Lecture Script

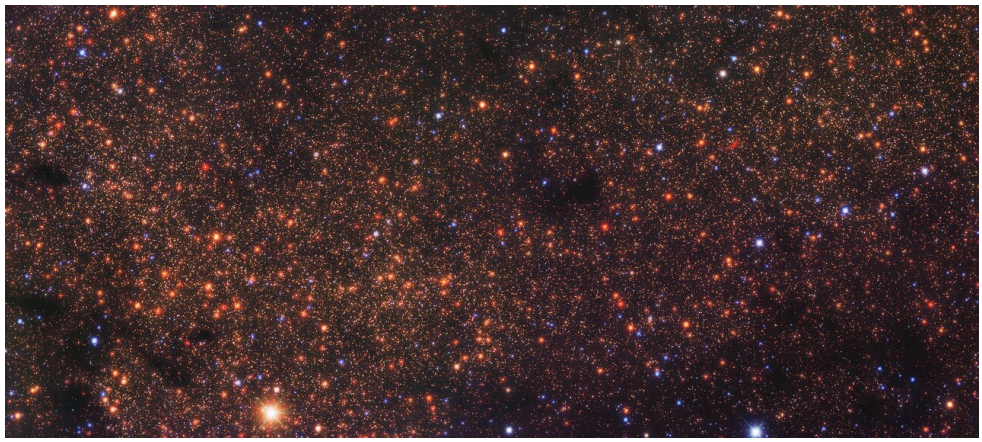
Simulation-Based Inference

Outline

1. x, θ, z

- Last lecture, we connected simulators to data through the likelihood $p(x \mid \theta)$
- If the likelihood is tractable, we can do inference with MCMC or variational methods
- Often, the likelihood is intractable because of complex latent/nuisance parameters z (stochastic structure of the forward model).
- Here, we can still run the simulator forward—sample $x \sim p(x \mid \theta)$—but we can't evaluate $p(x \mid \theta)$.
- We'll see how to do inference in this setting with **simulation-based inference (SBI)**.

2. The running example: point sources



VLT/HAWK-I infrared image of Sagittarius C near the Milky Way centre (ESO).

- Setting: Telescope observing a patch of sky
- Image contains **point sources** (e.g., stars), each with a flux S_i and position $\mathbf{r}_i$
- Detector bins photon arrivals into pixels and blurs with a point spread function (finite resolution)

The physics question: what is the **source-count distribution**?

$$\frac{dN}{dS} = A \cdot S^n$$

- Parameters of interest are $\theta = (A, n)$—the amplitude and slope of the power law.
- These control how many sources there are and how their brightnesses are distributed.
- Latent variables are the number of sources N, their fluxes $\{S_i\}$, and positions $\{\mathbf{r}_i\}$.
- We'll use this same setup at increasing complexity, and watch the likelihood go from tractable to intractable.

Known N: tractable

Suppose we know $N = 5$. Maybe we also know their positions. The forward model maps sources to a predicted rate μ_p per pixel, and data is a Poisson realization:

$$p(x \mid z, \theta) = \prod_{\text{pixels } p} \text{Poisson}(x_p;\ \mu_p)$$

If positions are unknown, they become **nuisance parameters** $z = \{S_i, \mathbf{r}_i\}_{i=1}^{5}$:

$$\underbrace{(A, n)}_{\theta} \ \rightarrow \ \underbrace{\{S_i, \mathbf{r}_i\}_{i=1}^{5}}_{z} \ \rightarrow \ x$$

The marginal likelihood integrates them out:

$$p(x \mid \theta) = \int p(x \mid z, \theta)\, p(z \mid \theta)\, dz$$

But we don't need to compute this integral. Same idea as the hierarchical models from L12—sample the **joint** $p(\theta, z \mid x)$ with MCMC:

$$p(\theta, z \mid x) \propto \underbrace{p(x \mid z, \theta)}_{\text{tractable}} \cdot p(z \mid \theta) \cdot p(\theta)$$

- MCMC explores the full (θ, z) space
- To get the marginal on θ, throw away the z samples
- Marginalization is a byproduct of sampling
- Works because z is low-dimensional (17 dims) and $p(x \mid z, \theta)$ is tractable

Unknown N: intractable

Now we don't know how many sources there are. N is itself random:

$$N \sim \text{Poisson}\left(\int_{S_{\min}}^{S_{\max}} A \cdot S^{n}\, dS\right)$$

Each source: $S_i \sim \text{PowerLaw}(A, n)$, $\mathbf{r}_i \sim \text{Uniform}(\text{sky})$. The latent space:

$$z = \left(N,\ \{S_i, \mathbf{r}_i\}_{i=1}^{N}\right)$$

This is **variable-dimensional**—different runs have different numbers of sources.

To evaluate $p(x \mid \theta)$, sum over all possible N and integrate over all configurations:

$$p(x \mid \theta) = \sum_{N=0}^{\infty} p(N \mid \theta) \int \prod_{i=1}^{N} p(S_i \mid \theta)\, p(\mathbf{r}_i) \cdot p(x \mid \{S_i, \mathbf{r}_i\}, \theta)\ d\mathbf{r}_{1:N}\, dS_{1:N}$$

This is, for all practical purposes, intractable!

Could we still sample the joint with MCMC? Three things go wrong:

1. The dimension of z changes with N.
2. Even at fixed N, the space grows. With hundreds of sources, the joint is very high-dimensional.
3. Source labels are exchangeable. Swapping $(S_1, \mathbf{r}_1)$ and $(S_2, \mathbf{r}_2)$ gives the same map. This permutation symmetry creates $N!$ equivalent modes.

