Likelihood-Free Inference with Generative Neural Networks via Scoring Rule Minimization

Lorenzo Pacchiardi

Department of Statistics, University of Oxford

lorenzo.pacchiardi@stats.ox.ac.uk

ISBA 2022, 30th June 2022

Background: Likelihood-Free Inference

Likelihood-Free Inference (LFI)

Intractable-Likelihood model $P(\cdot|\boldsymbol{\theta})$

- **can simulate data**:

$$\boldsymbol{y} \sim P(\cdot|\boldsymbol{\theta}), \;\; \boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}^d$$

- cannot evaluate $p(\boldsymbol{y}|\boldsymbol{\theta})$

Likelihood-Free Inference (LFI)

Intractable-Likelihood model $P(\cdot|\boldsymbol{\theta})$

- **can simulate data**:

$$\boldsymbol{y} \sim P(\cdot|\boldsymbol{\theta}), \ \boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}^d$$

- cannot evaluate $p(\boldsymbol{y}|\boldsymbol{\theta})$

standard posterior $\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})$ is unaccessible!

Likelihood-Free Inference (LFI)

Intractable-Likelihood model $P(\cdot|\boldsymbol{\theta})$

- **can simulate data**:

$$\boldsymbol{y} \sim P(\cdot|\boldsymbol{\theta}), \;\; \boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}^d$$

- cannot evaluate $p(\boldsymbol{y}|\boldsymbol{\theta})$

standard posterior $\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})$ is unaccessible!

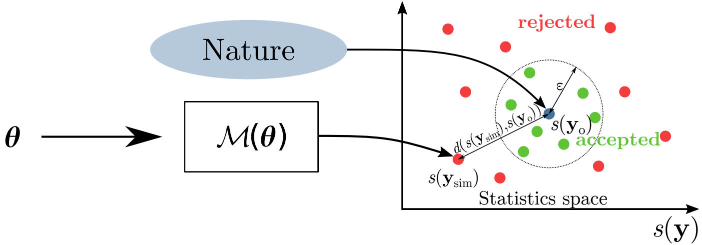
Likelihood-Free Inference (LFI) approaches:

- Allow to sample from approximate posterior,
- rely on **drawing simulations** from the model.

E.g.: Approximate Bayesian Computation

Iterate:

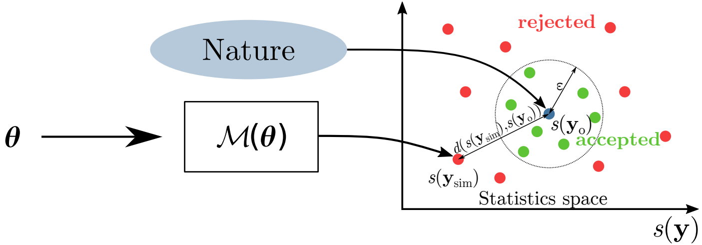
1. Draw $\boldsymbol{\theta}^{(j)} \sim \pi(\boldsymbol{\theta})$
2. Simulate a dataset $\boldsymbol{y}_{\text{sim}} \sim P(\cdot|\boldsymbol{\theta}^{(j)})$
3. Compute some **statistics** $s(\boldsymbol{y})$ of the simulated and observed datasets
4. If **distance** $d(s(\boldsymbol{y}_{\text{sim}}), s(\boldsymbol{y}_o)) \leq \epsilon$ (threshold) $\implies$ accept $\boldsymbol{\theta}^{(j)}$; otherwise, reject



E.g.: Approximate Bayesian Computation

Iterate:

1. Draw $\boldsymbol{\theta}^{(j)} \sim \pi(\boldsymbol{\theta})$
2. Simulate a dataset $\boldsymbol{y}_{\text{sim}} \sim P(\cdot|\boldsymbol{\theta}^{(j)})$
3. Compute some **statistics** $s(\boldsymbol{y})$ of the simulated and observed datasets
4. If **distance** $d(s(\boldsymbol{y}_{\text{sim}}), s(\boldsymbol{y}_o)) \leq \epsilon$ (threshold) $\implies$ accept $\boldsymbol{\theta}^{(j)}$; otherwise, reject



Accepted $\boldsymbol{\theta}^{(j)}$'s are distributed according to the ABC posterior:

$$\pi^{\epsilon}(\boldsymbol{\theta}|s(\boldsymbol{y}_o)) \propto \pi(\boldsymbol{\theta}) \int \mathbb{1}\left[d(s(\boldsymbol{y}_o), s(\boldsymbol{y}_{\text{sim}})) \leq \epsilon\right] p(\boldsymbol{y}_{\text{sim}}|\boldsymbol{\theta}) d\boldsymbol{y}_{\text{sim}}$$

Background: conditional generative networks

Conditional generative networks

Defined by:

- a neural network $f_\phi : \mathcal{Z} \times \mathcal{Y} \to \Theta$;
- a probability distribution $P_{\mathbf{z}}$ over the space $\mathcal{Z}$.

Conditional generative networks

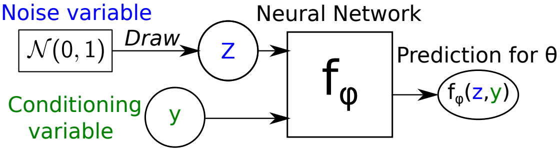
Defined by:

- a neural network $f_\phi : \mathcal{Z} \times \mathcal{Y} \to \Theta$;
- a probability distribution $P_{\boldsymbol{z}}$ over the space $\mathcal{Z}$.

