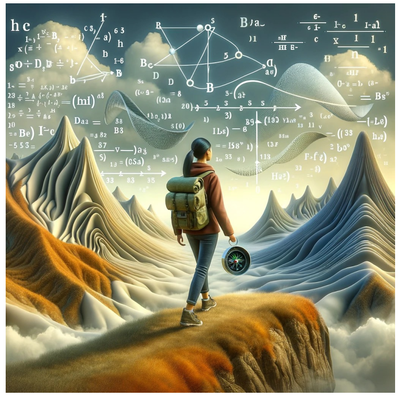
Simulation-based inference



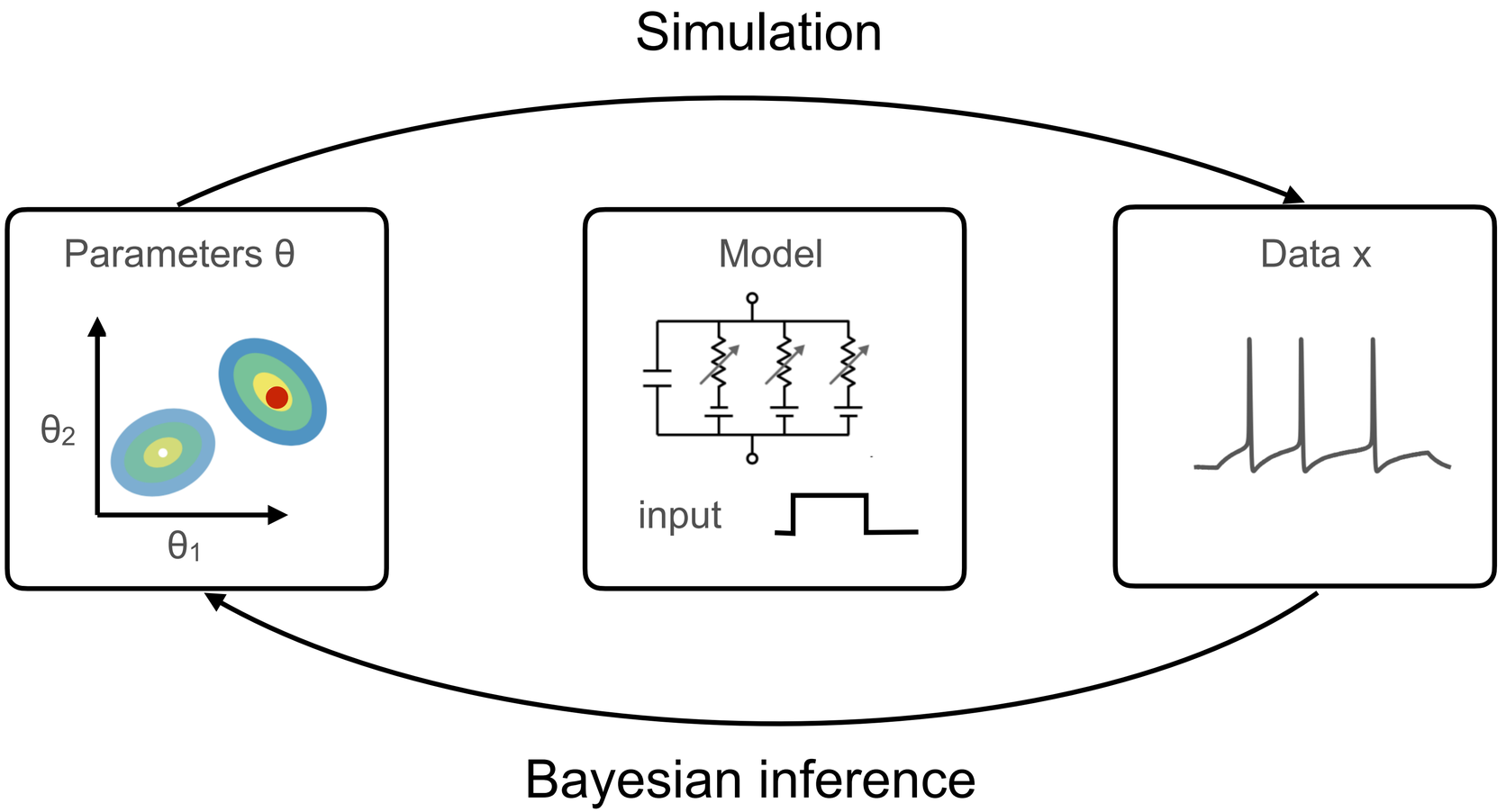
Lecture 2: Neural Posterior Estimation

April 2024

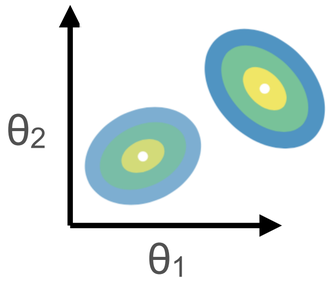
Pedro Gonçalves, Anastasia Krouglova
goncalveslab.sites.vib.be/en

Guy Moss
mackelab.org

Bayesian inference finds model-parameters which are consistent with data and prior knowledge



$$p(\theta \mid x) \propto p(x \mid \theta)\pi(\theta)$$



$$p(\theta \mid x) \propto p(x \mid \theta)\pi(\theta)$$

posterior: hard to compute

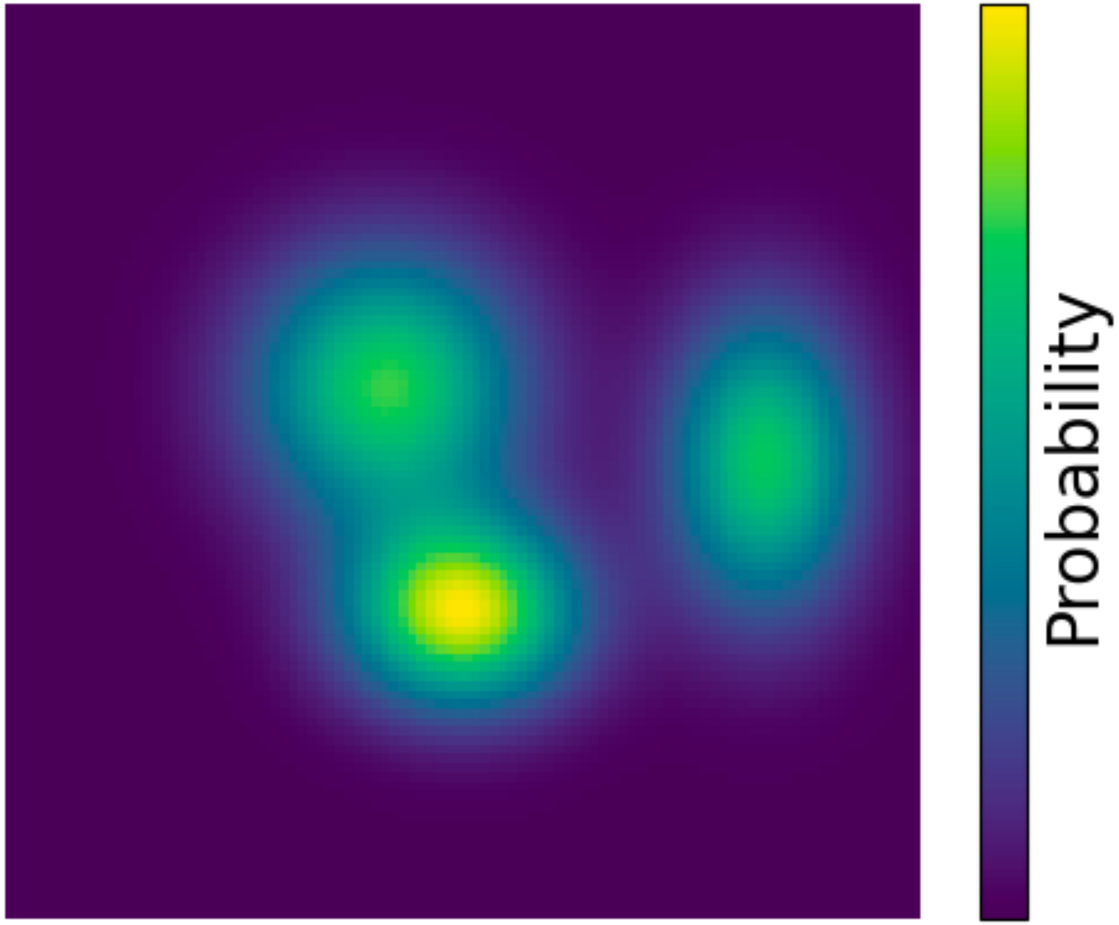
likelihood: often not tractable

For many mechanistic models, we can **simulate x**, but we cannot (easily) evaluate the likelihood **p(x|θ)**.

Models often defined through **black-box** simulators.

➡ A solution: simulation-based inference!

