Neural Simulation-Based Inference

November 2023

Table of Contents

Table of Contents

Simulators

Scientists just love developing mechanistic models of phenomena:

- Particle physics

Simulators

Scientists just love developing mechanistic models of phenomena:

- Particle physics
- Neuroscience

Simulators

Scientists just love developing mechanistic models of phenomena:

- Particle physics
- Neuroscience
- Population genetics

Simulators

Scientists just love developing mechanistic models of phenomena:

- Particle physics
- Neuroscience
- Population genetics
- Epidemiology

Simulators

Scientists just love developing mechanistic models of phenomena:

- Particle physics
- Neuroscience
- Population genetics
- Epidemiology
- Climate/Earth sciences

Simulators

Scientists just love developing mechanistic models of phenomena:

- Particle physics
- Neuroscience
- Population genetics
- Epidemiology
- Climate/Earth sciences
- Astrophysics

Simulators

Scientists just love developing mechanistic models of phenomena:

- Particle physics
- Neuroscience
- Population genetics
- Epidemiology
- Climate/Earth sciences
- Astrophysics

Simulators

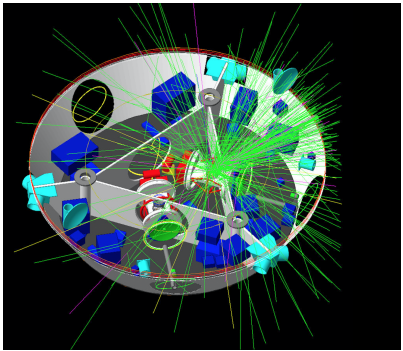
Scientists just love developing mechanistic models of phenomena:

- Particle physics
- Neuroscience
- Population genetics
- Epidemiology
- Climate/Earth sciences
- Astrophysics

Thanks to modern computing, they can turn these into high-precision simulators.

Simulators: An Example

Geant4 - particle simulator (CERN 2023).



Science (like, most of it)

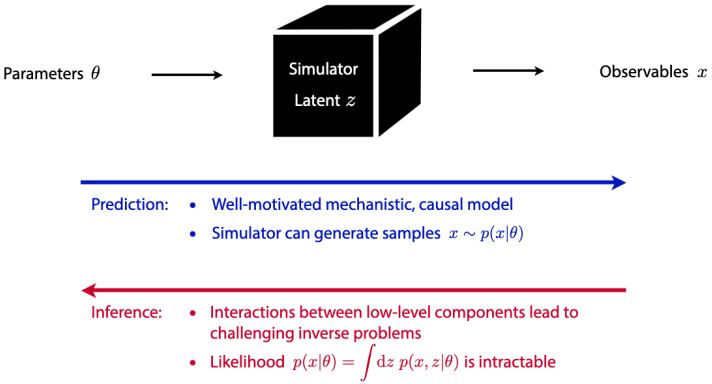


Figure: Credit: Johann Brehmer (Brehmer 2022).

Likelihood-Free Inference

The Problem: Performing inference over parameters θ with data x

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

when we don't have access to the likelihood $p(x|\theta)$, but we can sample $x \sim p(x|\theta)$.

Table of Contents

Traditional Methods: Summary Statistics

- Choose a good summary statistic $s(x)$ based on domain knowledge (feature engineering)—this is *hard*.

Traditional Methods: Summary Statistics

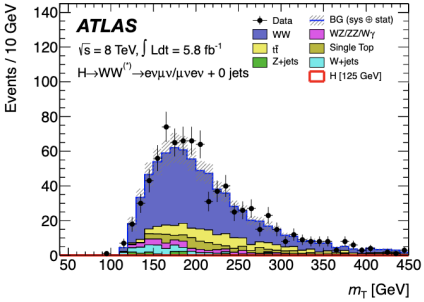
- Choose a good summary statistic $s(x)$ based on domain knowledge (feature engineering)—this is *hard*.
- Approximate the likelihood $p(s|\theta)$ using histograms or kernel density estimation (e.g. Diggle and Gratton 1984).

Traditional Methods: Summary Statistics

- Choose a good summary statistic $s(x)$ based on domain knowledge (feature engineering)—this is *hard*.
- Approximate the likelihood $p(s|\theta)$ using histograms or kernel density estimation (e.g. Diggle and Gratton 1984).
- Does not scale well with the dimension of $s(x)$.

Traditional Methods: Summary Statistics

- Choose a good summary statistic $s(x)$ based on domain knowledge (feature engineering)—this is *hard*.
- Approximate the likelihood $p(s|\theta)$ using histograms or kernel density estimation (e.g. Diggle and Gratton 1984).
- Does not scale well with the dimension of $s(x)$.



Essentially what was used for the Higgs boson discovery (ATLAS Collaboration 2012).

Traditional Methods: Approximate Bayesian Computation

Given true observed data x_o, want to find a set of suitable parameters $\theta \sim p(\theta|x_o)$.

1. Sample $\theta \sim p(\theta)$.

(Rubin 1984)

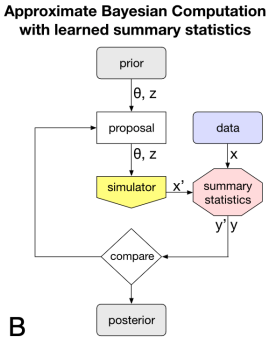


Figure: Cranmer, Brehmer, and Louppe 2020

Traditional Methods: Approximate Bayesian Computation

Given true observed data x_o, want to find a set of suitable parameters $\theta \sim p(\theta|x_o)$.

1. Sample $\theta \sim p(\theta)$.
2. Simulate $x \sim p(x|\theta)$.

(Rubin 1984)

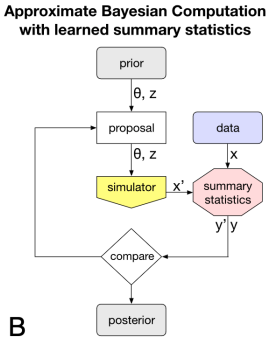


Figure: Cranmer, Brehmer, and Louppe 2020

Traditional Methods: Approximate Bayesian Computation

Given true observed data x_o, want to find a set of suitable parameters $\theta \sim p(\theta|x_o)$.

1. Sample $\theta \sim p(\theta)$.
2. Simulate $x \sim p(x|\theta)$.
3. Calculate summary statistics $s(x_o)$ and $s(x)$.

(Rubin 1984)

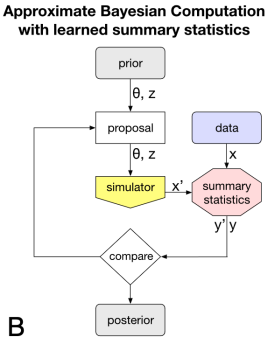


Figure: Cranmer, Brehmer, and Louppe 2020

Traditional Methods: Approximate Bayesian Computation

Given true observed data x_o, want to find a set of suitable parameters $\theta \sim p(\theta|x_o)$.

