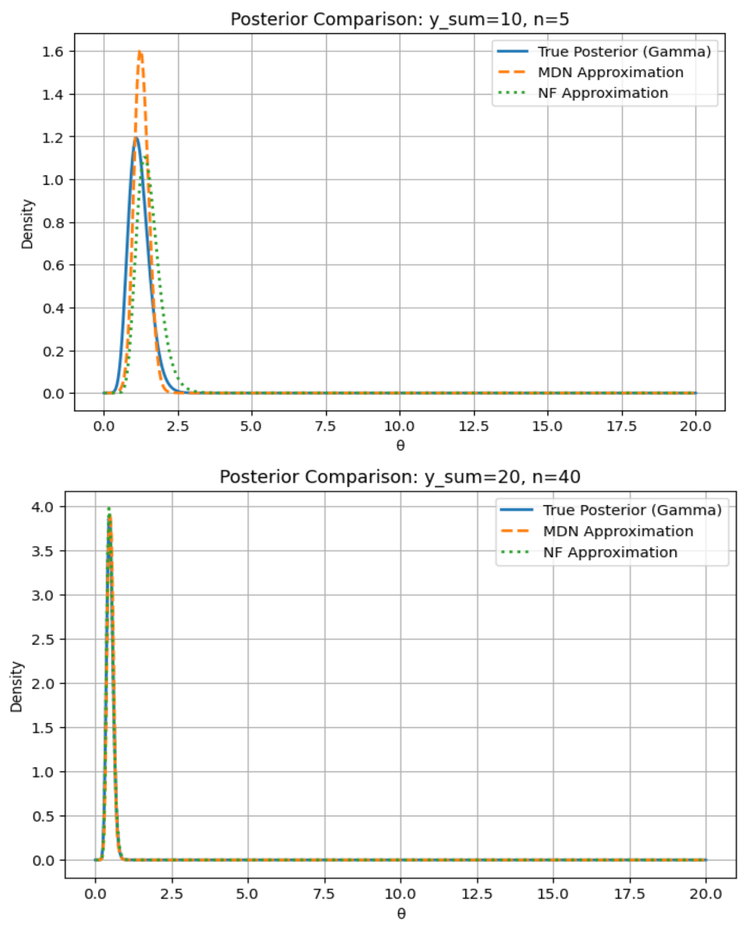

Figure 3: Comparison between MDN & NF in Gamma Example

Experiment 3 : SIR Model

In the case of the SIR model, given N (number of population) and y_{obs} (number of infected people), what is the possible distribution of the infection rate θ where:

- θ: Infection rate parameter
- y : observed number of infections
- N: Household size

We simulate infection spread in a household using a simplified stochastic SIR process.
For different household size N , we sample :

$$\theta \sim Exponential(1)$$

Apply simple stochastic SIR process and obtain the result as shown in the figure below.

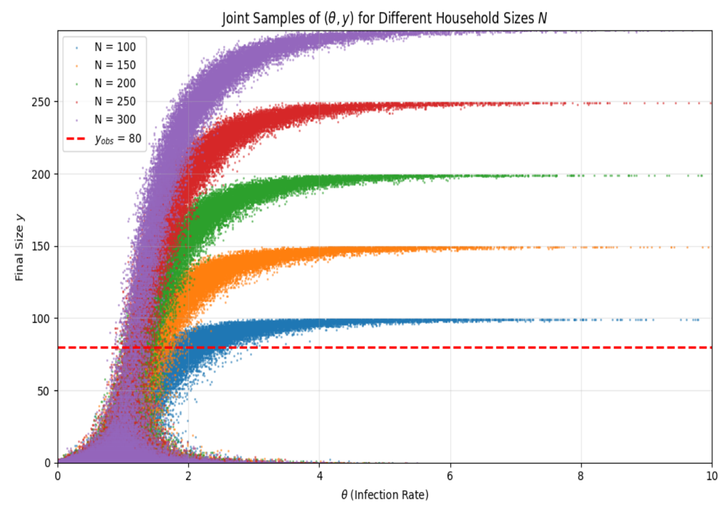

Figure 4: Simulation of SIR Model

The red dashed line represents the observed number of infections $y_{obs} = 80$, which intersects all the θ values used to generate the data.

We can see that for small N, reaching 80 infections requires a relatively larger θ than that of relatively smaller θ

We collect the triples (θ, y, N), and train the neural network through Mixture Density Network and Normalizing Flow.
As we can see, our range of N lies between 0 and 300, and what we want to see is whether the model can perform well outside the training range.
Our results are shown below.

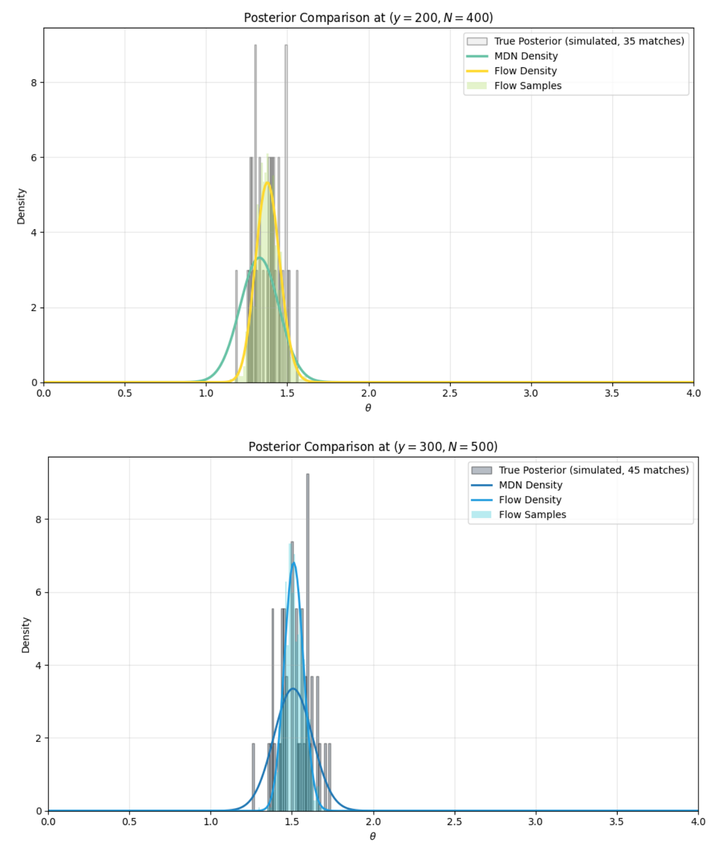

Figure 5: Comparison between MDN & NF in SIR Model

- Both models can approximate the posterior distribution of θ, where the **Normalizing Flow provides a more accurate match** to the simulated truth, especially outside the training range (N=400,500).
- The experiment demonstrates that **Flow-based models generalize better** than MDNs when extrapolating to unseen household sizes.

Reflection

Through this project, I have gained valuable insights into both the theoretical and practical aspects of Simulation-Based Inference (SBI). I learned how to design experiments that bridge statistical theory with machine learning techniques. I would like to acknowledge the invaluable guidance of my supervisor- his continuous support, constructive feedback, and encouragement played a crucial role in shaping the direction of this project.