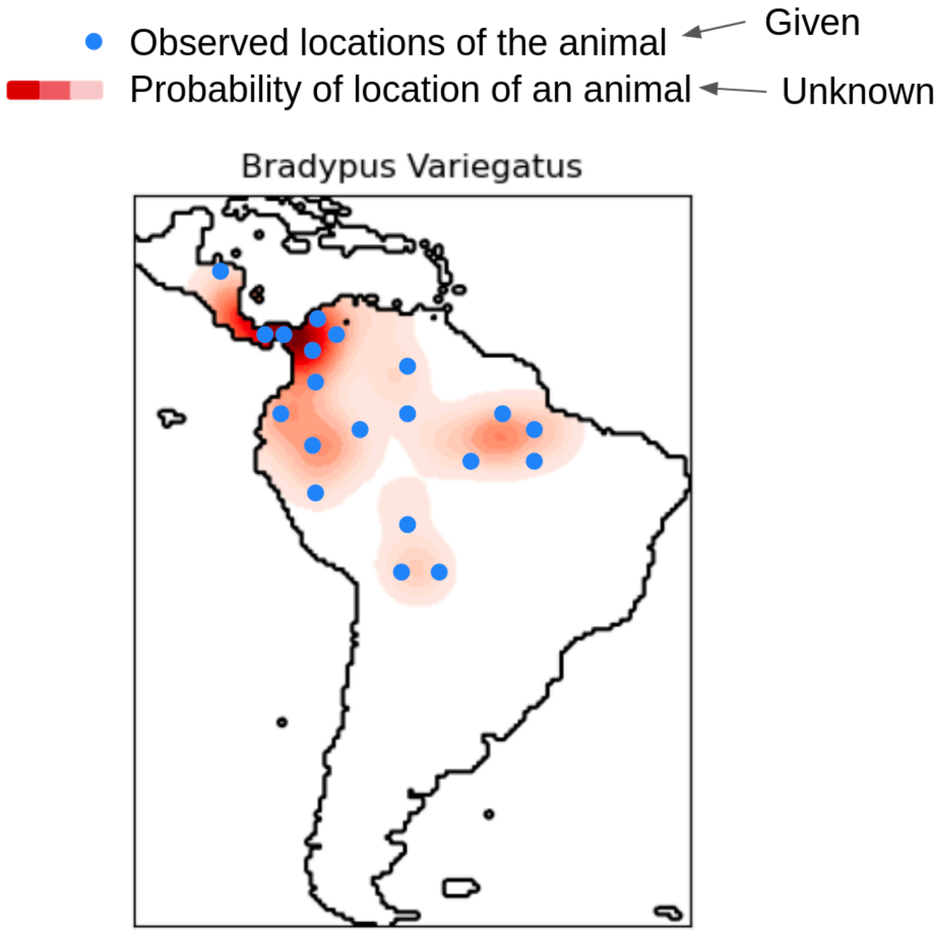
2.1 Learning distributions



Example: estimating the spread of a species

Brown-throated sloth
(Bradypus Variegatus)

https://en.wikipedia.org/wiki/Brown-throated_sloth



https://scikit-learn.org/stable/auto_examples/neighbors/plot_species_kde.html

The task

More formal:

Suppose we have samples $X = \{x_1, x_2, \ldots, x_N\}$ from an unknown distribution $p(x)$.

We want to approximate this distribution with some (normalised) function $f_\phi(x) \approx p(x)$.

1. What function could we choose as $f_\phi(x)$?

2. What does "$\approx$" mean here?

For now: “$\approx$” means Maximum Likelihood Estimation (MLE)

We estimate the parameters of f_ϕ by maximising the likelihood of the data assuming that f_ϕ is the true model:

$$\phi^* = \arg\max \Pi_i f_\phi(x_i)$$

Intuition: the data we observe should be the most likely outcome of the model

(*Later on:* Bayesian inference, where we include prior knowledge)

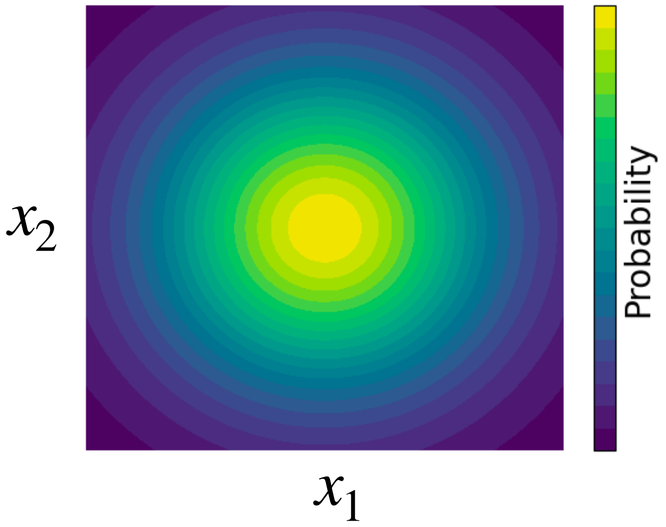
Example: f_ϕ as Gaussian Likelihood

Probability density function (pdf) of the Gaussian distribution with mean μ and covariance matrix Σ:

$$\mathcal{N}(x \mid \mu, \Sigma) = \frac{\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{\sqrt{(2\pi)^k \det(\Sigma)}},$$

where k is the dimension of the data x.

What is ϕ here?



MLE for Gaussian Distribution

Maximise the likelihood of the data X: $\Pi_i \mathcal{N}(x_i \mid \mu, \Sigma)$

Equivalent: Maximize log-likelihood (*why?*):

$$\Sigma_{i=1}^{N} \log(\mathcal{N}(x_i \mid \mu, \Sigma)) = \sum_{i=1}^{N} \log \left[\frac{\exp(-0.5(x_i - \mu)^T \Sigma^{-1}(x_i - \mu))}{\sqrt{(2\pi)^k \det(\Sigma)}} \right]$$

$$= -\frac{N}{2} \log((2\pi)^k \mid \det \Sigma \mid) - \frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^\top \Sigma^{-1}(x_i - \mu))$$

MLE for 1D Gaussian

We restrict the derivation to 1D Gaussians here for simplicity, but this proof can be extended to any dimension k

$$\mathscr{L}(X) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i - \mu)^2$$

MLE for mean

To find the MLE for the mean, we take the partial derivative of the likelihood and set to 0:

$$\frac{\partial}{\partial \mu}\mathscr{L} = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i - \mu) := 0$$

Which yields: $\mu_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N} x_i$

This is the mean of the data!

MLE for variance

To find the MLE for the variance, we take the partial derivative of the likelihood and set to 0:

$$\frac{\partial \mathscr{L}}{\partial \sigma^2} = -\frac{N}{2}\frac{2\pi}{2\pi\sigma^2} - \sum_{i=1}^{N}\frac{-1}{2\sigma^4}(x_i - \mu)^2$$

$$= -\frac{1}{2\sigma^2}\left(N - \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_n - \mu)^2\right)$$

Substituting $\mu_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and rearranging gives:

$\sigma^2_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{MLE})^2$, which is the variance of the data!

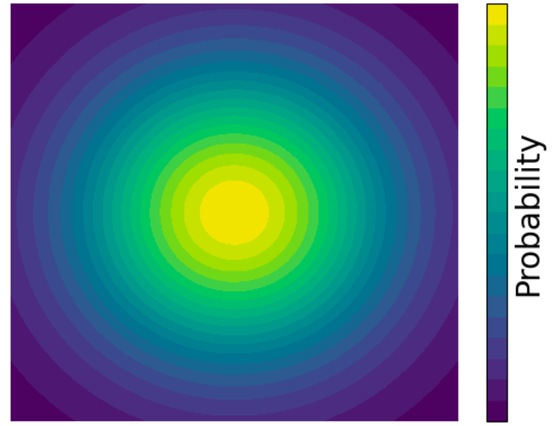
Recap: MLE for multivariate Gaussians

For higher dimensional Gaussians, the same considerations (with more algebra) lead to the MLE estimates for mean and covariance just being the mean and covariance of the data:

$$\mu_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i,$$

$$\Sigma_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}_i - \mu_{\text{MLE}})(\mathbf{x}_i - \mu_{\text{MLE}})^\top$$

Gaussians



Gaussians can be good fits to:

- *Unimodal* data (distributions with one peak)
- *Symmetric* data (distribution has circular/ellipse shape)

Gaussians are **not** good fit to:

- *Multimodal* data (distributions with many separate peaks)
- *Asymmetric* data (distributions that have “weird” shapes)

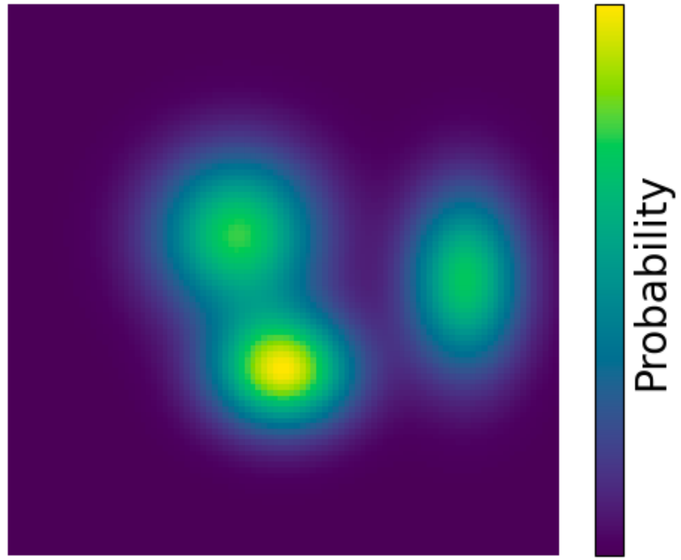
Mixture of Gaussians

The pdf for a **Mixture of Gaussians (MoG)** distribution is given by

$$p(x \mid \alpha_1, \ldots, \alpha_C, \mu_1, \ldots \mu_C, \Sigma_1, \ldots \Sigma_C)$$
$$= \alpha_1 \mathcal{N}(x_1 \mid \mu_1, \Sigma_1) + \ldots + \alpha_C \mathcal{N}(x \mid \mu_C, \Sigma_C)$$

Where we have C components with weights α_i, means μ_i, and covariances Σ_i.