These induce a distribution $Q_\phi(\cdot|\boldsymbol{y})$ over Θ by:

1. Sampling $\boldsymbol{z} \sim P_{\boldsymbol{z}}$,
2. computing $f_\phi(\boldsymbol{z}, \boldsymbol{y})$.

In statistical notation: $Q_\phi(\cdot|\boldsymbol{y}) = \underbrace{f_\phi(\cdot, \boldsymbol{y})\sharp P_{\boldsymbol{z}}}_{\text{“pushforward”}}$.



Conditional generative networks

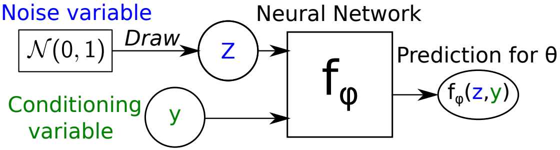
Defined by:

- a neural network $f_\phi : \mathcal{Z} \times \mathcal{Y} \to \Theta$;
- a probability distribution $P_{\boldsymbol{z}}$ over the space $\mathcal{Z}$.

These induce a distribution $Q_\phi(\cdot|\boldsymbol{y})$ over Θ by:

1. Sampling $\boldsymbol{z} \sim P_{\boldsymbol{z}}$,
2. computing $f_\phi(\boldsymbol{z}, \boldsymbol{y})$.

In statistical notation: $Q_\phi(\cdot|\boldsymbol{y}) = \underbrace{f_\phi(\cdot, \boldsymbol{y})\sharp P_{\boldsymbol{z}}}_{\text{“pushforward”}}$.



Idea:

- Use a **generative network to approximate the posterior**
- Can **easily obtain samples** from it (no MCMC)

Conditional generative networks

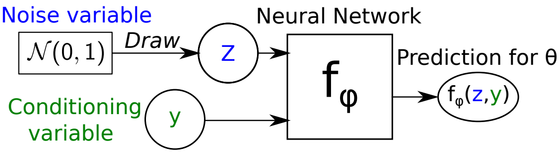
Defined by:

- a neural network $f_\phi : \mathcal{Z} \times \mathcal{Y} \to \Theta$;
- a probability distribution $P_{\boldsymbol{z}}$ over the space $\mathcal{Z}$.

These induce a distribution $Q_\phi(\cdot|\boldsymbol{y})$ over Θ by:

1. Sampling $\boldsymbol{z} \sim P_{\boldsymbol{z}}$,
2. computing $f_\phi(\boldsymbol{z}, \boldsymbol{y})$.

In statistical notation: $Q_\phi(\cdot|\boldsymbol{y}) = \underbrace{f_\phi(\cdot, \boldsymbol{y})\sharp P_{\boldsymbol{z}}}_{\text{“pushforward”}}$.



Idea:

- Use a **generative network to approximate the posterior**
- Can **easily obtain samples** from it (no MCMC)
- But you **can't evaluate the density**: how can you train it?

Posterior inference via Generative Adversarial Networks (Ramesh et al., 2022)

Posterior via Generative Adversarial Networks (GANs) (Ramesh et al., 2022)

- Consider a **discriminator** neural network $D_\psi : \Theta \times \mathcal{Y} \to [0, 1]$
- Define loss (Goodfellow et al., 2014):

$$L(\phi, \psi) := \mathbb{E}_{\boldsymbol{Y} \sim P} \left[\mathbb{E}_{\boldsymbol{\Theta} \sim \Pi(\cdot|\boldsymbol{Y})} (\log D_\psi(\boldsymbol{\Theta}, \boldsymbol{Y})) + \mathbb{E}_{\tilde{\boldsymbol{\Theta}} \sim Q_\phi(\cdot|\boldsymbol{Y})} \left(\log \left(1 - D_\psi(\tilde{\boldsymbol{\Theta}}, \boldsymbol{Y}) \right) \right) \right]$$

Posterior via Generative Adversarial Networks (GANs) (Ramesh et al., 2022)

- Consider a **discriminator** neural network $D_\psi : \Theta \times \mathcal{Y} \to [0, 1]$
- Define loss (Goodfellow et al., 2014):

$$L(\phi, \psi) := \mathbb{E}_{\boldsymbol{Y} \sim P} \left[\mathbb{E}_{\boldsymbol{\Theta} \sim \Pi(\cdot|\boldsymbol{Y})} (\log D_\psi(\boldsymbol{\Theta}, \boldsymbol{Y})) + \mathbb{E}_{\tilde{\boldsymbol{\Theta}} \sim Q_\phi(\cdot|\boldsymbol{Y})} \left(\log \left(1 - D_\psi(\tilde{\boldsymbol{\Theta}}, \boldsymbol{Y}) \right) \right) \right]$$

- Solving $\min_\phi \max_\psi L(\phi, \psi) \implies Q_\phi(\cdot|\boldsymbol{y}) = \Pi(\cdot|\boldsymbol{y}) \ \forall \ \boldsymbol{y} : p(\boldsymbol{y}) > 0$