1. Sample $\theta \sim p(\theta)$.
2. Simulate $x \sim p(x|\theta)$.
3. Calculate summary statistics $s(x_o)$ and $s(x)$.
4. Accept θ if the distance $\rho(s(x_o), s(x)) \leq \epsilon$, reject otherwise.

(Rubin 1984)

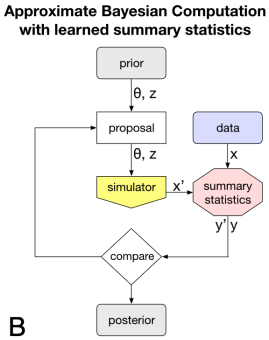


Figure: Cranmer, Brehmer, and Louppe 2020

Traditional Methods: Approximate Bayesian Computation

Given true observed data x_o, want to find a set of suitable parameters $\theta \sim p(\theta|x_o)$.

1. Sample $\theta \sim p(\theta)$.
2. Simulate $x \sim p(x|\theta)$.
3. Calculate summary statistics $s(x_o)$ and $s(x)$.
4. Accept θ if the distance $\rho(s(x_o), s(x)) \leq \epsilon$, reject otherwise.
5. Repeat.

(Rubin 1984)

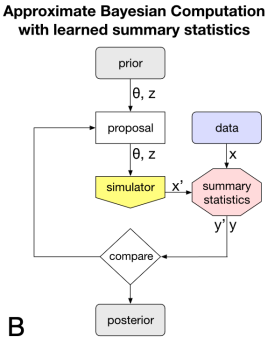


Figure: Cranmer, Brehmer, and Louppe 2020

Traditional Methods: Approximate Bayesian Computation

Given true observed data x_o, want to find a set of suitable parameters $\theta \sim p(\theta|x_o)$.

1. Sample $\theta \sim p(\theta)$.
2. Simulate $x \sim p(x|\theta)$.
3. Calculate summary statistics $s(x_o)$ and $s(x)$.
4. Accept θ if the distance $\rho(s(x_o), s(x)) \leq \epsilon$, reject otherwise.
5. Repeat.

(Rubin 1984)

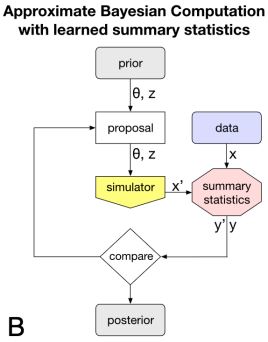


Figure: Cranmer, Brehmer, and Louppe 2020

Traditional Methods: Approximate Bayesian Computation

Given true observed data x_o, want to find a set of suitable parameters $\theta \sim p(\theta|x_o)$.

1. Sample $\theta \sim p(\theta)$.
2. Simulate $x \sim p(x|\theta)$.
3. Calculate summary statistics $s(x_o)$ and $s(x)$.
4. Accept θ if the distance $\rho(s(x_o), s(x)) \leq \epsilon$, reject otherwise.
5. Repeat.

(Rubin 1984)

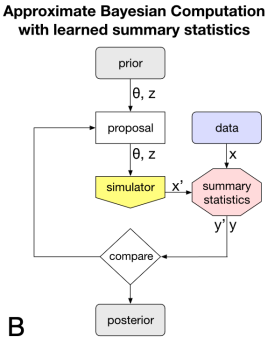


Figure: Cranmer, Brehmer, and Louppe 2020

Problems with ABC

Three main problems:

1. Choosing $s(x)$ to be informative and low-dimensional.

Problems with ABC

Three main problems:

1. Choosing $s(x)$ to be informative and low-dimensional.
2. Choosing ρ and ϵ.

Problems with ABC

Three main problems:

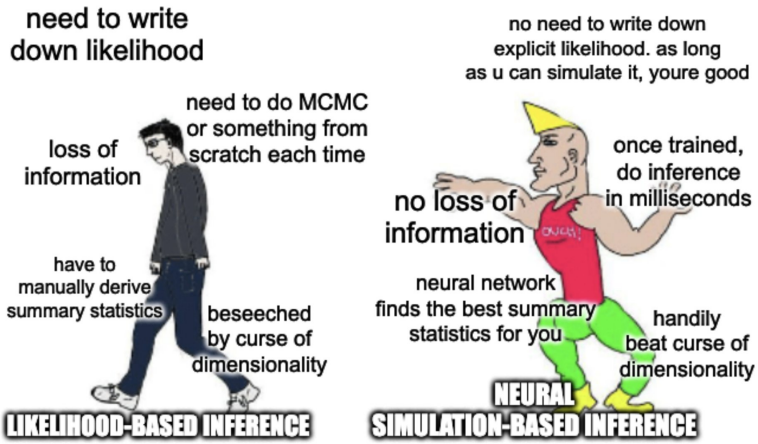
1. Choosing $s(x)$ to be informative and low-dimensional.
2. Choosing ρ and ϵ.
3. Tends to be very slow for small ϵ.

Problems with ABC

Three main problems:

1. Choosing $s(x)$ to be informative and low-dimensional.
2. Choosing ρ and ϵ.
3. Tends to be very slow for small ϵ.
   - (Simulation is typically the bottleneck in SBI.)

Modern Simulation-Based Inference



Using Neural Networks

Two main approaches to Neural SBI:

1. Supervised Learning: Use NN to model a likelihood ratio (NRE)

$$r(x|\theta) := \frac{p(x|\theta)}{p_{\text{ref}}(x)} \text{ or } r(x|\theta_0, \theta_1) := \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

Using Neural Networks

Two main approaches to Neural SBI:

1. Supervised Learning: Use NN to model a likelihood ratio (NRE)

$$r(x|\theta) := \frac{p(x|\theta)}{p_{\text{ref}}(x)} \text{ or } r(x|\theta_0, \theta_1) := \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

2. Unsupervised Learning: Use conditional neural density estimates (e.g. normalizing flows)
   - Neural Posterior Estimation (NPE): Model $p(\theta|x)$
   - Neural Likelihood Estimation (NLE): Model $p(x|\theta)$

Table of Contents

Neural Likelihood Ratio Estimation (NRE)

Neyman-Pearson lemma (Neyman, E. S. Pearson, and K. Pearson 1933): The most powerful test statistic to compare two hypotheses θ_0 and θ_1 for an observation x is the likelihood ratio:

$$r(x|\theta_0, \theta_1) := \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

NRE: Likelihood Ratio Trick

Use supervised learning to obtain a classifier $\mathrm{d}(x)$ to distinguish between

- $x \sim p(x|\theta_0)$ with class label $y = 1$, and
- $x \sim p(x|\theta_1)$ with class label $y = 0$.