What is ϕ here?



MLE for Mixture of Gaussians

The MLE is not analytically derivable in the case of a mixture of Gaussians.

But, we can fit the parameters $\alpha_1, \ldots, \alpha_C, \mu_1, \ldots \mu_C, \Sigma_1, \ldots \Sigma_C$ by minimising the negative log likelihood (NLL) (with gradient descent).

But is this enough?

Imagine running a taxi company

Image from Dutordoir et al. 2018

What is the probability of a taxi's drop-off location given the pick-up location?

We can model this as a **conditional probability**:

$$p(\text{drop-off} \mid \text{pick-up}) = p(x \mid \theta)$$

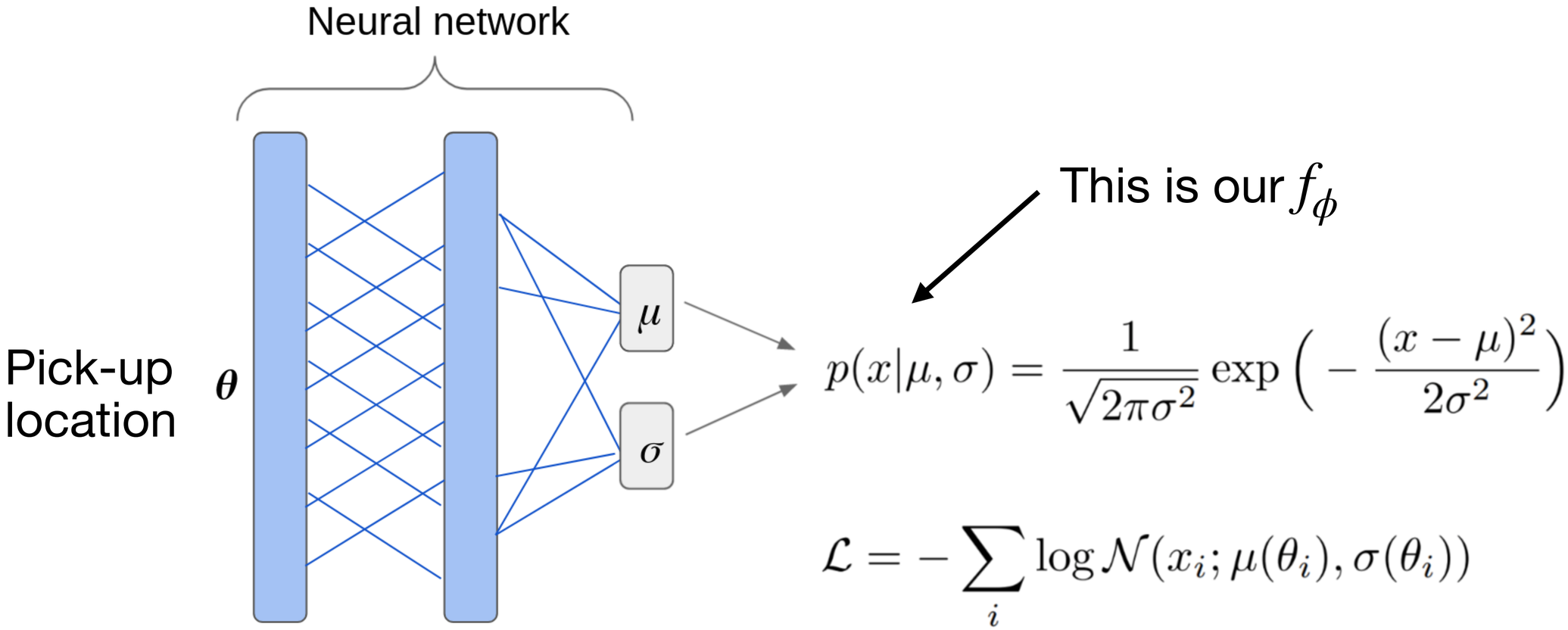
Estimating conditional distributions

Paired data:

$\mathscr{D} = \{(x_i, \theta_i)\}_{i \in I}$ (e.g. x = drop-off, θ = pick-up location)

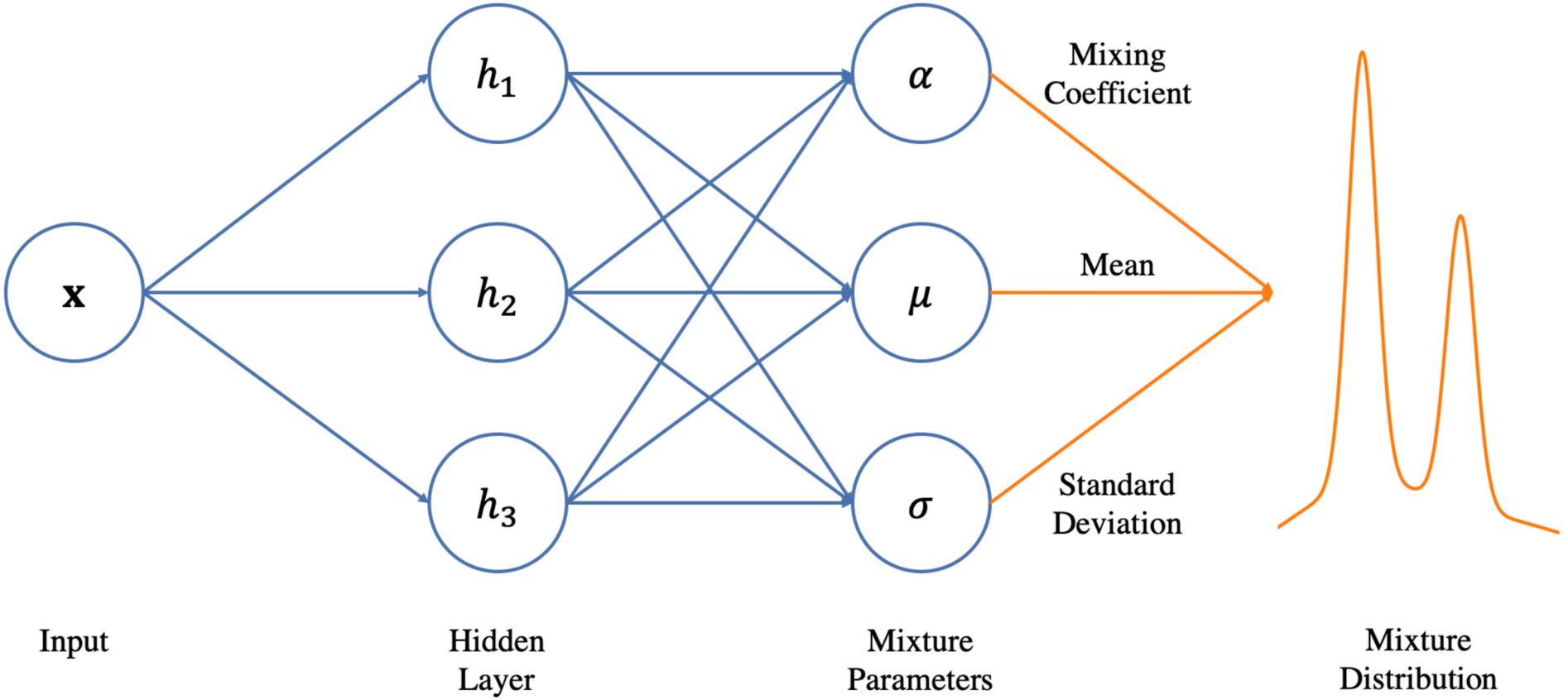
In the same logic as before, we want to maximise the log-likelihood $\sum_{i=1}^{N} \log(f_\phi(x_i \mid \theta_i))$ for some (normalised) function f_ϕ.

Gaussian density network…



Which parameters do we optimise here?