Posterior via Generative Adversarial Networks (GANs) (Ramesh et al., 2022)

- Consider a **discriminator** neural network $D_\psi : \Theta \times \mathcal{Y} \to [0,1]$
- Define loss (Goodfellow et al., 2014):

$$L(\phi,\psi) := \mathbb{E}_{\boldsymbol{Y}\sim P}\left[\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})}\left(\log D_\psi(\boldsymbol{\Theta},\boldsymbol{Y})\right) + \mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}\left(\log\left(1 - D_\psi(\tilde{\boldsymbol{\Theta}},\boldsymbol{Y})\right)\right)\right]$$

- Solving $\min_\phi \max_\psi L(\phi,\psi) \implies Q_\phi(\cdot|\boldsymbol{y}) = \Pi(\cdot|\boldsymbol{y})\ \forall\ \boldsymbol{y} : p(\boldsymbol{y}) > 0$

Adversarial training

$$\boxed{\min_{\phi}\max_{\psi}\mathbb{E}_{\boldsymbol{Y}\sim P}\left[\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})}\left(\log D_\psi(\boldsymbol{\Theta},\boldsymbol{Y})\right)+\mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}\left(\log\left(1-D_\psi(\tilde{\boldsymbol{\Theta}},\boldsymbol{Y})\right)\right)\right]}$$

1. Generate a dataset of **parameter-simulations pairs** from the likelihood-free model:

$$(\boldsymbol{\theta}_i,\boldsymbol{y}_i)_{i=1}^n, \boldsymbol{\theta}_i\sim\Pi \text{ and } \boldsymbol{y}_i\sim P(\cdot|\boldsymbol{\theta}_i)$$

Adversarial training

$$\boxed{\min_{\phi} \max_{\psi} \mathbb{E}_{\boldsymbol{Y}\sim P}\left[\mathbb{E}_{\boldsymbol{\Theta}\sim \Pi(\cdot|\boldsymbol{Y})}\left(\log D_\psi(\boldsymbol{\Theta}, \boldsymbol{Y})\right) + \mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}\left(\log\left(1 - D_\psi(\tilde{\boldsymbol{\Theta}}, \boldsymbol{Y})\right)\right)\right]}$$

1. Generate a dataset of **parameter-simulations pairs** from the likelihood-free model:

$$(\boldsymbol{\theta}_i, \boldsymbol{y}_i)_{i=1}^n, \boldsymbol{\theta}_i \sim \Pi \text{ and } \boldsymbol{y}_i \sim P(\cdot|\boldsymbol{\theta}_i) \iff \boldsymbol{y}_i \sim P \text{ and } \boldsymbol{\theta}_i \sim \Pi(\cdot|\boldsymbol{y}_i)$$

Adversarial training

$$\boxed{\min_{\phi}\max_{\psi}\mathbb{E}_{\boldsymbol{Y}\sim P}\left[\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})}\left(\log D_\psi(\boldsymbol{\Theta},\boldsymbol{Y})\right)+\mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}\left(\log\left(1-D_\psi(\tilde{\boldsymbol{\Theta}},\boldsymbol{Y})\right)\right)\right]}$$

1. Generate a dataset of **parameter-simulations pairs** from the likelihood-free model:

$$(\boldsymbol{\theta}_i,\boldsymbol{y}_i)_{i=1}^n, \boldsymbol{\theta}_i\sim\Pi \text{ and } \boldsymbol{y}_i\sim P(\cdot|\boldsymbol{\theta}_i) \iff \boldsymbol{y}_i\sim P \text{ and } \boldsymbol{\theta}_i\sim\Pi(\cdot|\boldsymbol{y}_i)$$

2. Train by **alternating** stochastic gradient ascent/descent over ψ and ϕ:

Require: Generative net f_ϕ, discriminator D_ψ, learning rates ϵ, γ.

1: **for** each training pair $(\boldsymbol{\theta}_i,\boldsymbol{y}_i)$ **do**
2: Sample $\boldsymbol{z}_i\sim P_{\boldsymbol{z}}$
3: Obtain $\tilde{\boldsymbol{\theta}}_i^\phi=f_\phi(\boldsymbol{z}_i,\boldsymbol{y}_i)$
4: Set $\psi\leftarrow\psi+\gamma\cdot\nabla_\psi\left[\log D_\psi(\boldsymbol{\theta}_i,\boldsymbol{y}_i)+\log(1-D_\psi(\tilde{\boldsymbol{\theta}}_i^\phi,\boldsymbol{y}_i))\right]$
5: Set $\phi\leftarrow\phi-\epsilon\cdot\nabla_\phi\left[\log(1-D_\psi(\tilde{\boldsymbol{\theta}}_i^\phi,\boldsymbol{y}_i))\right]$
6: **end for**

- Use batch of training data to estimate $\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})}\cdots$
- and **draws from the generative network** to estimate $\mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}$

Adversarial training

$$\boxed{\min_{\phi}\max_{\psi}\mathbb{E}_{\boldsymbol{Y}\sim P}\left[\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})}\left(\log D_\psi(\boldsymbol{\Theta},\boldsymbol{Y})\right)+\mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}\left(\log\left(1-D_\psi(\tilde{\boldsymbol{\Theta}},\boldsymbol{Y})\right)\right)\right]}$$