NRE: Likelihood Ratio Trick

Use supervised learning to obtain a classifier $\mathrm{d}(x)$ to distinguish between

- $x \sim p(x|\theta_0)$ with class label $y = 1$, and
- $x \sim p(x|\theta_1)$ with class label $y = 0$.

This has optimal classifier

$$\mathrm{d}^*(x) = p(y = 1|x) = \frac{p(x|\theta_0)}{p(x|\theta_0) + p(x|\theta_1)}$$

NRE: Likelihood Ratio Trick

Use supervised learning to obtain a classifier $\mathrm{d}(x)$ to distinguish between

- $x \sim p(x|\theta_0)$ with class label $y = 1$, and
- $x \sim p(x|\theta_1)$ with class label $y = 0$.

This has optimal classifier

$$\mathrm{d}^*(x) = p(y = 1|x) = \frac{p(x|\theta_0)}{p(x|\theta_0) + p(x|\theta_1)}$$

Which can be used to obtain the likelihood ratio

$$r(x|\theta_0, \theta_1) = \frac{p(x|\theta_0)}{p(x|\theta_1)} = \frac{\mathrm{d}^*(x)}{1 - \mathrm{d}^*(x)}$$

.

NRE: Parameterized Classifier

Clearly we can't train a new classifier d for every possible pair of $\theta_0, \theta_1 \in \Theta$.

NRE: Parameterized Classifier

Clearly we can't train a new classifier d for every possible pair of $\theta_0, \theta_1 \in \Theta$.

- So train a 'parameterised classifier'* $d(x, \theta)$ that takes in a proposed θ as input.

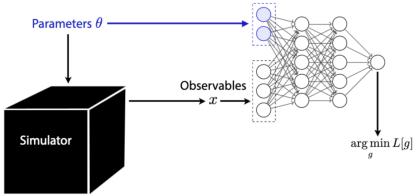


Figure: Credit: Johann Brehmer (Brehmer 2022).

*(Cranmer, Pavez, and Louppe 2016; Hermans, Begy, and Louppe 2020)

NRE: Parameterized Classifier

Clearly we can't train a new classifier d for every possible pair of $\theta_0, \theta_1 \in \Theta$.

- So train a 'parameterised classifier'* $d(x, \theta)$ that takes in a proposed θ as input.
- Try to target the likelihood ratio against some reference $p_{\text{ref}}(x)$

$$r(x|\theta) = \frac{p(x|\theta)}{p_{\text{ref}}(x)}$$

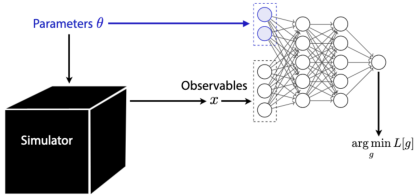


Figure: Credit: Johann Brehmer (Brehmer 2022).

*(Cranmer, Pavez, and Louppe 2016; Hermans, Begy, and Louppe 2020)

NRE: Parameterized Classifier

- In particular, train parameterised classifier $\mathrm{d}(x, \theta)$ to distinguish between
  1. dependent $(x, \theta) \sim p(x, \theta)$ pairs with class label $y = 1$,

$$\theta \sim p(\theta), \quad x \sim p(x|\theta)$$

  2. independent (x, θ') pairs with class label $y = 0$

$$\theta' \sim p(\theta)$$

NRE: Parameterized Classifier

- In particular, train parameterised classifier $\mathrm{d}(x, \theta)$ to distinguish between
  1. dependent $(x, \theta) \sim p(x, \theta)$ pairs with class label $y = 1$,

$$\theta \sim p(\theta), \quad x \sim p(x|\theta)$$

  2. independent (x, θ') pairs with class label $y = 0$

$$\theta' \sim p(\theta)$$

- This gives us the optimal classifier

$$\mathrm{d}^*(x, \theta) = p(y = 1|x, \theta) = \frac{p(x, \theta)}{p(x, \theta) + p(x)p(\theta)}$$

NRE: Parameterized Classifier

With this optimal classifier

$$\mathrm{d}^*(x, \theta) = p(y = 1|x, \theta) = \frac{p(x, \theta)}{p(x, \theta) + p(x)p(\theta)}$$

NRE: Parameterized Classifier

With this optimal classifier

$$\mathrm{d}^*(x, \theta) = p(y = 1|x, \theta) = \frac{p(x, \theta)}{p(x, \theta) + p(x)p(\theta)}$$

the likelihood trick gives us:

$$\frac{\mathrm{d}^*(x, \theta)}{1 - \mathrm{d}^*(x, \theta)} = \frac{p(x, \theta)}{p(x)p(\theta)} = \frac{p(x|\theta)}{p(x)} = r(x|\theta)$$

i.e. we're implicitly using $p_{\text{ref}}(x) = p(x)$.

NRE: Algorithm Psuedocode

Algorithm 1 Optimization of $\mathbf{d}_\phi(\mathbf{x}, \boldsymbol{\theta})$.

Inputs:	Criterion ℓ (e.g., BCE)
	Implicit generative model $p(\mathbf{x} \mid \boldsymbol{\theta})$
	Prior $p(\boldsymbol{\theta})$
Outputs:	Parameterized classifier $\mathbf{d}_\phi(\mathbf{x}, \boldsymbol{\theta})$
Hyperparameters:	Batch-size M

while not converged do
  Sample $\boldsymbol{\theta} \leftarrow \{\boldsymbol{\theta}_m \sim p(\boldsymbol{\theta})\}_{m=1}^{M}$
  Sample $\boldsymbol{\theta}' \leftarrow \{\boldsymbol{\theta}_m \sim p(\boldsymbol{\theta})\}_{m=1}^{M}$
  Simulate $\mathbf{x} \leftarrow \{\mathbf{x}_m \sim p(\mathbf{x} \mid \boldsymbol{\theta}_m)\}_{m=1}^{M}$
  $\mathcal{L} \leftarrow \ell(\mathbf{d}_\phi(\mathbf{x}, \boldsymbol{\theta}),\ 1) + \ell(\mathbf{d}_\phi(\mathbf{x}, \boldsymbol{\theta}'),\ 0)$
  $\phi \leftarrow \textbf{OPTIMIZER}(\phi,\ \nabla_\phi \mathcal{L})$
end while
return $\mathbf{d}_\phi$