… and Gaussian mixture density networks (MDN)

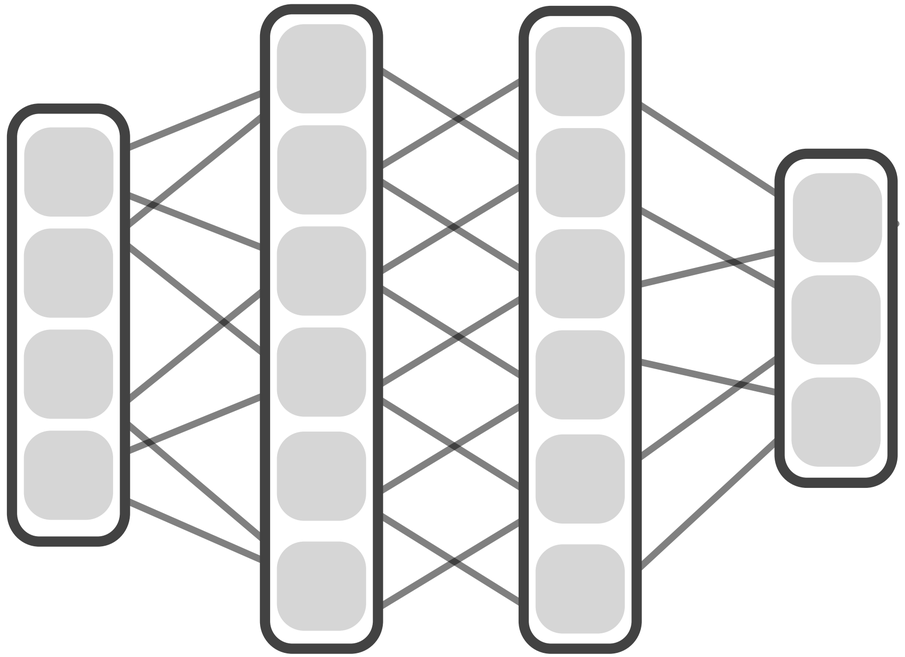


Interim summary

Interim summary

- We can approximate a distribution $p(x)$ from which we have samples $X_1, \ldots, X_n$, by maximising the (log) likelihood $f_\phi(X_1, \ldots, x_n)$ for some parametrised (and normalised) function $f_\phi(x)$
- Common choices for $f_\phi(x)$ are Gaussian or mixture of Gaussian (MoG) distributions
- MoG can capture multimodal data
- Mixture density networks are a way to parametrise the parameters of a MoG
- The same framework can be applied to conditional distributions

2.2 Neural Posterior Estimation



Back to simulators

- Assuming we have an observation x_o and a simulator, parameterised by θ from which we can draw samples $x \sim p(x \mid \theta)$.
- *Question*: Which parameters could have generated x_o?
- *Formal:* What is the posterior $p(\theta \mid x_o)$?
- *Simple solution:* ABC

Ball-throwing example

θ_1 Initial speed

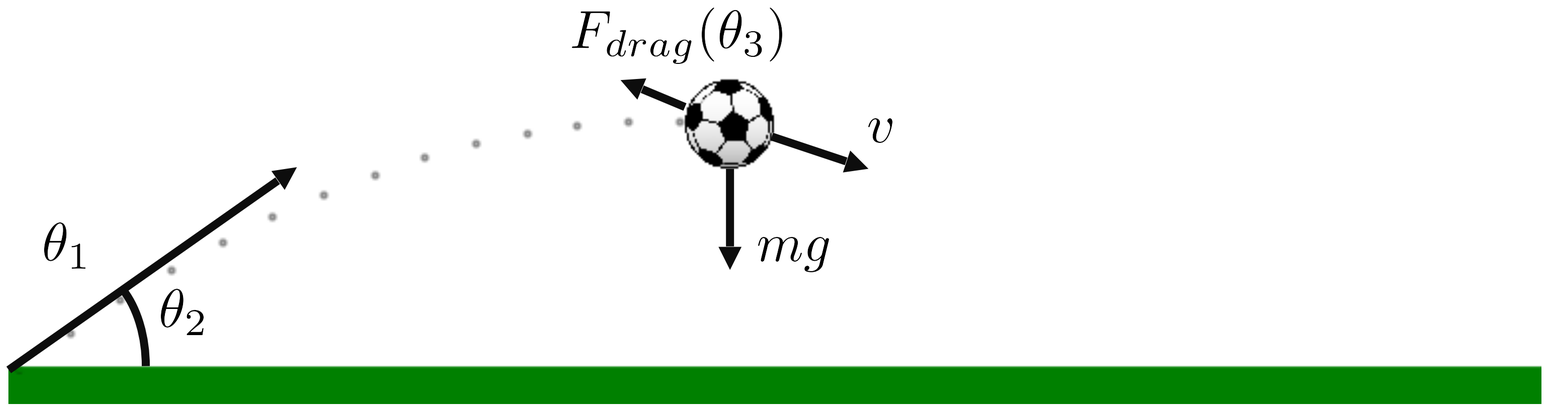
θ_2 Throwing angle

Ball-throwing example

θ_1 Initial speed

θ_2 Throwing angle

θ_3 Drag



Ball-throwing example

There are many possible trajectories…

Ball-throwing example

Assuming that we have observed one throw x_o, what are the possible parameters?

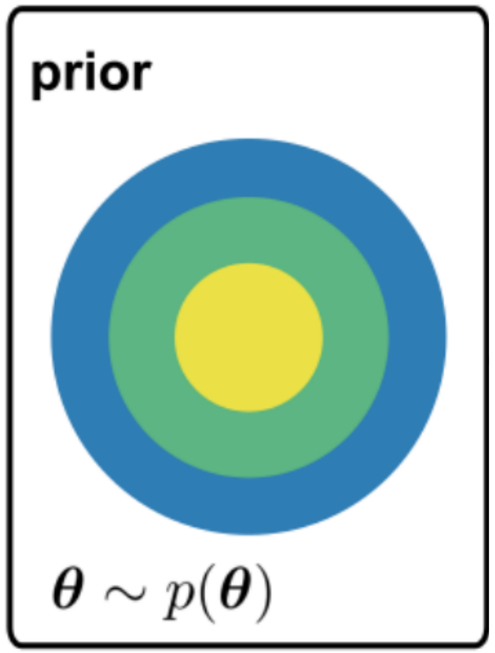
So, what is the posterior distribution $p(\theta \mid x_o)$?

Idea of Neural Posterior Estimation (NPE)

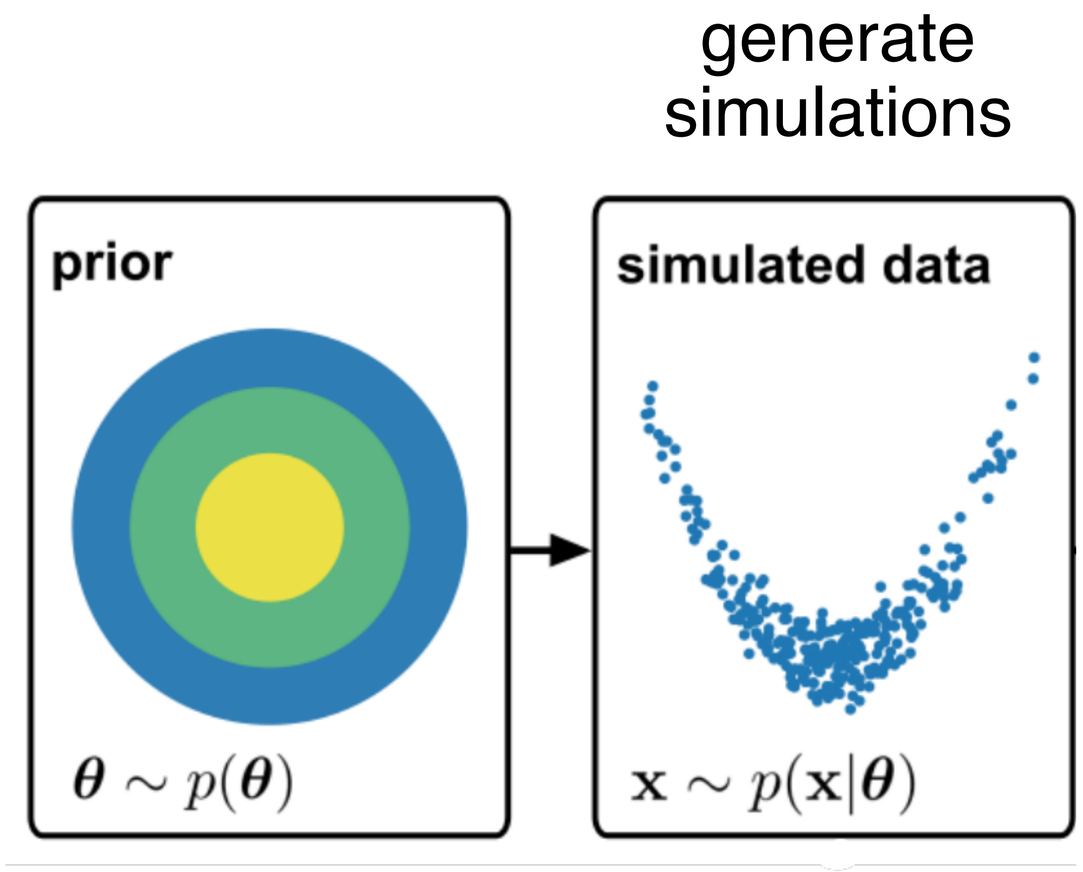
- Instead of accepting/rejecting “good” simulations as in ABC, we generate a *paired* training dataset $\mathscr{D} = \{(x_i, \theta_i)\}_{i=1,\dots,N}$ by running the simulator N times.
- We then learn a conditional density estimator $q_\phi(\theta \mid x)$ to approximate the true density $p(\theta \mid x)$.
- In the end, we evaluate q_ϕ at x_o to obtain the approximate posterior for the empirical observation: $p(\theta \mid x_o) \approx q_\phi(\theta \mid x_o)$.