1. Generate a dataset of **parameter-simulations pairs** from the likelihood-free model:

$$(\boldsymbol{\theta}_i,\boldsymbol{y}_i)_{i=1}^n, \boldsymbol{\theta}_i\sim\Pi \text{ and } \boldsymbol{y}_i\sim P(\cdot|\boldsymbol{\theta}_i) \iff \boldsymbol{y}_i\sim P \text{ and } \boldsymbol{\theta}_i\sim\Pi(\cdot|\boldsymbol{y}_i)$$

2. Train by **alternating** stochastic gradient ascent/descent over ψ and ϕ:

Require: Generative net f_ϕ, discriminator D_ψ, learning rates ϵ, γ.
1: **for** each training pair $(\boldsymbol{\theta}_i,\boldsymbol{y}_i)$ **do**
2: Sample $\boldsymbol{z}_i\sim P_{\boldsymbol{z}}$
3: Obtain $\tilde{\boldsymbol{\theta}}_i^\phi=f_\phi(\boldsymbol{z}_i,\boldsymbol{y}_i)$
4: Set $\psi\leftarrow\psi+\gamma\cdot\nabla_\psi\left[\log D_\psi(\boldsymbol{\theta}_i,\boldsymbol{y}_i)+\log(1-D_\psi(\tilde{\boldsymbol{\theta}}_i^\phi,\boldsymbol{y}_i))\right]$
5: Set $\phi\leftarrow\phi-\epsilon\cdot\nabla_\phi\left[\log(1-D_\psi(\tilde{\boldsymbol{\theta}}_i^\phi,\boldsymbol{y}_i))\right]$
6: **end for**

- Use batch of training data to estimate $\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})}\cdots$
- and **draws from the generative network** to estimate $\mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}$

3. For a real observation $\boldsymbol{y}_o$, you can directly get samples from $\Pi(\cdot|\boldsymbol{y}_o)$.

Adversarial training

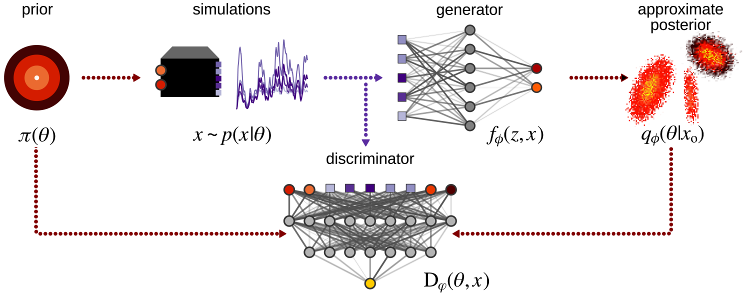


Figure: From Ramesh et al. (2022)

Issues with adversarial training

$$\boxed{\min_{\phi} \max_{\psi} \mathbb{E}_{\boldsymbol{Y} \sim P} \left[\mathbb{E}_{\boldsymbol{\Theta} \sim \Pi(\cdot|\boldsymbol{Y})} \left(\log D_\psi(\boldsymbol{\Theta}, \boldsymbol{Y}) \right) + \mathbb{E}_{\tilde{\boldsymbol{\Theta}} \sim Q_\phi(\cdot|\boldsymbol{Y})} \left(\log \left(1 - D_\psi(\tilde{\boldsymbol{\Theta}}, \boldsymbol{Y}) \right) \right) \right]}$$

Issues with adversarial training

$$\boxed{\min_{\phi} \max_{\psi} \mathbb{E}_{\boldsymbol{Y}\sim P} \left[\mathbb{E}_{\boldsymbol{\Theta}\sim \Pi(\cdot|\boldsymbol{Y})} \left(\log D_\psi(\boldsymbol{\Theta}, \boldsymbol{Y})\right) + \mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})} \left(\log \left(1 - D_\psi(\tilde{\boldsymbol{\Theta}}, \boldsymbol{Y})\right)\right)\right]}$$

Issues:

1. **Unstable** min-max problem.

Issues with adversarial training

$$\boxed{\min_{\phi} \max_{\psi} \mathbb{E}_{\boldsymbol{Y}\sim P}\left[\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})}\left(\log D_\psi(\boldsymbol{\Theta},\boldsymbol{Y})\right)+\mathbb{E}_{\tilde{\boldsymbol{\Theta}}\sim Q_\phi(\cdot|\boldsymbol{Y})}\left(\log\left(1-D_\psi(\tilde{\boldsymbol{\Theta}},\boldsymbol{Y})\right)\right)\right]}$$

Issues:

1. **Unstable** min-max problem.
2. **biased gradients** w.r.t. ϕ when:
   - doing few updates for ψ
   - and using it to estimate gradients w.r.t. ϕ

Issues with adversarial training

$$\boxed{\min_{\phi} \max_{\psi} \mathbb{E}_{\boldsymbol{Y} \sim P} \left[\mathbb{E}_{\boldsymbol{\Theta} \sim \Pi(\cdot|\boldsymbol{Y})} \left(\log D_{\psi}(\boldsymbol{\Theta}, \boldsymbol{Y}) \right) + \mathbb{E}_{\tilde{\boldsymbol{\Theta}} \sim Q_{\phi}(\cdot|\boldsymbol{Y})} \left(\log \left(1 - D_{\psi}(\tilde{\boldsymbol{\Theta}}, \boldsymbol{Y}) \right) \right) \right]}$$

Issues:

1. **Unstable** min-max problem.
2. **biased gradients** w.r.t. ϕ when:
   - doing few updates for ψ
   - and using it to estimate gradients w.r.t. ϕ
3. **Mode collapse**.