Figure: Hermans, Begy, and Louppe 2020

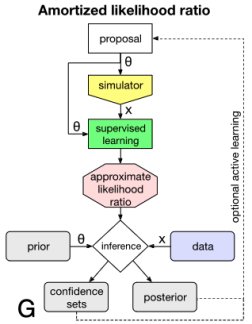


Figure: Cranmer, Brehmer, and Louppe 2020

NRE: Algorithm Diagram

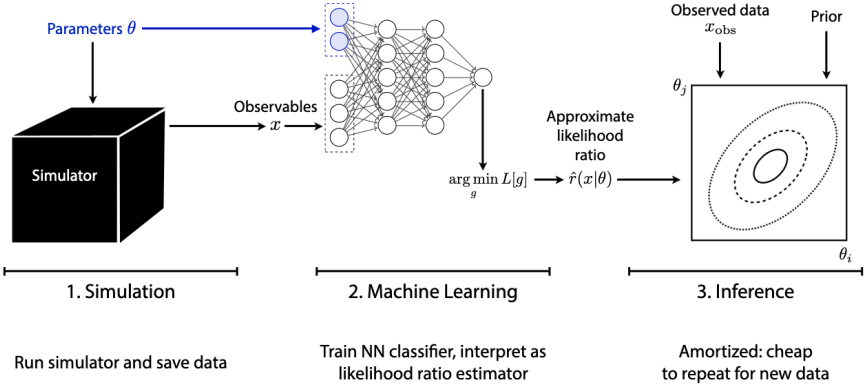


Figure: Hermans, Begy, and Louppe 2020

NRE: Inference

- Frequentist: Run for a whole lot of values of θ and get some confidence intervals.

NRE: Inference

- Frequentist: Run for a whole lot of values of θ and get some confidence intervals.
- Bayesian:
  - Metropolis-Hastings MCMC

$$\alpha = \min\left(1, \frac{p(\theta')p(x|\theta')q(\theta'|\theta_t)}{p(\theta_t)p(x|\theta_t)q(\theta_t|\theta')}\right)$$

  - Hamiltonian Monte Carlo

$$\nabla_\theta U(\theta) = -\frac{\nabla_\theta p(x|\theta)}{p(x|\theta)}$$

.

Table of Contents

Conditional Neural Density Estimation

- Train a NN q_ϕ to learn a density $p(u|v)$, given iid pairs $\{(u_i, v_i)\}_{i=1}^n$ from the joint distribution $p(u, v)$ by maximising the total log probability w.r.t. ϕ

$$\sum_{i=1}^{n} \log q_\phi(u_i|v_i)$$

Conditional Neural Density Estimation: Mixture Density Networks

- Mixure Density Network (MDN): $q_\phi(u|v)$ is a mixture of K Gaussians:

$$q_\phi(u|v) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(u|\mu_k, \Sigma_k)$$

where $\{\alpha_k\}, \{\mu_k\}, \{\Sigma_k\}$ are computed by a feed-forward NN.

Conditional Neural Density Estimation: Mixture Density Networks

- Mixure Density Network (MDN): $q_\phi(u|v)$ is a mixture of K Gaussians:

$$q_\phi(u|v) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(u|\mu_k, \Sigma_k)$$

where $\{\alpha_k\}, \{\mu_k\}, \{\Sigma_k\}$ are computed by a feed-forward NN.

  - Not very popular anymore (used in Papamakarios and Murray 2018), but nice and simple (and easily normalised).

Conditional Neural Density Estimation: Normalising Flows

- Normalising Flow: $q_\phi(u|v)$ is a transformation of a standard Gaussian density $\mathcal{N}(0, \mathrm{I})$ through a series of K invertible, differentiable functions $f_1, \ldots, f_K$ which each depend on v, i.e.:

$$z_0 \sim \mathcal{N}(0, \mathrm{I}), \quad z_k = f_k(z_{k-1}, v) \in \mathbb{R}^M, \quad u = z_K$$

Conditional Neural Density Estimation: Normalising Flows

- Normalising Flow: $q_\phi(u|v)$ is a transformation of a standard Gaussian density $\mathcal{N}(0, \mathrm{I})$ through a series of K invertible, differentiable functions $f_1, \ldots, f_K$ which each depend on v, i.e.:

$$z_0 \sim \mathcal{N}(0, \mathrm{I}), \quad z_k = f_k(z_{k-1}, v) \in \mathbb{R}^M, \quad u = z_K$$

- By a change of variables we get that

$$q_\phi(u|v) = \mathcal{N}(z_0|0, \mathrm{I}) \prod_{k=1}^{K} \left| \det \left(\frac{\partial f_k}{\partial z_{k-1}} \right) \right|^{-1}.$$

Conditional Neural Density Estimation: Normalising Flows

- Normalising Flow: $q_\phi(u|v)$ is a transformation of a standard Gaussian density $\mathcal{N}(0, \mathrm{I})$ through a series of K invertible, differentiable functions $f_1, \ldots, f_K$ which each depend on v, i.e.:

$$z_0 \sim \mathcal{N}(0, \mathrm{I}), \quad z_k = f_k(z_{k-1}, v) \in \mathbb{R}^M, \quad u = z_K$$

- By a change of variables we get that

$$q_\phi(u|v) = \mathcal{N}(z_0|0, \mathrm{I}) \prod_{k=1}^{K} \left| \det \left(\frac{\partial f_k}{\partial z_{k-1}} \right) \right|^{-1}.$$

- One choice is a Masked Autoregressive Flow (MAF) (Papamakarios, Pavlakou, and Murray 2018).

Normalising Flows: MAF

- Each component $z_k^{(j)}$ is computed depending on the previous components $z_k^{(1)}, \ldots, z_k^{(j-1)}$:

$$p(z_k^{(j)} | z_k^{(1:j)}) = \mathcal{N}(z_k^{(j)} | \mu_k^{(j)}, (\exp \alpha_k^{(j)})^2)$$

where $\mu_k^{(j)}$ and $\alpha_k^{(j)}$ come from the NN receiving $z_k^{(1:j-1)}$ and v as input.

Normalising Flows: MAF

- Each component $z_k^{(j)}$ is computed depending on the previous components $z_k^{(1)}, \ldots, z_k^{(j-1)}$:

$$p(z_k^{(j)}|z_k^{(1:j)}) = \mathcal{N}(z_k^{(j)}|\mu_k^{(j)}, (\exp \alpha_k^{(j)})^2)$$

  where $\mu_k^{(j)}$ and $\alpha_k^{(j)}$ come from the NN receiving $z_k^{(1:j-1)}$ and v as input.
- The autoregressive nature of each f_k ensures their jacobians are all triangular:

$$\det \left(\frac{\partial f_k}{\partial z_{k-1}} \right) = \prod_{j=1}^{M} \exp(\alpha_k^{(j)})$$

Neural Posterior Estimation (NPE)

1. Generate n pairs $(x_i, \theta_i) \sim p(x_i, \theta_i)$ from the joint by drawing

$$\theta_i \sim p(\theta)$$
$$x_i \sim p(x|\theta_i)$$

(Papamakarios and Murray 2018)

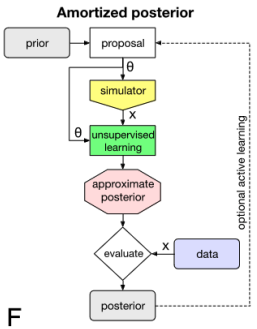


Figure: Cranmer, Brehmer, and Louppe 2020

Neural Posterior Estimation (NPE)

1. Generate n pairs $(x_i, \theta_i) \sim p(x_i, \theta_i)$ from the joint by drawing

$$\theta_i \sim p(\theta)$$
$$x_i \sim p(x|\theta_i)$$

2. Train a neural network q_ϕ to approximate $p(\theta|x)$ by maximising $\sum_{i=1}^{n} \log q_\phi(\theta_i|x_i)$.

(Papamakarios and Murray 2018)

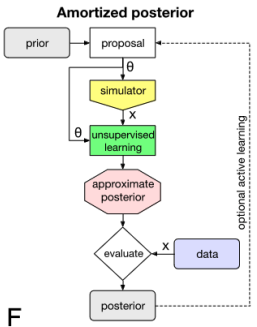


Figure: Cranmer, Brehmer, and Louppe 2020

Neural Likelihood Estimation (NLE)

Model the likelihood rather than the posterior.