NPE: step 1

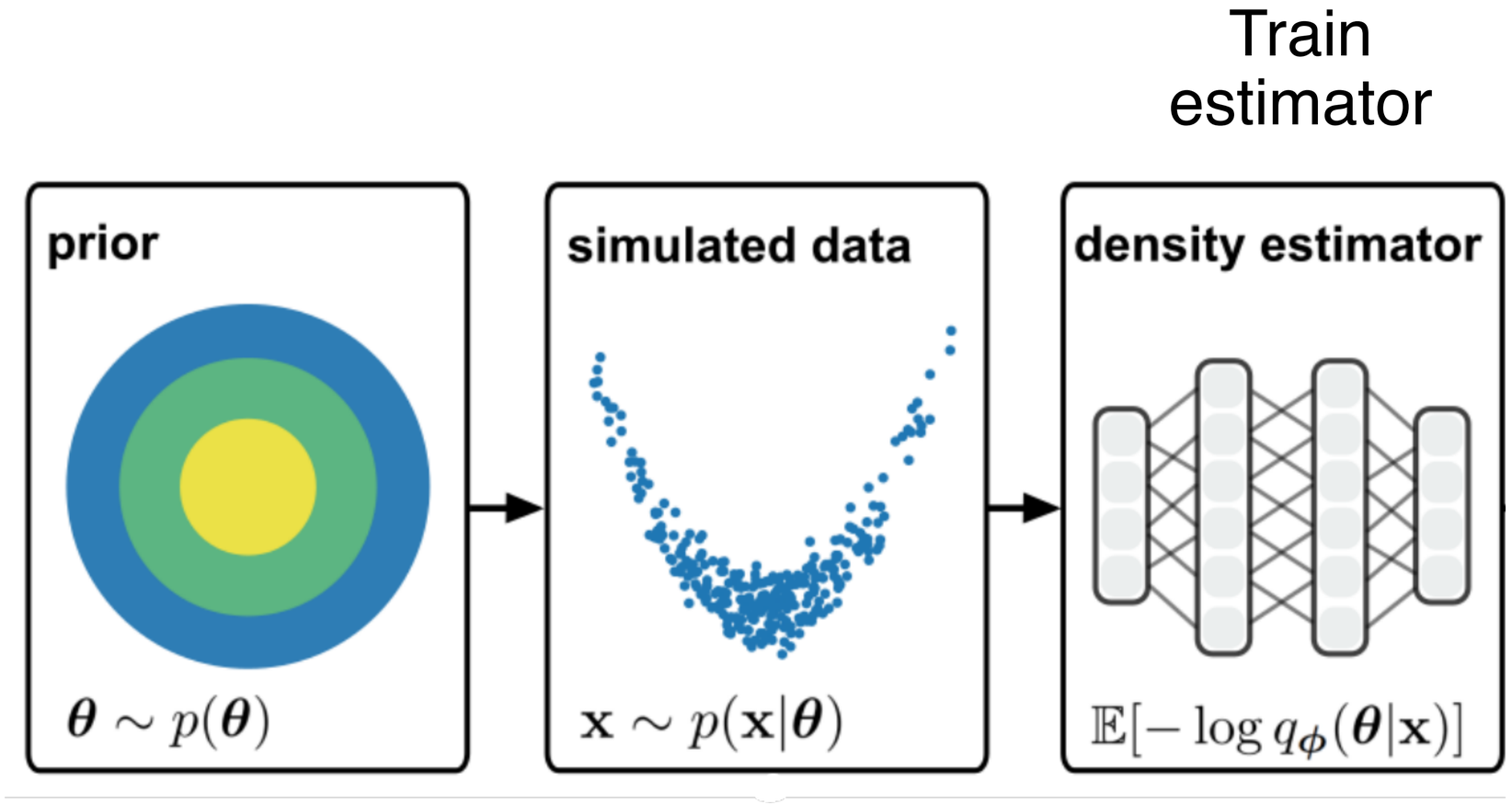
Sample from the prior



NPE: step 2

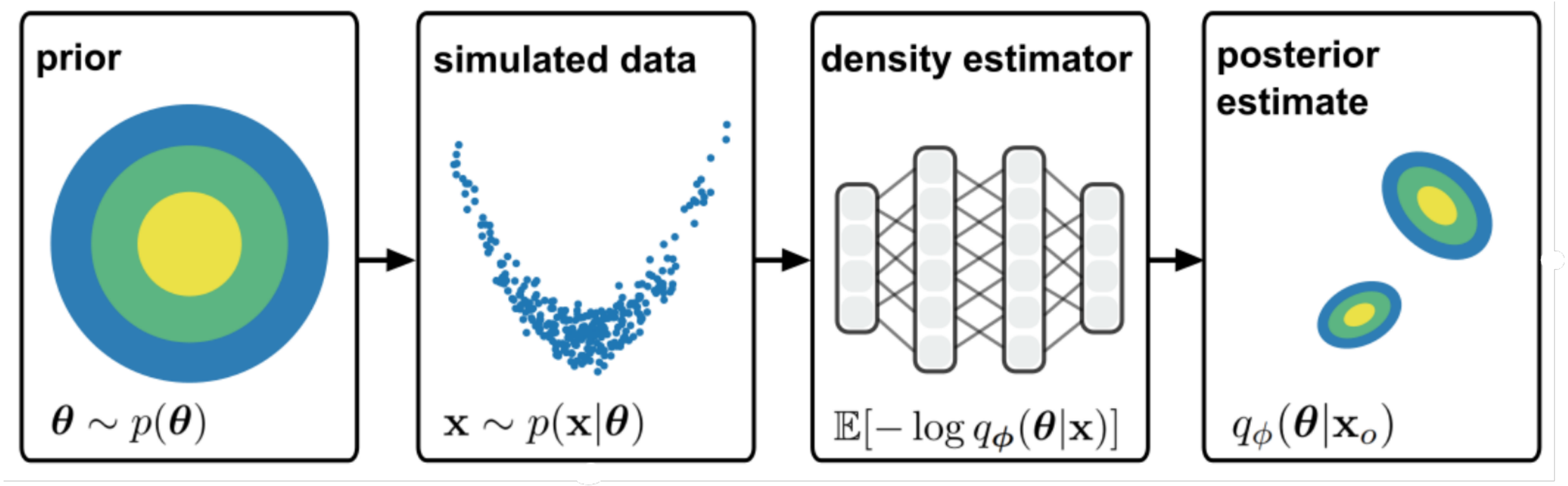

generate
simulations
prior
$\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$
simulated data
$\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$

NPE: step 3


Train
estimator
prior
$\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$
simulated data
$\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$
density estimator
$\mathbb{E}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}|\mathbf{x})]$

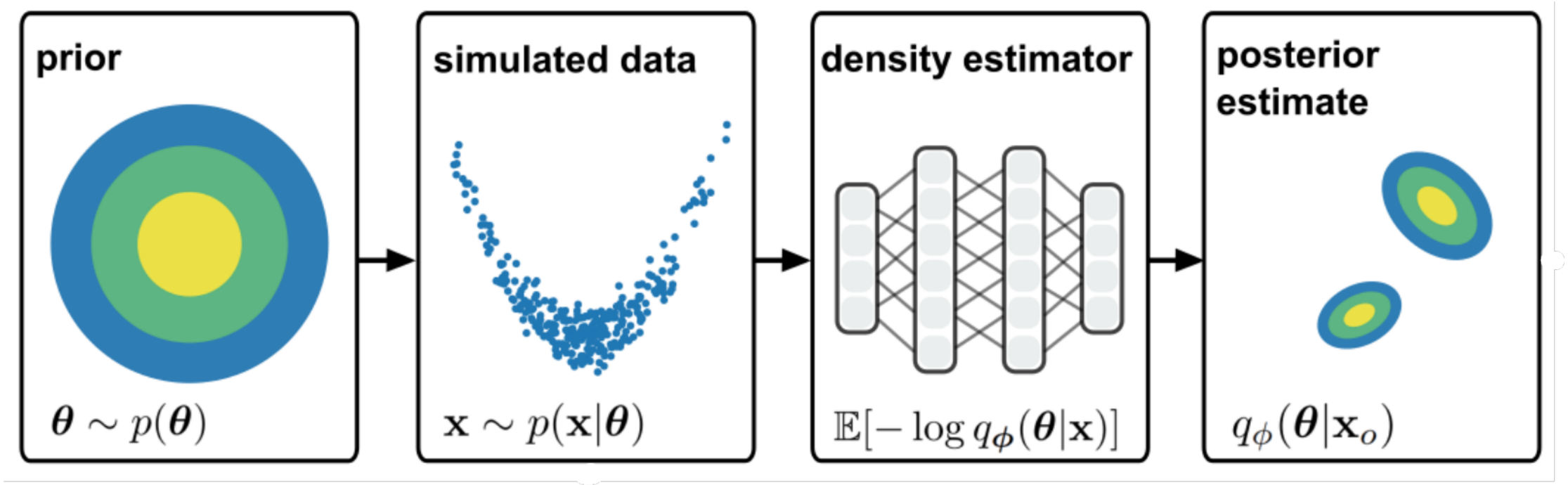
NPE: step 4

Evaluate at x_o



NPE: step 4

Evaluate at x_o



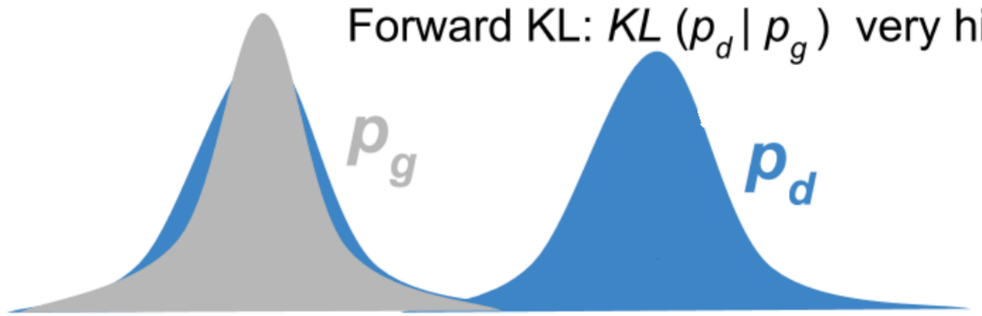
But why does it work? When does it work?

