Simulation Based Inference using Marginal Neural Ratio Estimation

Gianmarco Puleo

SISSA, March 26th, 2025

Summary and Goals

Given an observation of some data generated from the model

$$d_i = A\cos(\omega t_i + \phi) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \tag{1}$$

, we want to estimate the posterior

$$p(\theta|x) = \frac{\mathcal{L}(\theta, x)\pi(\theta)}{p(x)} = r(\theta, x)\pi(\theta) \tag{2}$$

where $\mathcal{L}(\theta, x)$ is the likelihood, $\pi(\theta)$ is the prior and $p(x)$ is the evidence. We present two ways of doing it:

1. Markov Chain Monte Carlo (MCMC)
2. Simulation Based Inference (SBI) via Neural Ratio Estimation (NRE)

Markov Chain Monte Carlo (Metropolis-Hastings)

We build a Markov chain of parameter samples, such that its limit distribution is the posterior.

Sketch of the algorithm:

1. Initialise parameters θ_0
2. Generate a proposal next state θ^* from a proposal distribution $q(\theta^*|\theta_0)$
3. Accept it with probability

$$P(\text{accept}) = \min\left(1, \frac{p(\theta^*)p(x|\theta^*)}{p(\theta_0)p(x|\theta_0)}\right)$$

4. Repeat by generating a proposal from the current state.

*For simplicity, assume a symmetric proposal and a uniform prior.

Issues with MCMC

- One should discard burn-in time and thin the chain by the autocorrelation time...
- The bottleneck is the repeated evaluations of the likelihood, which can be costly.
- In some cases, the likelihood is not even known in closed form!

Simulation Based Inference to the rescue

Assume we have a simulator $S : \theta \rightarrow d$ that generates data d from known parameters θ. Neural Ratio Estimation uses S to let a neural network learn the likelihood-to-evidence ratio $r(\theta, x) = p(x|\theta)/p(x)$. To this end we need to define:

1. A dataset
2. A task (i. e. , a loss function)
3. A model architecture

Neural Ratio Estimation: the dataset

The **dataset** is generated as follows

- Generate $x_i = (\theta_i, d_i)$ pairs from the joint distribution $p(\theta, d)$, by sampling $\theta_i \sim \pi(\theta)$ and simulating $d_i = S(\theta_i)$.
- Scramble the pairs by randomly permuting the parameter instances, resulting in samples from the marginal distribution $p(d)p(\theta)$.
- To each pair, associate a binary label $C_i = 1$ if the pair is scrambled and $C_i = 0$ if it is not.

Goal

It can be shown that

$$p(C_i = 1|x_i) = \frac{1}{1 + r(\theta_i, d_i)}.$$

$\Longrightarrow$ we want to perform a classification task!

Neural Ratio Estimation: the loss function

Let $f_\phi(\theta)$ be the neural network output with trainable parameters ϕ. It is trained by minimising the binary cross-entropy loss with gradient descent:

$$\mathcal{L}(\phi) = -\frac{1}{N}\sum_{i=1}^{N}\left\{ C_i \log \sigma(f_\phi(x_i)) + (1 - C_i) \log \sigma(1 - f_\phi(x_i)) \right\} \tag{3}$$

where σ is the sigmoid function.
NRE is enabled by the following equality:

$$\sigma(f_\phi(x)) = \frac{1}{1 + \exp(-f_\phi(x))} \approx p(C_i = 1|x_i) = \frac{1}{1 + r(\theta_i, d_i)} \tag{4}$$

Neural Ratio Estimation: the loss function

Let $f_\phi(\theta)$ be the neural network output with trainable parameters ϕ. It is trained by minimising the binary cross-entropy loss with gradient descent:

$$\mathcal{L}(\phi) = -\frac{1}{N}\sum_{i=1}^{N}\left\{ C_i \log\sigma(f_\phi(x_i)) + (1 - C_i)\log\sigma(1 - f_\phi(x_i)) \right\} \tag{3}$$

where σ is the sigmoid function.
NRE is enabled by the following equality:

$$\sigma(f_\phi(x)) = \frac{1}{1+\exp(-f_\phi(x))} \approx p(C_i = 1|x_i) = \frac{1}{1+r(\theta_i, d_i)} \tag{4}$$

$$\implies f_\phi(x) \approx -\log(r(\theta_i, d_i)). \tag{5}$$

Model Architecture

We actually want all 1-dimensional and 2-dimensional marginal posteriors. This suggests the following architecture:

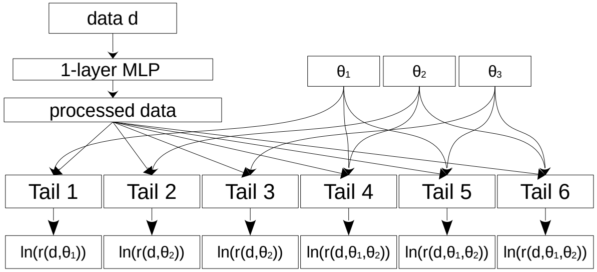


Figure: Neural network architecture for marginal NRE. The tails are fully connected layers.