1. Train a neural network q_ϕ to approximate $p(x|\theta)$ by maximising $\sum_{i=1}^{n} \log q_\phi(x_i|\theta_i)$.

(Papamakarios, Sterratt, and Murray 2019)

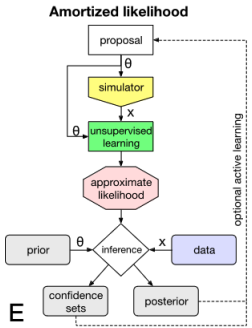


Figure: Cranmer, Brehmer, and Louppe 2020

Neural Likelihood Estimation (NLE)

Model the likelihood rather than the posterior.

1. Train a neural network q_ϕ to approximate $p(x|\theta)$ by maximising $\sum_{i=1}^{n} \log q_\phi(x_i|\theta_i)$.
2. Once q_ϕ is trained, proceed by using your likelihood in MCMC to obtain posterior samples.

(Papamakarios, Sterratt, and Murray 2019)

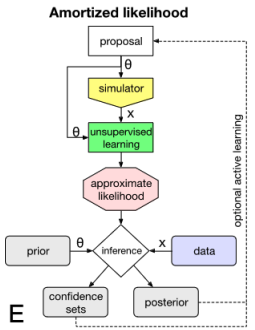


Figure: Cranmer, Brehmer, and Louppe 2020

Table of Contents

Problems with NPE & NLE

- NPE & NLE require a lot of data to accurately describe the posterior everywhere.

Problems with NPE & NLE

- NPE & NLE require a lot of data to accurately describe the posterior everywhere.
- But we might only want to learn the posterior in parameter- and data-space relevant to our real-world observations x_o.

Problems with NPE & NLE

- NPE & NLE require a lot of data to accurately describe the posterior everywhere.
- But we might only want to learn the posterior in parameter- and data-space relevant to our real-world observations x_o.
- Solution: sequentially choose where in Θ our next training pair (θ_i, x_i) should come from so as to be maximally informative.

SNPE (Type A)

- Instead of drawing $\theta \sim p(\theta)$ for each training example, draw from some other proposal distribution $\tilde{p}(\theta)$.

SNPE (Type A)

- Instead of drawing $\theta \sim p(\theta)$ for each training example, draw from some other proposal distribution $\tilde{p}(\theta)$.
  - Such as the current posterior approximation $q_\phi(\theta|x_o)$!

SNPE (Type A)

- Instead of drawing $\theta \sim p(\theta)$ for each training example, draw from some other proposal distribution $\tilde{p}(\theta)$.
  - Such as the current posterior approximation $q_\phi(\theta|x_o)$!
  - (Update this proposal every $N \in \mathbb{N}$ samples.)

SNPE (Type A)

- Instead of drawing $\theta \sim p(\theta)$ for each training example, draw from some other proposal distribution $\tilde{p}(\theta)$.
  - Such as the current posterior approximation $q_\phi(\theta|x_o)$!
  - (Update this proposal every $N \in \mathbb{N}$ samples.)
- Problem: This would end up biasing us away from the true posterior $p(\theta|x_o) \propto p(x_o|\theta)p(\theta)$ and toward $p(\theta|x_o)\tilde{p}(\theta)$ (ignoring normalisation).

SNPE (Type A)

- Instead of drawing $\theta \sim p(\theta)$ for each training example, draw from some other proposal distribution $\tilde{p}(\theta)$.
  - Such as the current posterior approximation $q_\phi(\theta|x_o)$!
  - (Update this proposal every $N \in \mathbb{N}$ samples.)
- Problem: This would end up biasing us away from the true posterior $p(\theta|x_o) \propto p(x_o|\theta)p(\theta)$ and toward $p(\theta|x_o)\tilde{p}(\theta)$ (ignoring normalisation).
- Solution: Model the posterior as $p(\theta|x = x_o) \approx \frac{p(\theta)}{\tilde{p}(\theta)} q_\phi(\theta|x_o)$. (Papamakarios and Murray 2018)

SNPE (Type A)

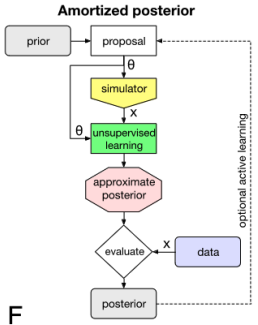
Algorithm 1: Training of proposal prior

initialize $q_\phi(\boldsymbol{\theta} \mid \mathbf{x})$ with one component
$\tilde{p}(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$
repeat
  for $n = 1..N$ **do**
    sample $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$
    sample $\mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n)$
  end
  retrain $q_\phi(\boldsymbol{\theta} \mid \mathbf{x})$ on $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}$
  $\tilde{p}(\boldsymbol{\theta}) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})}\, q_\phi(\boldsymbol{\theta} \mid \mathbf{x}_o)$
until $\tilde{p}(\boldsymbol{\theta})$ *has converged*;

Algorithm 2: Training of posterior

initialize $q_\phi(\boldsymbol{\theta} \mid \mathbf{x})$ with K components
`// if` q_ϕ `available by Algorithm 1`
`// initialize by replicating its`
`// one component` K `times`
for $n = 1..N$ **do**
  sample $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$
  sample $\mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n)$
end
train $q_\phi(\boldsymbol{\theta} \mid \mathbf{x})$ on $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}$
$\hat{p}(\boldsymbol{\theta} \mid \mathbf{x} = \mathbf{x}_o) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})}\, q_\phi(\boldsymbol{\theta} \mid \mathbf{x}_o)$

Figure: Hermans, Begy, and Louppe 2020



F

Figure: Cranmer, Brehmer, and Louppe 2020

SNPE (Type B)

- SNPE-A works, but the division $\frac{p(\theta)}{\tilde{p}(\theta)}$ can be numerically unstable.