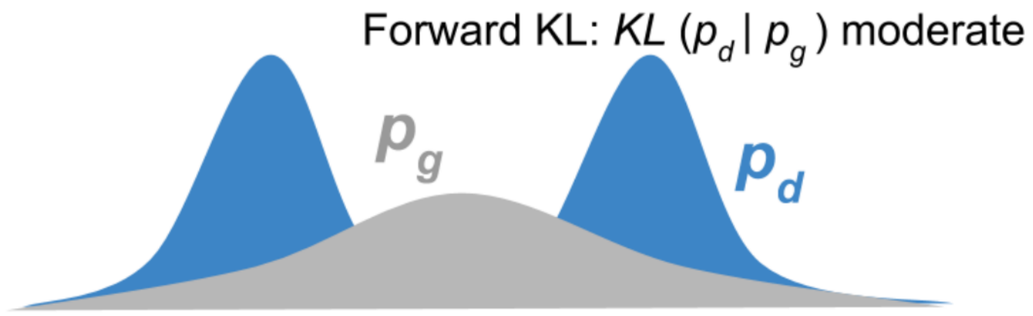
NPE: training objective

We want our density estimator q_ϕ for any given $x_o \sim p(x)$ that

$$q_\phi = \text{argmin} \underbrace{D_{KL}(p(\theta \mid x_o), q_\phi(\theta \mid x_o))}$$

Difference to the true posterior $p(\theta \mid x_o)$, as measured by the **Kullback-Leibler divergence**





NPE: training objective

We want our density estimator q_ϕ for any given $x_o \sim p(x)$ that

$$q_\phi = \text{argmin} \underbrace{D_{KL}(p(\theta \mid x_o), q_\phi(\theta \mid x_o))}$$

Difference to the true posterior $p(\theta \mid x_o)$, as measured by the **Kullback-Leibler divergence**

Hence we would like to minimize the loss

$$\mathscr{L} = \mathbb{E}_{p(x)}[D_{KL}(p(\theta \mid x), q_\phi(\theta \mid x))]$$

But how can we compute that?

NPE: training objective

We can rewrite this objective as following

$$\mathscr{L}(\phi) = \mathbb{E}_{p(x)}[D_{KL}\left(p(\theta\,|\,x), q_\phi(\theta\,|\,x)\right)]$$

$$= \mathbb{E}_{p(x)}\left[\mathbb{E}_{p(\theta|x)}\left[\log \frac{p(\theta\,|\,x)}{q_\phi(\theta\,|\,x)}\right]\right]$$

$$= \mathbb{E}_{p(x,\theta)}\left[-\log q_\phi(\theta\,|\,x)\right] + const\,.$$

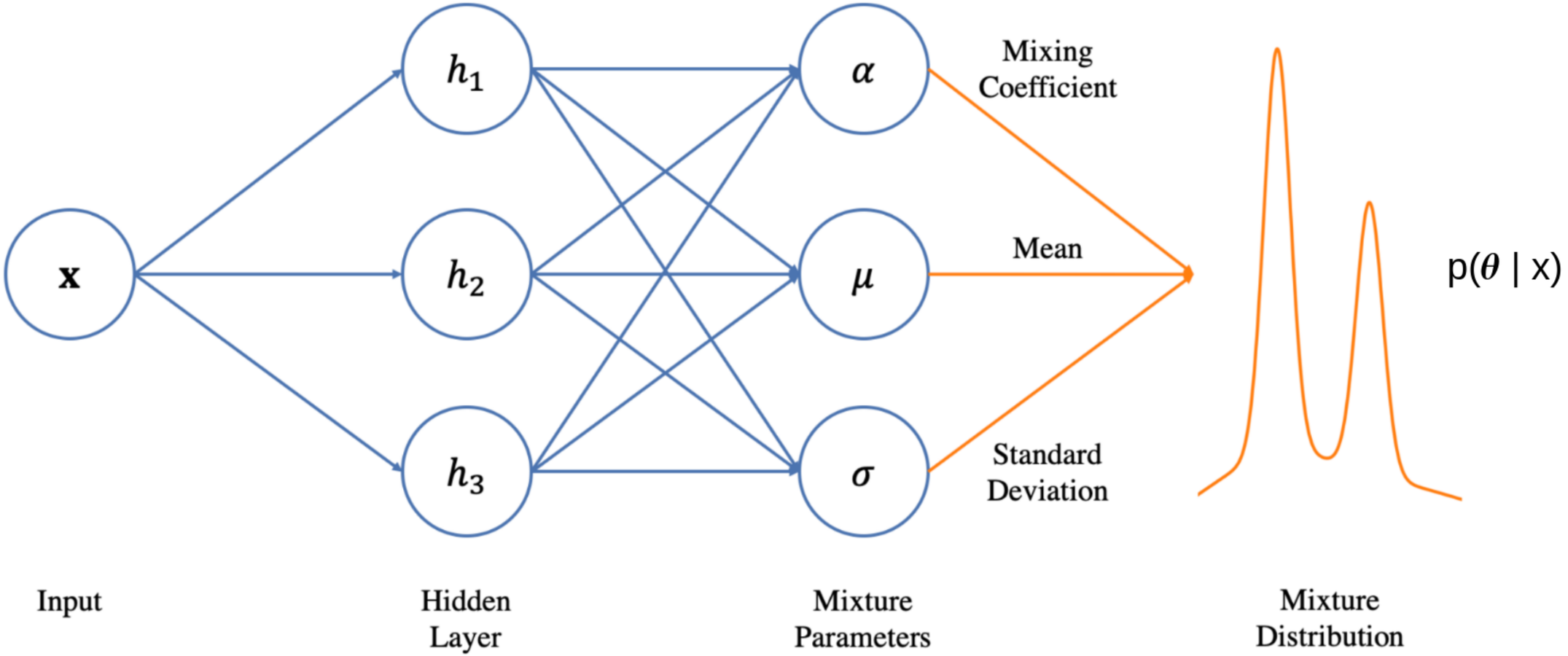
$$\approx \frac{1}{N}\sum_{i=1}^{N} -\log q_\phi(\theta_i\,|\,x_i) + const\,.$$

independent of ϕ !

We can compute this by sampling from the prior and simulating!

$p(x, \theta) = p(\theta) \cdot p(x\,|\,\theta)$

Gaussian MDN for NPE



What do we need to consider?

The **number of simulations**.

- The more the better! But simulations might be slow ☹
- High dimensional parameter spaces may need a LOT of simulations.

The choice of **conditional density estimator** q_ϕ.

- Expressivity: how flexible does the estimator need to be?
- We have some options, but this can have a big impact! More on that soon.

The **optimization** problem.

- Hard to tackle.
- ML best-practices required…

What do we get?

- The posterior is **fully amortised**:

For any observation x_o, we get the posterior $q_\phi(\theta \mid x_o)$ by *one* forward pass.

- The **prior** can be arbitrary:

We only need samples.

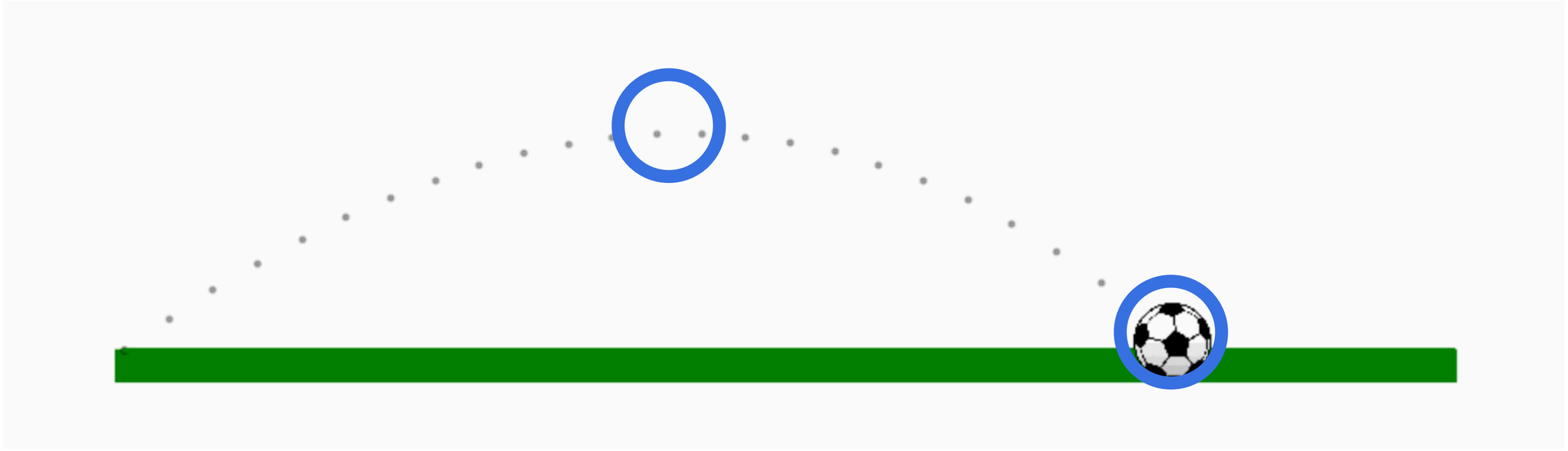
- The **simulator** can be arbitrary (“black box”):

We only need samples, given any possible parameter from the prior.