Posterior inference via Scoring Rules Minimization for Generative Networks

Scoring Rules

$S(Q_\phi, \boldsymbol{\theta}) \rightarrow$ "*penalty* when stating a distribution Q_ϕ for a realization $\boldsymbol{\theta}$".

Scoring Rules

$S(Q_\phi, \boldsymbol{\theta}) \rightarrow$ "*penalty* when stating a distribution Q_ϕ for a realization $\boldsymbol{\theta}$".

Expected Scoring Rule: if $\boldsymbol{\Theta} \sim \Pi$:

$$S(Q_\phi, \Pi) := \mathbb{E}_{\boldsymbol{\Theta} \sim \Pi} S(Q_\phi, \boldsymbol{\Theta}).$$

Scoring Rules

$S(Q_\phi, \boldsymbol{\theta}) \rightarrow$ "*penalty* when stating a distribution Q_ϕ for a realization $\boldsymbol{\theta}$".

Expected Scoring Rule: if $\boldsymbol{\Theta} \sim \Pi$:

$$S(Q_\phi, \Pi) := \mathbb{E}_{\boldsymbol{\Theta} \sim \Pi} S(Q_\phi, \boldsymbol{\Theta}).$$

Strictly proper S: $S(Q_\phi, \Pi)$ is uniquely minimized by $Q_\phi = \Pi$.

Some strictly proper scoring rules

Energy Score:

$$S_{\mathsf{E}}^{(\beta)}(Q_\phi, \boldsymbol{\theta}) = 2 \cdot \mathbb{E}\left[\|\tilde{\boldsymbol{\Theta}} - \boldsymbol{\theta}\|_2^\beta\right] - \mathbb{E}\left[\|\tilde{\boldsymbol{\Theta}} - \tilde{\boldsymbol{\Theta}}'\|_2^\beta\right], \quad \tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}' \sim Q_\phi,$$

Kernel Score: related to MMD^2

$$S_k(Q_\phi, \boldsymbol{\theta}) = \mathbb{E}[k(\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}')] - 2 \cdot \mathbb{E}[k(\tilde{\boldsymbol{\Theta}}, \boldsymbol{\theta})], \quad \tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}' \sim Q_\phi.$$

$k \rightarrow$ positive definite kernel.

Conditional Generative Networks via Scoring Rule Minimization

- Define:

$$\boxed{J(\phi) := \mathbb{E}_{\boldsymbol{Y} \sim P} \mathbb{E}_{\boldsymbol{\Theta} \sim \Pi(\cdot|\boldsymbol{Y})} S(Q_\phi(\cdot|\boldsymbol{Y}), \boldsymbol{\Theta})}$$

- For **strictly proper** S, $\min_\phi J(\phi) \implies Q_\phi(\cdot|\boldsymbol{y}) = \Pi(\cdot|\boldsymbol{y}) \ \forall \ \boldsymbol{y} : p(\boldsymbol{y}) > 0$.

Conditional Generative Networks via Scoring Rule Minimization

- Define:

$$\boxed{J(\phi) := \mathbb{E}_{\boldsymbol{Y}\sim P}\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})} S(Q_\phi(\cdot|\boldsymbol{Y}), \boldsymbol{\Theta})}$$

- For **strictly proper** S, $\min_\phi J(\phi) \implies Q_\phi(\cdot|\boldsymbol{y}) = \Pi(\cdot|\boldsymbol{y})\ \forall\ \boldsymbol{y} : p(\boldsymbol{y}) > 0$.

Unbiased empirical estimate:

$$\operatorname*{arg\,min}_\phi \left[\hat{J}(\phi) := \frac{1}{n}\sum_{i=1}^n S(Q_\phi(\cdot|\boldsymbol{y}_i), \boldsymbol{\theta}_i)\right], \quad \boldsymbol{\theta}_i \sim \Pi \text{ and } \boldsymbol{y}_i \sim P(\cdot|\boldsymbol{\theta}_i),$$

Conditional Generative Networks via Scoring Rule Minimization

- Define:

$$\boxed{J(\phi) := \mathbb{E}_{\boldsymbol{Y}\sim P}\mathbb{E}_{\boldsymbol{\Theta}\sim\Pi(\cdot|\boldsymbol{Y})} S(Q_\phi(\cdot|\boldsymbol{Y}), \boldsymbol{\Theta})}$$

- For **strictly proper** S, $\min_\phi J(\phi) \implies Q_\phi(\cdot|\boldsymbol{y}) = \Pi(\cdot|\boldsymbol{y})\ \forall\ \boldsymbol{y} : p(\boldsymbol{y}) > 0$.