Training technicalities

- the data were generated by equation (1) with $\sigma = 0.1$ at 10 evenly spaced time steps $t_i \in [0, 1]$.
- The priors for the parameters are uniform distributions:

$$\omega \sim \mathcal{U}(2, 4), \quad \phi \sim \mathcal{U}(0, 6.28), \quad A \sim \mathcal{U}(0.5, 1.5). \tag{6}$$

- the total number of trainable parameters is 6286.
- I used the Adam optimizer with learning rate 10^{-3} and SGD with batches of 100 samples. I used a dataset of 10^6 samples to train. A validation set was used to monitor the training and for early stopping.
- The training took approximately one hour on my laptop (no GPU).
- I used `lightning` and `PyTorch` for model training. The code is available at https://github.com/g-puleo/bayes-i-exam.git.

Results: marginal posteriors I

In order to plot the posterior distribution for some fixed observation d_{obs}:

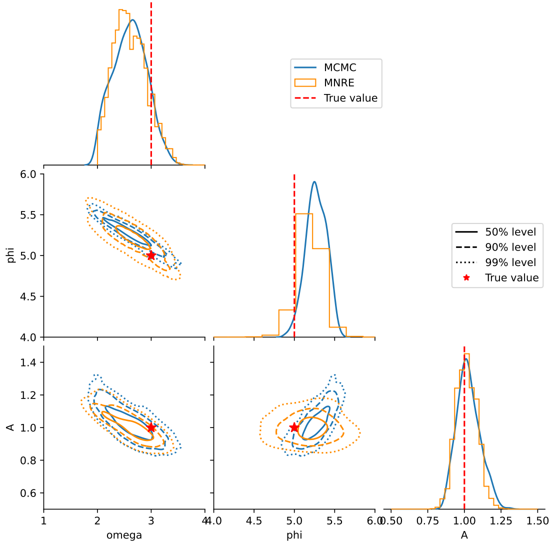
- sample parameters θ_i from the priors,
- and then we reweight them by the NRE estimates of the likelihood-to-evidence ratio,

$$w_i = \exp(-f_\phi(\theta_i, d_{\mathrm{obs}}))$$

.

Results: marginal posteriors II

This allows to produce histograms of the marginal posterior distributions.



Marginal posteriors for the parameters $\theta = (\omega, A, \phi)$. The blue lines are the MCMC posteriors, while the orange lines are the NRE estimates.

Results: p-p plot

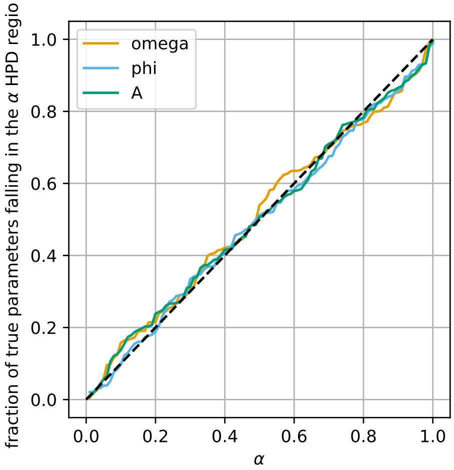


Figure: P-P plot for the NRE estimates. For each parameter we are close to the diagonal.

Results: coverage map

For each point in a (A, ω) grid, we evaluate the coverage of the α-HPD region of the posterior on a set of 1000 observation from different ϕ values.

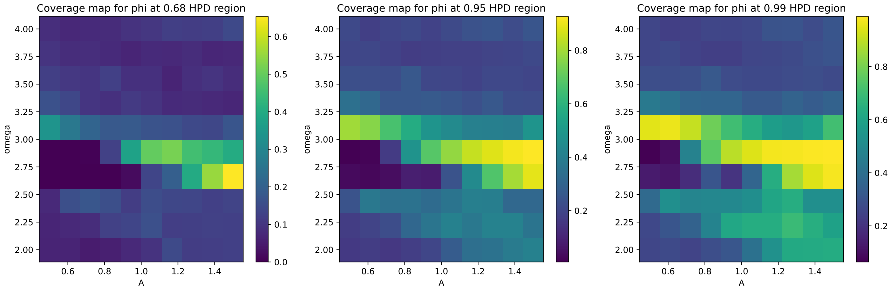


Figure: Coverage map for the NRE estimates. The neural network does not seem optimal in all regions of the parameter space where it was trained.

Conclusion

- The trained NRE network can produce **amortized** estimates of the **posterior**, thus the likelihood costs are completely avoided.
- The model was implemented in a scalable and efficient way using `PyTorch` and `lightning`.
- The coverage of these posterior is optimal in some regions of parameter space.
- The reason for sub-optimal coverage properties in some regions might be linked to having a model with too few parameters for the amount of datapoints shown.
- Future work could explore increasing model complexity to improve coverage in underperforming regions.