Dealing with high dimensions of simulator output

Often we are not interested in the full simulator output, but only some features describing the simulation ("*summary statistics*").

What are potential summary features here?



Dealing with high dimensions of simulator output

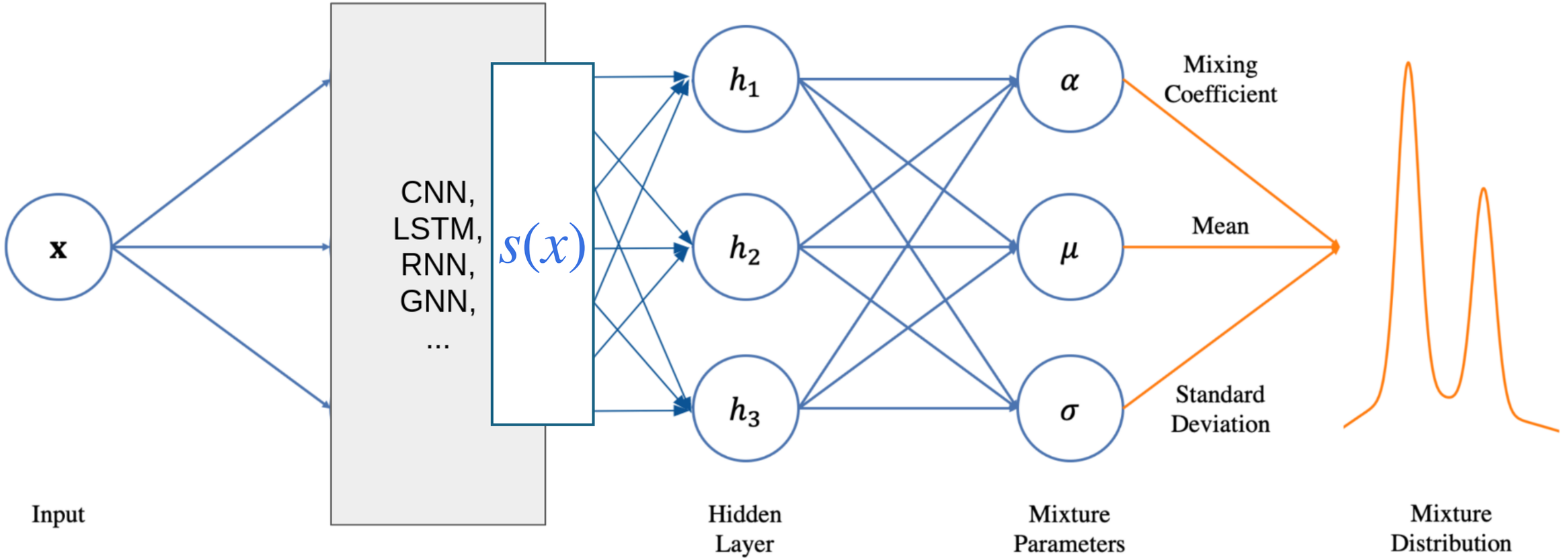
- Instead of conditioning on the whole simulation x, we can condition on the summary statistics $s(x)$ instead:

$$p(\theta \mid s(x)) \approx q_\phi(\theta \mid s(x))$$

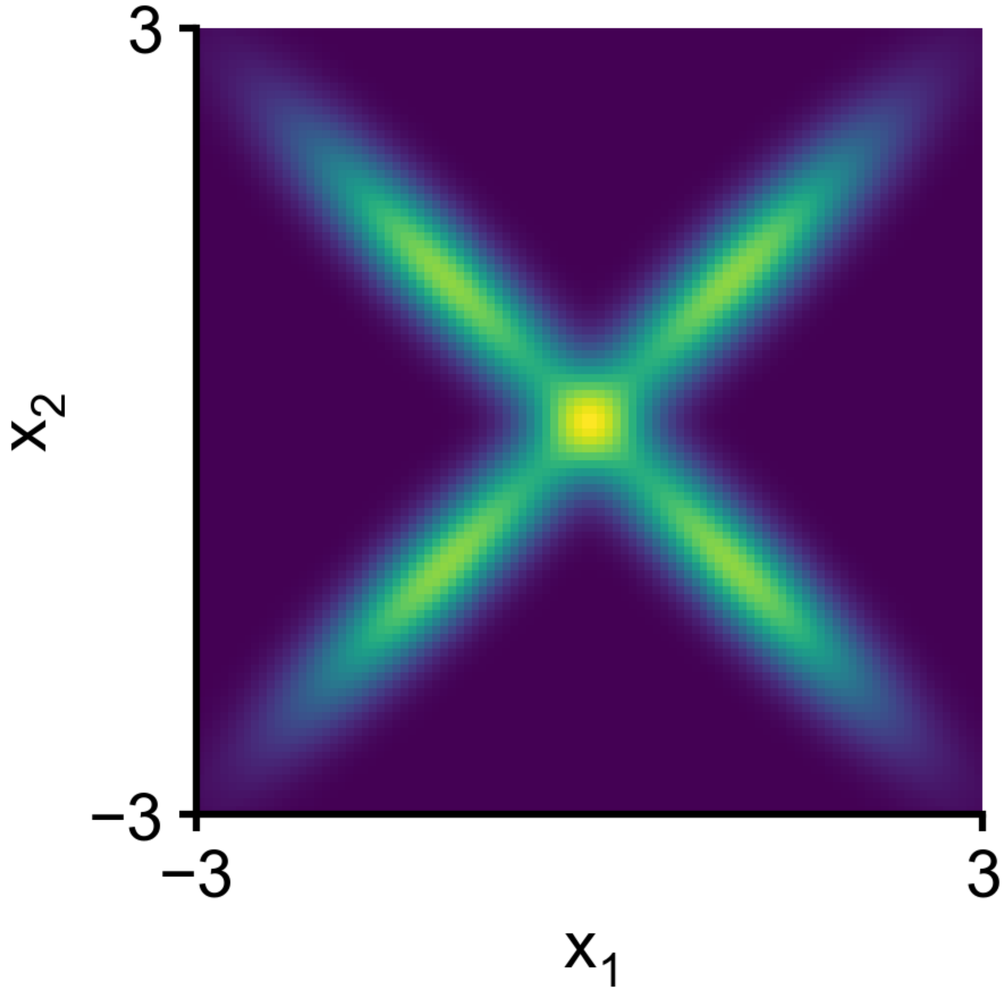
Everything else stays the same.

- We only infer the parameters for which the model reproduces the summary statistics $s(x)$.