Unbiased empirical estimate:

$$\arg\min_\phi \left[\hat{J}(\phi) := \frac{1}{n}\sum_{i=1}^{n} S(Q_\phi(\cdot|\boldsymbol{y}_i), \boldsymbol{\theta}_i) \right], \quad \boldsymbol{\theta}_i \sim \Pi \text{ and } \boldsymbol{y}_i \sim P(\cdot|\boldsymbol{\theta}_i),$$

$\implies$ Unbiased estimate of $\nabla_\phi S(Q_\phi(\cdot|\boldsymbol{y}_i), \boldsymbol{\theta}_i)$ **is enough** to train via SGD.

How to get unbiased gradient estimates

We **need unbiased estimate** of $\nabla_\phi S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta})$.

How to get unbiased gradient estimates

We **need unbiased estimate** of $\nabla_\phi S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta})$.

- We take $S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta}) = \mathbb{E}_{\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}' \sim Q_\phi(\cdot|\boldsymbol{y})} \left[g(\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}', \boldsymbol{\theta}) \right]$ for some function g

How to get unbiased gradient estimates

We **need unbiased estimate** of $\nabla_\phi S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta})$.

- We take $S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta}) = \mathbb{E}_{\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}' \sim Q_\phi(\cdot|\boldsymbol{y})} \left[g(\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}', \boldsymbol{\theta}) \right]$ for some function $g \implies$ we can get **unbiased estimator from draws** $\tilde{\boldsymbol{\Theta}}_j^\phi = f_\phi(\mathbf{Z}_j, \boldsymbol{y}) \sim Q_\phi(\cdot|\boldsymbol{y}),\ j = 1, \ldots, m$:

$$\hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \frac{1}{m(m-1)} \sum_{j \neq k} g(\tilde{\boldsymbol{\Theta}}_j^\phi, \tilde{\boldsymbol{\Theta}}_k^\phi, \boldsymbol{\theta})$$

How to get unbiased gradient estimates

We **need unbiased estimate** of $\nabla_\phi S(Q_\phi(\cdot|\mathbf{y}), \boldsymbol{\theta})$.

- We take $S(Q_\phi(\cdot|\mathbf{y}), \boldsymbol{\theta}) = \mathbb{E}_{\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}' \sim Q_\phi(\cdot|\mathbf{y})} \left[g(\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}', \boldsymbol{\theta}) \right]$ for some function $g \implies$ we can get **unbiased estimator from draws** $\tilde{\boldsymbol{\Theta}}_j^\phi = f_\phi(\mathbf{Z}_j, \mathbf{y}) \sim Q_\phi(\cdot|\mathbf{y}),\ j = 1, \ldots, m$:

$$\hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \frac{1}{m(m-1)} \sum_{j \neq k} g(\tilde{\boldsymbol{\Theta}}_j^\phi, \tilde{\boldsymbol{\Theta}}_k^\phi, \boldsymbol{\theta})$$

- With autodifferentiation: obtain $\nabla_\phi \hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \nabla_\phi \hat{S}(\{f_\phi(\mathbf{Z}_j, \mathbf{y})\}_j, \boldsymbol{\theta})$.

How to get unbiased gradient estimates

We **need unbiased estimate** of $\nabla_\phi S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta})$.

- We take $S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta}) = \mathbb{E}_{\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}' \sim Q_\phi(\cdot|\boldsymbol{y})} \left[g(\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}', \boldsymbol{\theta}) \right]$ for some function $g \implies$ we can get **unbiased estimator from draws** $\tilde{\boldsymbol{\Theta}}_j^\phi = f_\phi(\mathbf{Z}_j, \boldsymbol{y}) \sim Q_\phi(\cdot|\boldsymbol{y}),\ j = 1, \ldots, m$:

$$\hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \frac{1}{m(m-1)} \sum_{j \neq k} g(\tilde{\boldsymbol{\Theta}}_j^\phi, \tilde{\boldsymbol{\Theta}}_k^\phi, \boldsymbol{\theta})$$

- With autodifferentiation: obtain $\nabla_\phi \hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \nabla_\phi \hat{S}(\{f_\phi(\mathbf{Z}_j, \boldsymbol{y})\}_j, \boldsymbol{\theta})$.
- But (swapping expectation and gradient):

$$\mathbb{E} \nabla_\phi \hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \nabla_\phi \mathbb{E} \hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \nabla_\phi S(Q_\phi, \boldsymbol{\theta})$$

How to get unbiased gradient estimates

We **need unbiased estimate** of $\nabla_\phi S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta})$.

- We take $S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta}) = \mathbb{E}_{\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}' \sim Q_\phi(\cdot|\boldsymbol{y})} \left[g(\tilde{\boldsymbol{\Theta}}, \tilde{\boldsymbol{\Theta}}', \boldsymbol{\theta}) \right]$ for some function $g \implies$ we can get **unbiased estimator from draws** $\tilde{\boldsymbol{\Theta}}_j^\phi = f_\phi(\mathbf{Z}_j, \boldsymbol{y}) \sim Q_\phi(\cdot|\boldsymbol{y}), \ j = 1, \ldots, m$:

$$\hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \frac{1}{m(m-1)} \sum_{j \neq k} g(\tilde{\boldsymbol{\Theta}}_j^\phi, \tilde{\boldsymbol{\Theta}}_k^\phi, \boldsymbol{\theta})$$