SNPE (Type B)

- SNPE-A works, but the division $\frac{p(\theta)}{\tilde{p}(\theta)}$ can be numerically unstable.
- Solution: use importance weights $w_i = \frac{p(\theta_i)}{\tilde{p}(\theta_i)}$ in your NN objective (Lueckmann, Goncalves, et al. 2017):

$$\sum_{i=1}^{n} w_i \log q_\phi(\theta_i | x_i)$$

SNPE (Type B)

- SNPE-A works, but the division $\frac{p(\theta)}{\tilde{p}(\theta)}$ can be numerically unstable.
- Solution: use importance weights $w_i = \frac{p(\theta_i)}{\tilde{p}(\theta_i)}$ in your NN objective (Lueckmann, Goncalves, et al. 2017):

$$\sum_{i=1}^{n} w_i \log q_\phi(\theta_i | x_i)$$

- (Though often w_i have high variance $\rightarrow$ NN gradients have high variance $\rightarrow$ training instability.)

SNLE

- Essentially the same thing but have q_ϕ model the likelihood.

Algorithm 1: Sequential Neural Likelihood (SNL)

Input : observed data $\mathbf{x}_o$, estimator $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$, number of rounds R, simulations per round N

Output: approximate posterior $\hat{p}(\boldsymbol{\theta} \mid \mathbf{x}_o)$

set $\hat{p}_0(\boldsymbol{\theta} \mid \mathbf{x}_o) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
for $r = 1 : R$ **do**
  for $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta} \mid \mathbf{x}_o)$ with MCMC
    simulate $\mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n)$
    add $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ into $\mathcal{D}$
  (re-)train $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$ on $\mathcal{D}$ and set $\hat{p}_r(\boldsymbol{\theta} \mid \mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$
return $\hat{p}_R(\boldsymbol{\theta} \mid \mathbf{x}_o)$

Figure: Papamakarios, Sterratt, and Murray 2019

SNLE

- Essentially the same thing but have q_ϕ model the likelihood.
  - Avoids the posterior-bias problems.

Algorithm 1: Sequential Neural Likelihood (SNL)

Input : observed data $\mathbf{x}_o$, estimator $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$, number of rounds R, simulations per round N

Output: approximate posterior $\hat{p}(\boldsymbol{\theta} \mid \mathbf{x}_o)$

set $\hat{p}_0(\boldsymbol{\theta} \mid \mathbf{x}_o) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
for $r = 1 : R$ **do**
  for $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta} \mid \mathbf{x}_o)$ with MCMC
    simulate $\mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n)$
    add $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ into $\mathcal{D}$
  (re-)train $q_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta})$ on $\mathcal{D}$ and set
  $\hat{p}_r(\boldsymbol{\theta} \mid \mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$
return $\hat{p}_R(\boldsymbol{\theta} \mid \mathbf{x}_o)$

Figure: Papamakarios, Sterratt, and Murray 2019

SNLE

- Essentially the same thing but have q_ϕ model the likelihood.
  - Avoids the posterior-bias problems.
- Note: q_ϕ is no longer a posterior, so we have to sample our $\theta_i \sim \tilde{p}(\theta|x_o)$ using MCMC (using $q_\phi(x_o|\theta)$ as our likelihood).

Algorithm 1: Sequential Neural Likelihood (SNL)

Input : observed data $\mathbf{x}_o$, estimator $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$, number of rounds R, simulations per round N

Output: approximate posterior $\hat{p}(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$

set $\hat{p}_0(\boldsymbol{\theta}\,|\,\mathbf{x}_o) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
for $r = 1 : R$ **do**
  for $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$ with MCMC
    simulate $\mathbf{x}_n \sim p(\mathbf{x}\,|\,\boldsymbol{\theta}_n)$
    add $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ into $\mathcal{D}$
  (re-)train $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$ on $\mathcal{D}$ and set $\hat{p}_r(\boldsymbol{\theta}\,|\,\mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o\,|\,\boldsymbol{\theta})\, p(\boldsymbol{\theta})$
return $\hat{p}_R(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$

Figure: Papamakarios, Sterratt, and Murray 2019

SNLE

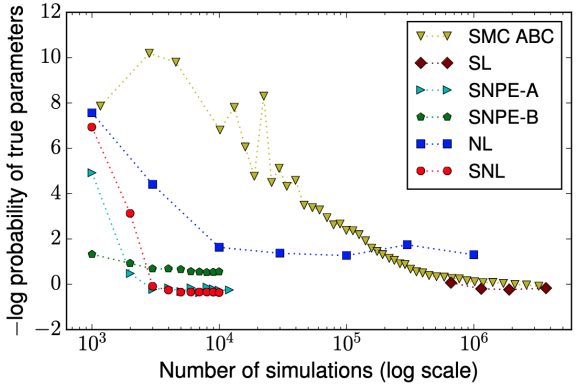


Figure: Papamakarios, Sterratt, and Murray 2019

SNRE

We can use this idea of sampling from the current approximation to the posterior to help us generate training data for NRE too! (Durkan, Murray, and Papamakarios 2020)

Table of Contents

Comparison of Algorithms

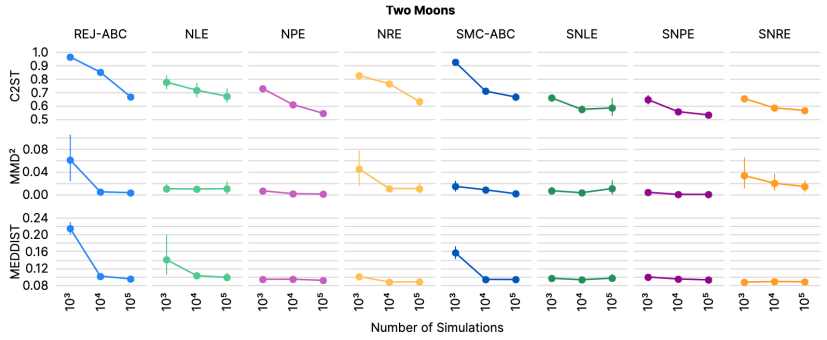


Figure 2: **Performance on Two Moons according to various metrics.** Best possible performance would be 0.5 for C2ST, 0 for MMD^2 and MEDDIST. Results for 10 observations each, means and 95% confidence intervals.