While we can't evaluate $p(x \mid \theta)$, we *can* run the simulator:

```
x = simulate_sources(A=200, n=-1.2)
```

Sample from $p(x \mid \theta)$ without evaluating it—this is the **implicit likelihood**.

3. Approximate Bayesian computation

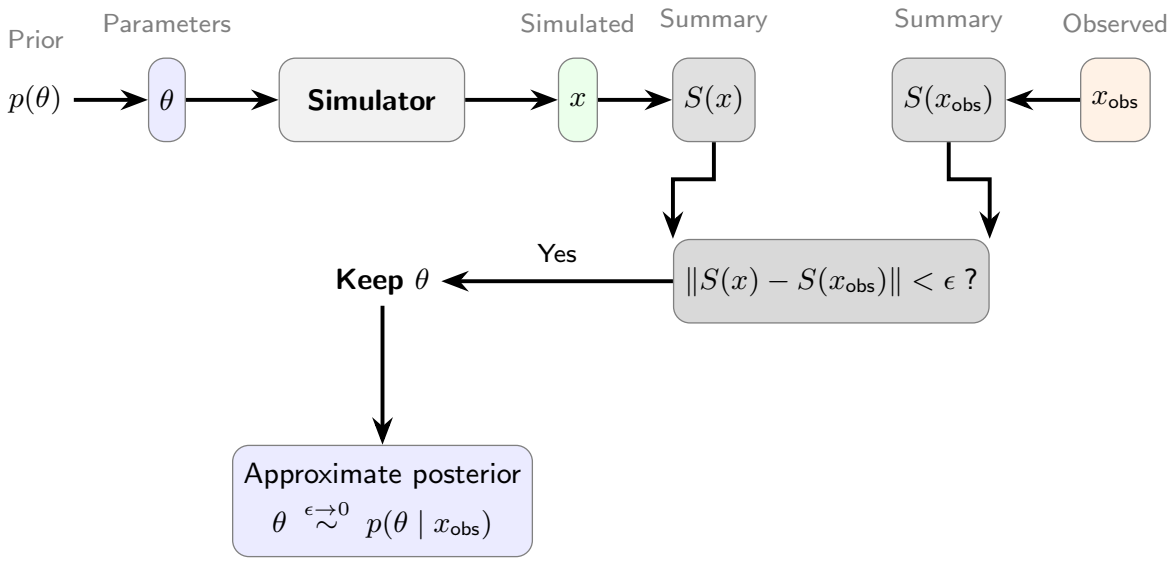
The idea

Keep parameter samples that produce simulated data close to the observed data.

Approximate Bayesian computation — rejection sampling:

1. Draw $\theta \sim p(\theta)$
2. Simulate $x_{\text{sim}} \sim p(x \mid \theta)$
3. If $x_{\text{sim}} \approx x_{\text{obs}}$, accept θ; otherwise reject
4. Repeat

Accepted samples approximate the posterior. If we could check $x_{\text{sim}} = x_{\text{obs}}$ exactly, this gives exact samples from $p(\theta \mid x_{\text{obs}})$. In continuous data, exact matches never happen, so we use a data-similarity measure.



Two compromises

Summary statistics. Comparing full maps pixel by pixel is hopeless. Compress to a low-dimensional summary $S(x)$:

$$\rho\big(S(x_{\text{sim}}),\ S(x_{\text{obs}})\big) < \epsilon$$

For our maps, S might be total flux, number of bright pixels, pixel intensity histogram. Unless S is **sufficient**, we lose information.

Tolerance ϵ. How close is close enough?

- $\epsilon \to 0$: exact posterior, but acceptance rate $\to 0$
- ϵ large: lots of accepted samples, but the posterior is smeared out

ABC doesn't scale!

Other problems:

- Low acceptance rate for reasonable ϵ
- New data x'_{obs} means starting from scratch
- Choice of ρ and ϵ is ad hoc

4. Neural posterior estimation (NPE)

The reframing

Instead of reject/accept one at a time, run the simulator many times up front, collect a training set, and learn the posterior.

Step 1: Simulate. Generate pairs from the joint:

$$\{(\theta_i, x_i)\}_{i=1}^{N}, \qquad \theta_i \sim p(\theta), \quad x_i \sim p(x \mid \theta_i)$$

Step 2: Learn. Train a conditional density estimator (a generative model) $q_\phi(\theta \mid x)$. This is just density estimation—fit q_ϕ to place high probability on the true θ_i given x_i:

$$\mathcal{L}(\phi) = -\frac{1}{N}\sum_{i=1}^{N} \log q_\phi(\theta_i \mid x_i)$$

Same loss as fitting any conditional density model. With enough data and capacity, $q_\phi(\theta \mid x) \to p(\theta \mid x)$.

Step 3: Infer. Given a new observation x_{obs}, evaluate $q_\phi(\theta \mid x_{\text{obs}})$ in a single forward pass. Instant posterior.

Architecture

We need $q_\phi(\theta \mid x)$ to be a flexible conditional density we can sample from and evaluate. Two components:

1. **Featurizer** $f_\psi(x)$: compresses data into a context vector c using a neural network appropriate for the data modality.
2. **Conditional density estimator** $q_\phi(\theta \mid c)$: takes context, outputs a distribution over θ. Simplest option: multivariate Gaussian.

Both trained end-to-end. The featurizer **learns its own summary statistics**.

Amortization

Once trained, the posterior conditional density estimator can be reused for any new observation. The cost of training is **amortized**—pay an upfront cost, then reuse for free.