- With autodifferentiation: obtain $\nabla_\phi \hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \nabla_\phi \hat{S}(\{f_\phi(\mathbf{Z}_j, \boldsymbol{y})\}_j, \boldsymbol{\theta})$.
- But (swapping expectation and gradient):

$$\mathbb{E} \nabla_\phi \hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \nabla_\phi \mathbb{E} \hat{S}(\{\tilde{\boldsymbol{\Theta}}_j^\phi\}_j, \boldsymbol{\theta}) = \nabla_\phi S(Q_\phi, \boldsymbol{\theta})$$

∇_ϕ of unbiased estimate of $S \equiv$ unbiased estimate of $\nabla_\phi S$

SR minimization

Require: Generative net f_ϕ, SR S, learning rate ϵ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**

end for

Adversarial training

Require: Generative net f_ϕ, discriminator D_ψ, learning rates ϵ, γ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**

end for

SR minimization

Require: Generative net f_ϕ, SR S, learning rate ϵ.

```
for each training pair (θ_i, y_i) do
    Sample multiple z_{i,1}, ..., z_{i,m} ~ P_z
end for
```

Adversarial training

Require: Generative net f_ϕ, discriminator D_ψ, learning rates ϵ, γ.

```
for each training pair (θ_i, y_i) do
    Sample z_i ~ P_z
end for
```

SR minimization

Require: Generative net f_ϕ, SR S, learning rate ϵ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**

  Sample **multiple** $\mathrm{z}_{i,1}, \ldots, \mathrm{z}_{i,m} \sim P_{\boldsymbol{z}}$

  Obtain $\tilde{\boldsymbol{\theta}}_{i,j}^{\phi} = f_\phi(\mathrm{z}_{i,j}, \boldsymbol{y}_i)$

end for

Adversarial training

Require: Generative net f_ϕ, discriminator D_ψ, learning rates ϵ, γ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**

  Sample $\boldsymbol{z}_i \sim P_{\boldsymbol{z}}$

  Obtain $\tilde{\boldsymbol{\theta}}_i^{\phi} = f_\phi(\boldsymbol{z}_i, \boldsymbol{y}_i)$

end for

SR minimization

Require: Generative net f_ϕ, SR S, learning rate ϵ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**

  Sample **multiple** $z_{i,1}, \dots, z_{i,m} \sim P_{\boldsymbol{z}}$

  Obtain $\tilde{\boldsymbol{\theta}}_{i,j}^{\phi} = f_\phi(z_{i,j}, \boldsymbol{y}_i)$

  Compute unbiased estimate $\hat{S}(\{\boldsymbol{\theta}_{i,j}^{\phi}\}_j, \boldsymbol{\theta}_i)$ from $\tilde{\boldsymbol{\theta}}_{i,j}^{\phi}$

  Set $\phi \leftarrow \phi - \epsilon \cdot \nabla_\phi \hat{S}(\{\boldsymbol{\theta}_{i,j}^{\phi}\}_j, \boldsymbol{\theta}_i)$

end for

Adversarial training

Require: Generative net f_ϕ, discriminator D_ψ, learning rates ϵ, γ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**

  Sample $\boldsymbol{z}_i \sim P_{\boldsymbol{z}}$

  Obtain $\tilde{\boldsymbol{\theta}}_{i}^{\phi} = f_\phi(\boldsymbol{z}_i, \boldsymbol{y}_i)$

  Set $\psi \leftarrow \psi + \gamma \cdot \nabla_\psi \Big[\log D_\psi(\boldsymbol{\theta}_i, \boldsymbol{y}_i) + \log(1 - D_\psi(\tilde{\boldsymbol{\theta}}_{i}^{\phi}, \boldsymbol{y}_i)) \Big]$

  Set $\phi \leftarrow \phi - \epsilon \cdot \nabla_\phi \Big[\log(1 - D_\psi(\tilde{\boldsymbol{\theta}}_{i}^{\phi}, \boldsymbol{y}_i)) \Big]$

end for

SR minimization

Require: Generative net f_ϕ, SR S, learning rate ϵ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**
    Sample **multiple** $\mathrm{z}_{i,1}, \ldots, \mathrm{z}_{i,m} \sim P_{\boldsymbol{z}}$
    Obtain $\tilde{\boldsymbol{\theta}}_{i,j}^{\phi} = f_\phi(\mathrm{z}_{i,j}, \boldsymbol{y}_i)$
    Compute unbiased estimate $\hat{S}(\{\boldsymbol{\theta}_{i,j}^{\phi}\}_j, \boldsymbol{\theta}_i)$ from $\tilde{\boldsymbol{\theta}}_{i,j}^{\phi}$
    Set $\phi \leftarrow \phi - \epsilon \cdot \nabla_\phi \hat{S}(\{\boldsymbol{\theta}_{i,j}^{\phi}\}_j, \boldsymbol{\theta}_i)$
end for

Adversarial training

Require: Generative net f_ϕ, discriminator D_ψ, learning rates ϵ, γ.

for each training pair $(\boldsymbol{\theta}_i, \boldsymbol{y}_i)$ **do**
    Sample $\boldsymbol{z}_i \sim P_{\boldsymbol{z}}$
    Obtain $\tilde{\boldsymbol{\theta}}_i^{\phi} = f_\phi(\boldsymbol{z}_i, \boldsymbol{y}_i)$
    Set $\psi \leftarrow \psi + \gamma \cdot \nabla_\psi \left[\log D_\psi(\boldsymbol{\theta}_i, \boldsymbol{y}_i) + \log(1 - D_\psi(\tilde{\boldsymbol{\theta}}_i^{\phi}, \boldsymbol{y}_i)) \right]$
    Set $\phi \leftarrow \phi - \epsilon \cdot \nabla_\phi \left[\log(1 - D_\psi(\tilde{\boldsymbol{\theta}}_i^{\phi}, \boldsymbol{y}_i)) \right]$
end for

Simpler training than GAN:

- **no discriminator** D_ψ
- **no min-max** objective

Need multiple simulations from Q_ϕ, but **good results with as little as 3**.

