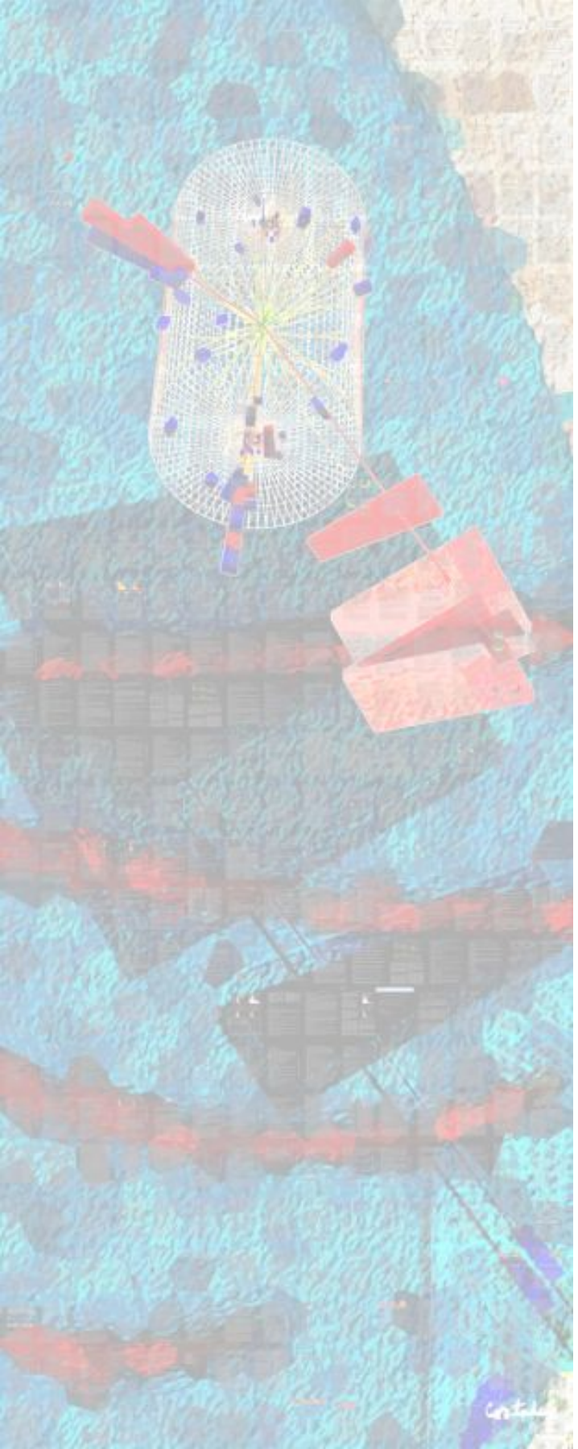


Inference with classifiers

AIMS 2026

Jonathon Langford

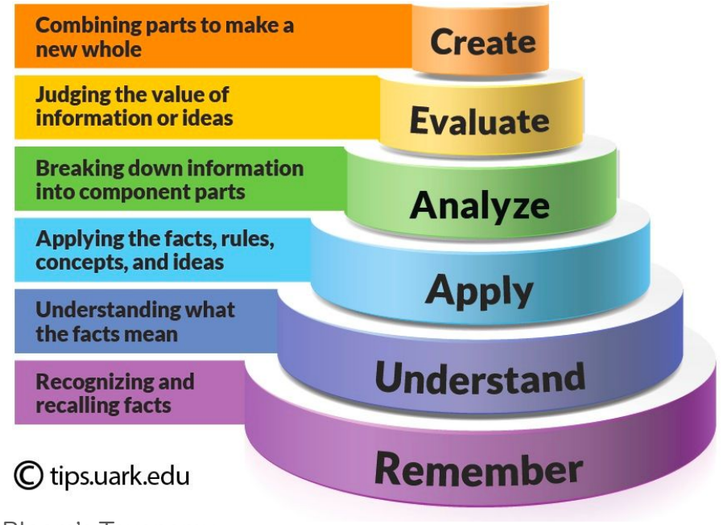
16th Feb 2026

Schmidt Sciences

Lecture contents & ILOs

- *Introduction to: likelihoods (intractable), simulation-based inference (SBI) and the frequentist paradigm*
- *Learning the log-likelihood-ratio with a simple ML classifier*
- *Hypothesis testing*
- *Parameter estimation*