Figure: Lueckmann, Boelts, et al. 2021

Figure: Sharrock et al. 2022

Comparison of Algorithms

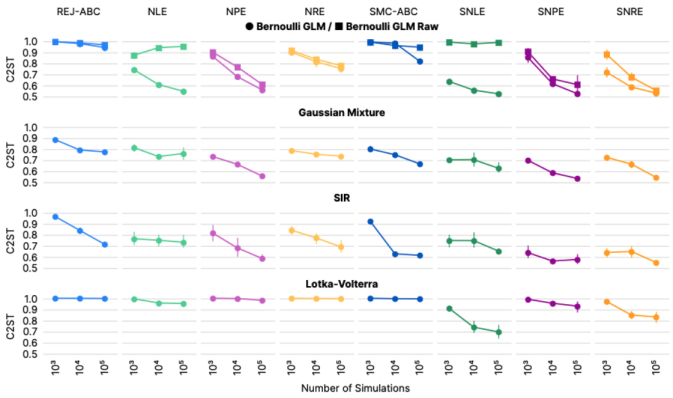


Figure 3: **Performance on other benchmark tasks.** Classification accuracy (C2ST) of REJ-ABC, SMC-ABC, NLE, SNLE, NPE, SNPE, NRE, SNRE for 10 observations each, means and 95% confidence intervals.

Comparison of Algorithms

What each algorithm gives you:

Comparison of Algorithms

What each algorithm gives you:

- ABC:
  - Samples from posterior $\theta \sim q(\theta|x_o)$.

Comparison of Algorithms

What each algorithm gives you:

- ABC:
  - Samples from posterior $\theta \sim q(\theta|x_o)$.
- (S)NLE + (S)NRE:
  - Samples from posterior $\theta \sim q(\theta|x_o)$ via MCMC.
  - Unnormalised evaluations $q(\theta|x_o)$.

Comparison of Algorithms

What each algorithm gives you:

- ABC:
  - Samples from posterior $\theta \sim q(\theta|x_o)$.
- (S)NLE + (S)NRE:
  - Samples from posterior $\theta \sim q(\theta|x_o)$ via MCMC.
  - Unnormalised evaluations $q(\theta|x_o)$.
- (S)NPE:
  - Samples from posterior $\theta \sim q(\theta|x_o)$ directly.
  - Normalised evaluations $q(\theta|x_o)$.

Comparison of Algorithms

- Different problems lend themselves to different algorithms as solutions...

Comparison of Algorithms

- Different problems lend themselves to different algorithms as solutions...
- Sequential algorithms might not always be best:

Comparison of Algorithms

- Different problems lend themselves to different algorithms as solutions...
- Sequential algorithms might not always be best:
  - Not needed if simulation is cheap.

Comparison of Algorithms

- Different problems lend themselves to different algorithms as solutions...
- Sequential algorithms might not always be best:
  - Not needed if simulation is cheap.
  - Performing SBI separately for many data points:

Comparison of Algorithms

- Different problems lend themselves to different algorithms as solutions...
- Sequential algorithms might not always be best:
  - Not needed if simulation is cheap.
  - Performing SBI separately for many data points:
    - Better to have an amortized likelihood/posterior that is accurate everywhere in the data-space $x_o \in \mathcal{X}$.

Comparison of Algorithms

- Different problems lend themselves to different algorithms as solutions...
- Sequential algorithms might not always be best:
  - Not needed if simulation is cheap.
  - Performing SBI separately for many data points:
    - Better to have an amortized likelihood/posterior that is accurate everywhere in the data-space $x_o \in \mathcal{X}$.
    - E.g. for iid data $x_{o_1}, x_{o_2} \ldots \sim p(x|\theta)$

$$p(\theta|x_{o_1}, x_{o_2} \ldots) \propto p(\theta) \prod_i \left[\underbrace{p(x_{o_i}|\theta)}_{\text{amortized likelihood}} \right].$$

Table of Contents

Conclusion

- SNPE/SNLE/SNRE are all pretty popular, but there are other methods out there too, e.g.:

Conclusion

- SNPE/SNLE/SNRE are all pretty popular, but there are other methods out there too, e.g.:
  - GATSBI (Ramesh et al. 2022) — GAN-inspired approach.

Conclusion

- SNPE/SNLE/SNRE are all pretty popular, but there are other methods out there too, e.g.:
  - GATSBI (Ramesh et al. 2022) — GAN-inspired approach.
  - 'Grey-box' approaches (Brehmer, Louppe, et al. 2020) — try to squeeze as much information out of the simulator as possible.

Conclusion

- SNPE/SNLE/SNRE are all pretty popular, but there are other methods out there too, e.g.:
  - GATSBI (Ramesh et al. 2022) — GAN-inspired approach.
  - 'Grey-box' approaches (Brehmer, Louppe, et al. 2020) — try to squeeze as much information out of the simulator as possible.
- SBI is pretty promising for a whole range of scientific subjects

Conclusion

- SNPE/SNLE/SNRE are all pretty popular, but there are other methods out there too, e.g.:
  - GATSBI (Ramesh et al. 2022) — GAN-inspired approach.
  - 'Grey-box' approaches (Brehmer, Louppe, et al. 2020) — try to squeeze as much information out of the simulator as possible.
- SBI is pretty promising for a whole range of scientific subjects
  - Particle physics (Brehmer and Cranmer 2020).

Conclusion

- SNPE/SNLE/SNRE are all pretty popular, but there are other methods out there too, e.g.:
  - GATSBI (Ramesh et al. 2022) — GAN-inspired approach.
  - 'Grey-box' approaches (Brehmer, Louppe, et al. 2020) — try to squeeze as much information out of the simulator as possible.
- SBI is pretty promising for a whole range of scientific subjects
  - Particle physics (Brehmer and Cranmer 2020).
  - Astrophysics — gravitational waves (Delaunoy et al. 2020), dark matter (Hermans, Banik, et al. 2021).

Conclusion

- SNPE/SNLE/SNRE are all pretty popular, but there are other methods out there too, e.g.:
  - GATSBI (Ramesh et al. 2022) — GAN-inspired approach.
  - 'Grey-box' approaches (Brehmer, Louppe, et al. 2020) — try to squeeze as much information out of the simulator as possible.
- SBI is pretty promising for a whole range of scientific subjects
  - Particle physics (Brehmer and Cranmer 2020).
  - Astrophysics — gravitational waves (Delaunoy et al. 2020), dark matter (Hermans, Banik, et al. 2021).
  - Climate science — hydrology of river basins in Colorado Hull et al. 2022.

Useful Papers

The frontier of simulation-based inference

Kyle Cranmer[a,b,1], **Johann Brehmer**[a,b], and **Gilles Louppe**[c]

[a]Center for Cosmology and Particle Physics, New York University, New York, NY 10003; [b]Center for Data Science, New York University, New York, NY 10011; and [c]Montefiore Institute, University of Liège, B-4000 Liège, Belgium

Benchmarking Simulation-Based Inference

Jan-Matthis Lueckmann[1,2] **Jan Boelts**[2] **David S. Greenberg**[2,3]
Pedro J. Gonçalves[4] **Jakob H. Macke**[1,2,5]

[1]University of Tübingen [2]Technical University of Munich [3]Helmholtz Centre Geesthacht
[4]Research Center caesar [5]Max Planck Institute for Intelligent Systems, Tübingen

References I

ATLAS Collaboration (Sept. 2012). "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1. arXiv:1207.7214 [hep-ex], pp. 1–29. ISSN: 03702693. DOI: 10.1016/j.physletb.2012.08.020. URL: http://arxiv.org/abs/1207.7214 (visited on 11/05/2023).