Simulations: noisy camera model

$\boldsymbol{\theta} \in \mathbb{R}^{28\times 28} \to$ EMNIST dataset, $\boldsymbol{y}_o \in \mathbb{R}^{28\times 28}$ generated from $\boldsymbol{\theta}$ by blurring.

Simulations: noisy camera model

$\boldsymbol{\theta} \in \mathbb{R}^{28\times 28} \rightarrow$ EMNIST dataset, $\boldsymbol{y}_o \in \mathbb{R}^{28\times 28}$ generated from $\boldsymbol{\theta}$ by blurring.

Table: Training time (secs)

GAN	45398
Energy	22633
Kernel	22545

Conclusions

- **Bayesian Likelihood-Free Inference with generative networks used to approximate the posterior.**
- Alternative training method which...
  - ... is **faster** than adversarial,
  - does **not suffer from instabilities** of min-max training
  - and leads to **better results**.

L. Pacchiardi and R. Dutta. Likelihood-free inference with generative neural networks via scoring rule minimization. *arXiv preprint* **arXiv:2205.15784**, 2022.

References

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

L. Pacchiardi, R. Adewoyin, P. Dueben, and R. Dutta. Probabilistic forecasting with conditional generative networks via scoring rule minimization. *arXiv preprint arXiv:2112.08217*, 2022.

P. Ramesh, J.-M. Lueckmann, J. Boelts, Á. Tejero-Cantero, D. S. Greenberg, P. J. Goncalves, and J. H. Macke. GATSBI: Generative adversarial training for simulation-based inference. In *International Conference on Learning Representations*, 2022.

Contacts

website: www.lorenzopacchiardi.me email: *lorenzo.pacchiardi@stats.ox.ac.uk*

Backup slides

Patched SR

- If $\boldsymbol{X}$ has some structure (say, it is on a 1D or 2D grid) raw SR discard that information.
- $\implies$ compute the SR on localized *patches* across the grid and cumulate the score; in this way, short-scale correlations are given more importance.
- The resulting SR is non-strictly proper $\implies$ add the global SR to make it strictly proper.

The patched SR is:

$$S_p(P, \boldsymbol{x}) = w_1 S(P, \boldsymbol{x}) + w_2 \sum_{p \in \mathcal{P}} S(P|_p, \boldsymbol{x}|_p),$$

where $w_1, w_2 > 0$, $|_p$ denotes the restriction of a distribution or of a vector to a patch p and $\mathcal{P}$ is a set of patches.

Connection with normalizing flows

- Normalizing flows = generative networks with invertible $f_\phi(\boldsymbol{z}, \boldsymbol{y})$ with respect to $\boldsymbol{z}$.
- Density evaluation is possible via change-of-variables formula $\implies$ ϕ is usually trained via maximum likelihood; e.g.:

$$\begin{aligned}
&\operatorname*{argmin}_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{Y}\sim P}\left[\mathbb{KL}\left(\Pi(\cdot \mid \boldsymbol{Y}) \| Q_{\boldsymbol{\phi}}(\cdot \mid \boldsymbol{Y})\right)\right] \\
&= \operatorname*{argmin}_{\phi} \mathbb{E}_{\boldsymbol{Y}\sim P}\mathbb{E}_{\boldsymbol{\theta}\sim\Pi(\cdot|\boldsymbol{Y})}\left[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{Y})\right] \\
&= \operatorname*{argmin}_{\phi} \mathbb{E}_{\boldsymbol{\theta}\sim\Pi}\mathbb{E}_{\boldsymbol{Y}\sim P(\cdot|\boldsymbol{\theta})}\left[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{Y})\right],
\end{aligned}$$

which corresponds to our SR-based approach by identifying $S(Q_\phi(\cdot|\boldsymbol{y}), \boldsymbol{\theta}) = -\log q_\phi(\boldsymbol{\theta}|\boldsymbol{y})$, which is the strictly-proper logarithmic scoring rule.

Additional results for noisy camera model

Table: Noisy Camera model: performance metrics, runtime and early stopping epoch for GAN and for the Energy and Kernel Score with patch size 8 and step 5. The latter methods achieved better performance with shorter training time. All methods are trained on a single GPU.

	RMSE ↓	Cal. Err. ↓	R^2 ↑	Runtime (sec)	Early stopping epoch
GAN	0.25 ± 0.19	0.50 ± 0.00	-23.94 ± 366.08	45398	3600
Energy	0.06 ± 0.05	0.36 ± 0.12	-2.14 ± 55.86	22633	4000
Kernel	0.07 ± 0.05	0.36 ± 0.12	-10.29 ± 222.12	22545	3200