Automated feature extraction



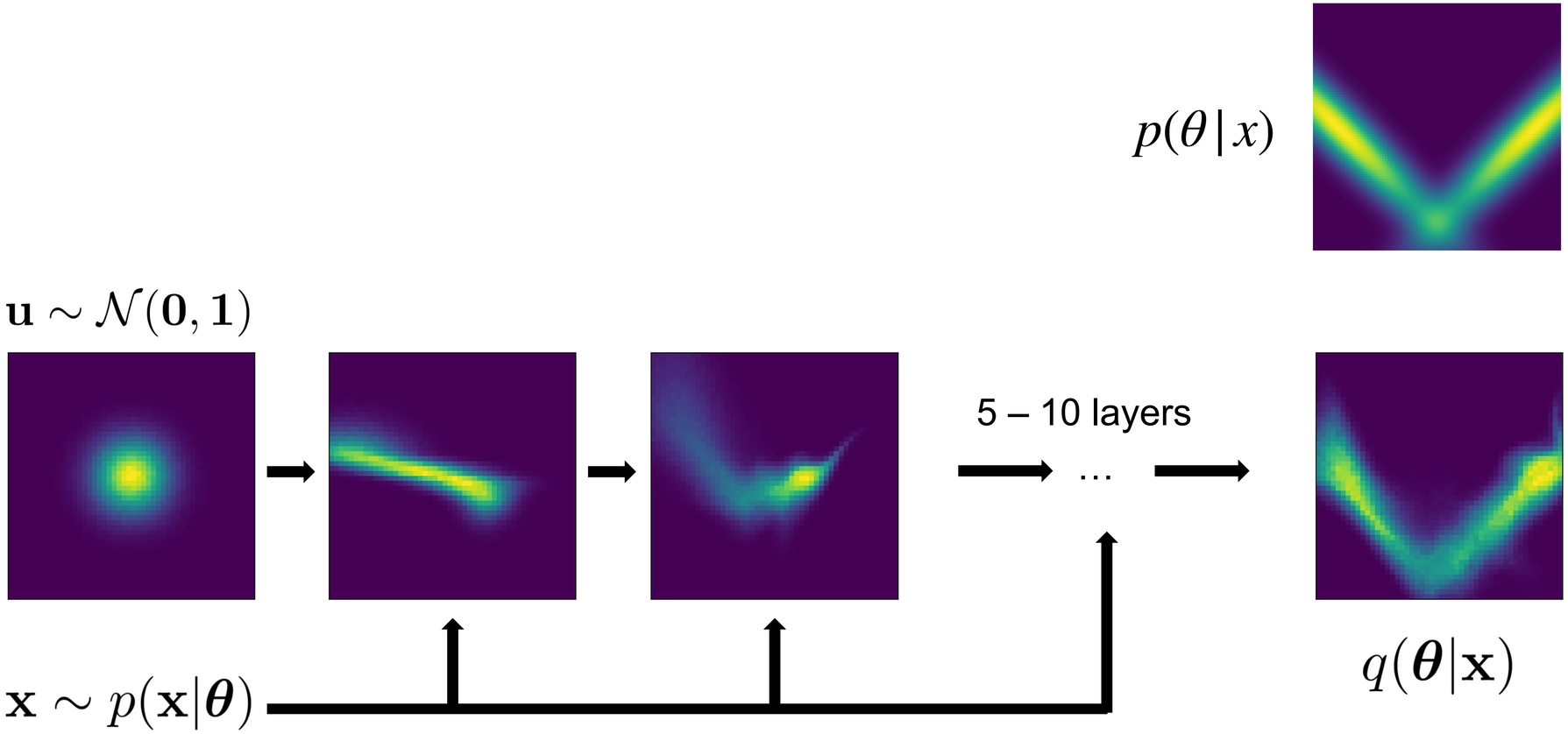
2.3 Normalising flows



Motivation

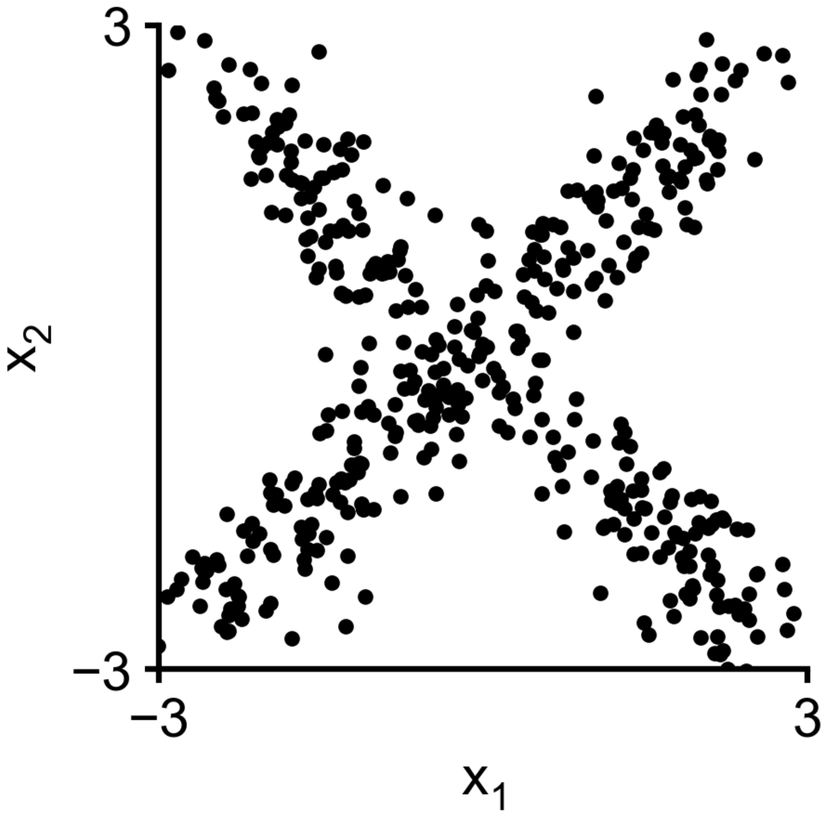
- Many tasks in Probabilistic Machine Learning require flexible models for (conditional) distributions, e.g., for SBI.
- Gaussian distributions are very convenient, but not flexible
- So: to make a flexible model, take a Gaussian, push it through a neural network.
- Gaussians + Change of Variables + Some Tricks = Normalising Flows
- Normalising Flows: Very useful for SBI (but also beyond)

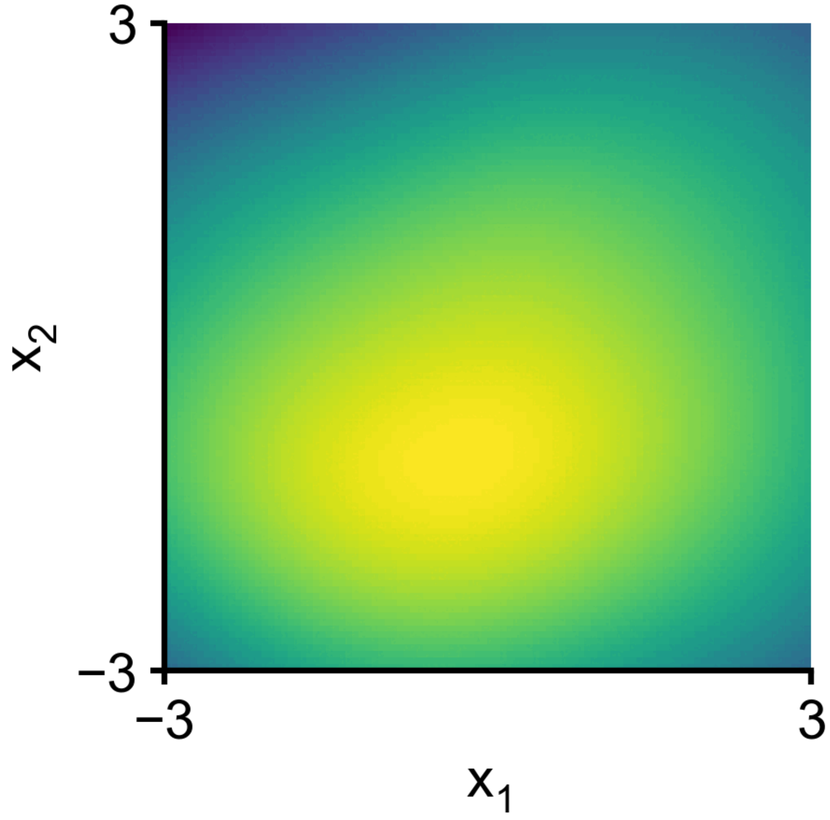
Normalizing Flows are flexible conditional density estimators



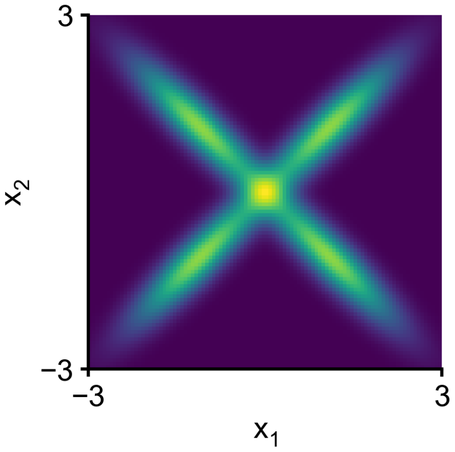
Papamakarios et al 2021 JMLR

Let us see this in action …





Ground truth:



Lecture 2: Neural Posterior Estimation (NPE)

- In NPE, we approximate an unknown posterior distribution $p(\theta \mid x)$ by minimising the KL-divergence to our model q_ϕ
- The four main steps of NPE:
  1. Sample from the prior: $\theta_i \sim p(\theta)$
  2. Run simulations: $x_i \sim p(x \mid \theta_i)$
  3. Train a neural density estimator $q_\phi(\theta \mid x)$ by minimising $\mathcal{L}(\phi) = \mathbb{E}[-\log q_\phi(\theta \mid x)]$
  4. Evaluate the estimator at x_o to get the approximate posterior $q_\phi(\theta \mid x_o) \approx p(\theta \mid x_o)$
- NPE is amortised: after training $q_\phi(\theta \mid x)$, we can evaluate it for any observation x_o
- Embedding networks are a way to automatically extract summary statistics
- NPE can be run with normalizing flows, which are highly flexible parametric densities

Further reading on NPE and Normalising flows

- Papamakarios, G., & Murray, I. (2016). Fast ε-free inference of simulation models with bayesian conditional density estimation. Advances in neural information processing systems, 29.

- Papamakarios, G. et al. (2021). Normalizing flows for probabilistic modeling and inference. Journal of Machine Learning Research, 22(57), 1-64.

- Bishop, C. M. (1994). Mixture density networks.