

# Simulation-based Inference

## Inverting Simulators

Talk by Stefan Wezel

mlcolab @ Tübingen University Cluster of Excellence

April 29, 2021

- The problem setting
- Traditional approaches and their issues
- Using Bayesian Neural Nets to alleviate them
- A worked example
- A more interesting, real-world example
- Outlook

# What and Why?

## An Example

- Imagine you are a astronomer
- You observe planetary movements
- and find that two parameters determine the movement (mass and velocity) (and some noise)
- So you build a mechanistic model that can simulate observations
- But what do we actually care about?
  - What was the mass and velocity of that planet

# What and Why?

## Formalizing our Example

- Mechanistic model (simulator) has parameters  $\theta$  and produces data  $\hat{x}$ 
  - It implicitly defines  $p(\hat{x}|\theta)$  (it yields samples from this distribution)
- Given a (real) observation  $x_0$ , we would be interested in the parameters that produced it
  - $p(\theta|x_0)$  -> posterior over parameters
  - In simple cases, we could apply Bayes' rule and compute it analytically
  - What is simple? If we know Likelihood
  - But often we would not

# What and Why?

## Simulation-based Inference to the rescue

- This is exactly a problem setting that scientists often deal with
  - They have a sophisticated model (the simulator) that encapsulates a lot of prior knowledge
  - But inference in this setting often boils down to: What were parameters that produced this observation
- Simulation-based Inference inference tries to solve this problem
- by inverting the simulator

# Simulation-based Inference

## Traditional Approach

- Broader term: Approximate Bayesian Computation (ABC)
  - Rejection ABC
  - Sampling ABC (perturbe initial params)
  - Sequential ABC
  - But gives only point estimates, not full posterior
  - within  $\epsilon$

# Learning a Posterior

## Bayesian Neural Nets to the Rescue

- Create  $n$  training samples
- learn posterior over parameters by parametrizing GMM with DNN

# How to use SBI?

Some advertisement

- Now that you are pretty excited about SBI and its applications
- mlcolab and mackelab is developing a python (and eventually Julia) library (that I've used earlier)
- If you are interested: try it out and give us your feedback
- link to colab
- link to github

The screenshot shows the PyPI page for the `sbi` package, version 0.15.1. The header is blue with the package name and version. Below the header, there's a dark blue bar with the command `pip install sbi` and a 'Latest version' button. The main content area is white and divided into two columns. The left column, titled 'Navigation', contains links for 'Project description', 'Release history', and 'Download files'. The right column, titled 'Project description', contains a horizontal bar with various badges (Python package, contributions, tests, coverage, license, and ACPL) and the text 'sbi: simulation-based inference'.

**sbi 0.15.1** [Latest version](#)

`pip install sbi`

Released: Mar 18, 2021

Simulation-based inference.

**Navigation**

- [Project description](#)
- [Release history](#)
- [Download files](#)

**Project description**

python package 0.15.1 contributions welcome tests passing coverage 88% license ACPL 3.0

PSS: 110-21105pass-02105

sbi: simulation-based inference



○

- Finding parameters/posterior over parameters with
  - Rejection ABC
  - Markov Chain Monte Carlo ABC
  - Sequential Monte Carlo ABC
  - ...
- Problems
  -

# What

is Simulation-based Inference?

- If we know generating factors, we can build a simulator
- But we want to constrain simulator output on observations from real world
- Thus we need realistic values for simulator parameters
- -> inverse problem
- SBI solves this inverse problem using Bayesian inference

# How

does Simulation-based Inference work?