Brehmer, Johann (July 2022). *slides/2022/simulation_based_inference_rodem_sinergia_2022.pdf at master · johannbrehmer/slides*. en. URL: https://github.com/johannbrehmer/slides/blob/master/2022/simulation_based_inference_rodem_sinergia_2022.pdf (visited on 11/06/2023).

References II

Brehmer, Johann and Kyle Cranmer (Nov. 2020). *Simulation-based inference methods for particle physics*. arXiv:2010.06439 [hep-ex, physics:hep-ph, physics:physics, stat]. DOI: 10.48550/arXiv.2010.06439. URL: http://arxiv.org/abs/2010.06439 (visited on 11/05/2023).

Brehmer, Johann, Gilles Louppe, et al. (Mar. 2020). "Mining gold from implicit models to improve likelihood-free inference". In: *Proceedings of the National Academy of Sciences* 117.10. arXiv:1805.12244 [hep-ph, physics:physics, stat], pp. 5242–5249. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1915980117. URL: http://arxiv.org/abs/1805.12244 (visited on 11/06/2023).

References III

CERN (2023). *Geant4*. en. URL: https://geant4.web.cern.ch// (visited on 11/06/2023).

Cranmer, Kyle, Johann Brehmer, and Gilles Louppe (Dec. 2020). "The frontier of simulation-based inference". en. In: *Proceedings of the National Academy of Sciences* 117.48, pp. 30055–30062. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1912789117. URL: https://pnas.org/doi/full/10.1073/pnas.1912789117 (visited on 11/05/2023).

References IV

Cranmer, Kyle, Juan Pavez, and Gilles Louppe (Mar. 2016). *Approximating Likelihood Ratios with Calibrated Discriminative Classifiers*. arXiv:1506.02169 [physics, stat]. DOI: 10.48550/arXiv.1506.02169. URL: http://arxiv.org/abs/1506.02169 (visited on 11/06/2023).

Delaunoy, Arnaud et al. (Dec. 2020). *Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization*. en. arXiv:2010.12931 [astro-ph, physics:gr-qc]. URL: http://arxiv.org/abs/2010.12931 (visited on 11/05/2023).

References V

Diggle, Peter J. and Richard J. Gratton (Jan. 1984). "Monte Carlo Methods of Inference for Implicit Statistical Models". en. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 46.2, pp. 193–212. ISSN: 0035-9246, 2517-6161. DOI: 10.1111/j.2517-6161.1984.tb01290.x. URL: https://rss.onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1984.tb01290.x (visited on 11/05/2023).

Durkan, Conor, Iain Murray, and George Papamakarios (2020). "On Contrastive Learning for Likelihood-free Inference". en. In: URL: https://arxiv.org/pdf/2002.03712.pdf.

References VI

Hermans, Joeri, Nilanjan Banik, et al. (Aug. 2021). "Towards constraining warm dark matter with stellar streams through neural simulation-based inference". In: *Monthly Notices of the Royal Astronomical Society* 507.2. arXiv:2011.14923 [astro-ph, stat], pp. 1999–2011. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stab2181. URL: http://arxiv.org/abs/2011.14923 (visited on 11/05/2023).

Hermans, Joeri, Volodimir Begy, and Gilles Louppe (2020). "Likelihood-free MCMC with Amortized Approximate Ratio Estimators". en. In: *ICML* 37. URL: http://proceedings.mlr.press/v119/hermans20a/hermans20a.pdf.

References VII

Hull, Robert et al. (Nov. 2022). *Using simulation-based inference to determine the parameters of an integrated hydrologic model: a case study from the upper Colorado River basin*. en. preprint. Groundwater hydrology/Modelling approaches. DOI: 10.5194/hess-2022-345. URL: https://hess.copernicus.org/preprints/hess-2022-345/ (visited on 11/05/2023).

Lueckmann, Jan-Matthis, Jan Boelts, et al. (2021). "Benchmarking Simulation-Based Inference". en. In: URL: http://proceedings.mlr.press/v130/lueckmann21a/lueckmann21a.pdf.

References VIII

Lueckmann, Jan-Matthis, Pedro J. Goncalves, et al. (Nov. 2017). *Flexible statistical inference for mechanistic models of neural dynamics*. arXiv:1711.01861 [stat]. DOI: 10.48550/arXiv.1711.01861. URL: http://arxiv.org/abs/1711.01861 (visited on 11/06/2023).

Neyman, Jerzy, Egon Sharpe Pearson, and Karl Pearson (Feb. 1933). "IX. On the problem of the most efficient tests of statistical hypotheses". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231.694-706. Publisher: Royal Society, pp. 289–337. DOI: 10.1098/rsta.1933.0009. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.1933.0009 (visited on 11/05/2023).

References IX

Papamakarios, George and Iain Murray (Apr. 2018). *Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation*. arXiv:1605.06376 [cs, stat]. DOI: 10.48550/arXiv.1605.06376. URL: http://arxiv.org/abs/1605.06376 (visited on 11/06/2023).

Papamakarios, George, Theo Pavlakou, and Iain Murray (June 2018). *Masked Autoregressive Flow for Density Estimation*. arXiv:1705.07057 [cs, stat]. DOI: 10.48550/arXiv.1705.07057. URL: http://arxiv.org/abs/1705.07057 (visited on 11/06/2023).

References X

Papamakarios, George, David C. Sterratt, and Iain Murray (Jan. 2019). *Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows*. en. arXiv:1805.07226 [cs, stat]. URL: http://arxiv.org/abs/1805.07226 (visited on 11/05/2023).

Ramesh, Poornima et al. (Mar. 2022). *GATSBI: Generative Adversarial Training for Simulation-Based Inference*. en. arXiv:2203.06481 [cs, stat]. URL: http://arxiv.org/abs/2203.06481 (visited on 11/03/2023).

References XI

Rubin, Donald B. (Dec. 1984). "Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician". In: *The Annals of Statistics* 12.4. Publisher: Institute of Mathematical Statistics, pp. 1151–1172. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/aos/1176346785. URL: https://projecteuclid.org/journals/annals-of-statistics/volume-12/issue-4/Bayesianly-Justifiable-and-Relevant-Frequency-Calculations-for-the-Applied-Statistician/10.1214/aos/1176346785.full (visited on 11/05/2023).

References XII

Sharrock, Louis et al. (Nov. 2022). *Sequential Neural Score Estimation: Likelihood-Free Inference with Conditional Score Based Diffusion Models*. en. arXiv:2210.04872 [cs, stat]. URL: http://arxiv.org/abs/2210.04872 (visited on 11/05/2023).