The accompanying notebooks to this lecture can be found here: [ADD LINKS]



Bloom's Taxonomy

Intended Learning Outcomes

- **Understand** the need for simulation-based inference in the modern era of science
- **Understand** how to use a simple ML classifier to learn the likelihood-ratio and **apply** this knowledge to perform a hypothesis test on a research problem with an unknown likelihood
- Extend the approach to learning the conditional likelihood-ratio via a parametric classifier and **apply** this to a parameter estimation problem. **Evaluate** the performance by comparing to the analytic solution

Likelihoods are key

- Likelihoods (“probability density”) hold the key to inference
  - You perform an experiment N_{obs} times and observe the dataset $\mathcal{D} = \{x_i\}_{i=1}^{N_{\text{obs}}}$, where $x \in \mathbb{R}^d$
  - You have a **model**, defined by parameters θ, that describes the data: $p(x|\theta)$

$$p(\mathcal{D}|\theta) = \prod_{x_i \in \mathcal{D}}^{N_{\text{obs}}} p(x_i|\theta)$$

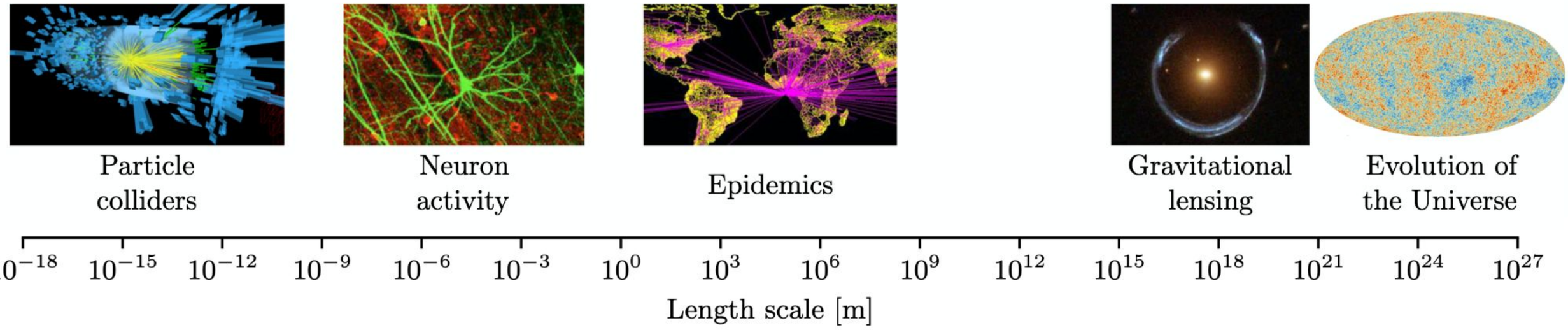
- **<u>Simple example:</u>** x is tossing a “HEAD” on a coin. Define θ as probability of throwing a HEAD:

$$p(\text{HEADS}|\theta) = \theta$$
$$p(\text{TAILS}|\theta) = 1 - \theta$$

- We throw a coin 5 times and get the sequence $\mathcal{D} = \{x_i\}_{i=1}^{N_{\text{obs}}}$ = (H, T, T, H, T)
  - First assume coin is fair (**Null hypothesis**, $\mathcal{H}_0$) i.e. θ = 0.5 $\rightarrow$ $p(\mathcal{D}|\theta = 0.5) = 0.5 \cdot (1 - 0.5) \cdot (1 - 0.5) \cdot 0.5 \cdot (1 - 0.5) = 0.5^5 \approx 0.031$
  - **Alternative hypothesis** ($\mathcal{H}_1$) for a biased coin e.g. θ = 0.6 $\rightarrow$ $p(\mathcal{D}|\theta = 0.6) = 0.6 \cdot (1 - 0.6) \cdot (1 - 0.6) \cdot 0.6 \cdot (1 - 0.6) \approx 0.023$
  - Our data prefers $\mathcal{H}_0$ to $\mathcal{H}_1$ i.e. **we have used the likelihood to infer something about θ**
- Comparison (ratio) of likelihoods is a key concept in hypothesis testing → We will formalize this in a few slides

Scientific era of simulations

Figure taken from [arXiv:1911.01429]

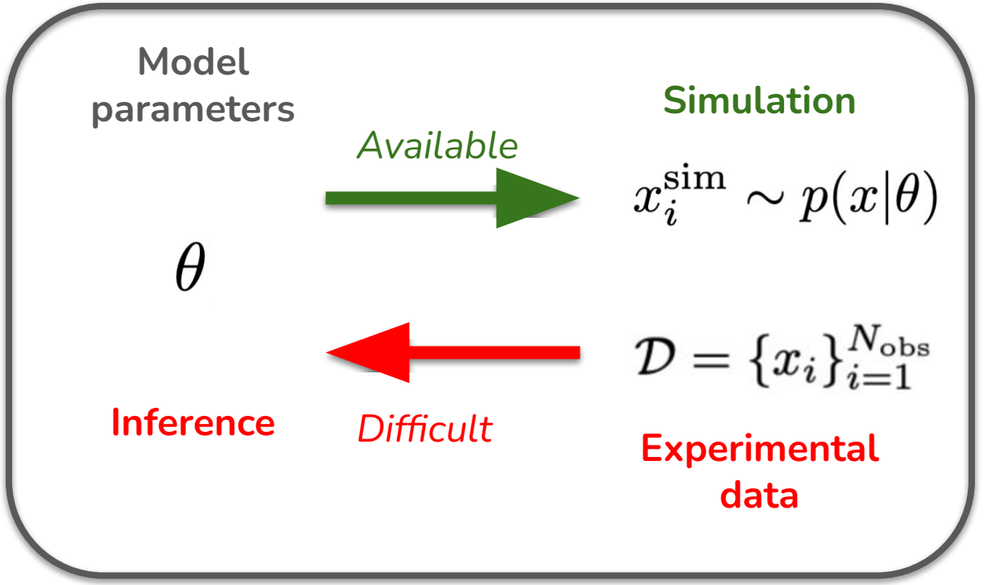


- Incredible complexity across all experimental length scales
  - Impossible to calculate likelihoods (intractable)

$$p(\mathcal{D}|\theta) = \prod_{x_i \in \mathcal{D}}^{N_{obs}} \int dz\, p(x_i|\theta, z)$$

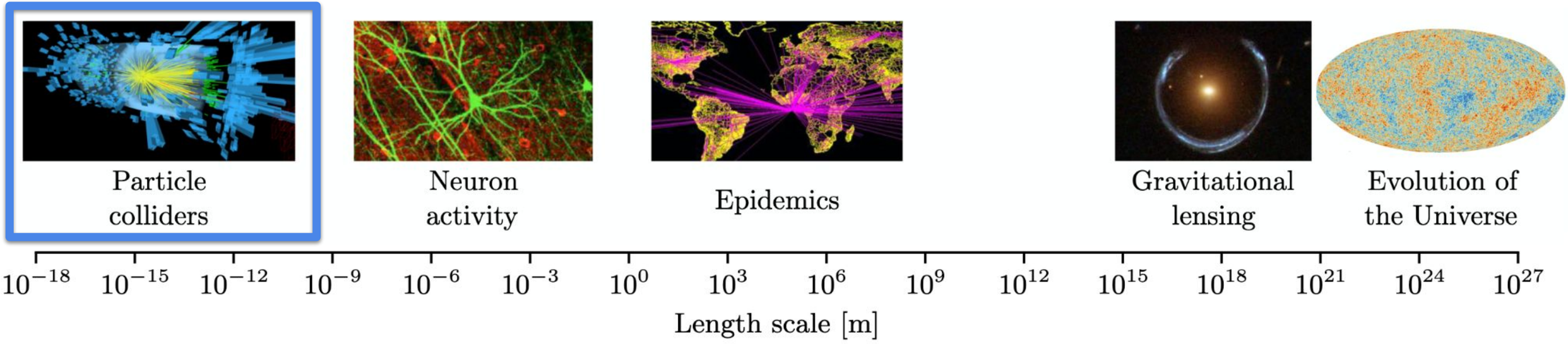
Latent features

- But we have access to **high-fidelity simulations**



Scientific era of simulations

Figure taken from [arXiv:1911.01429]

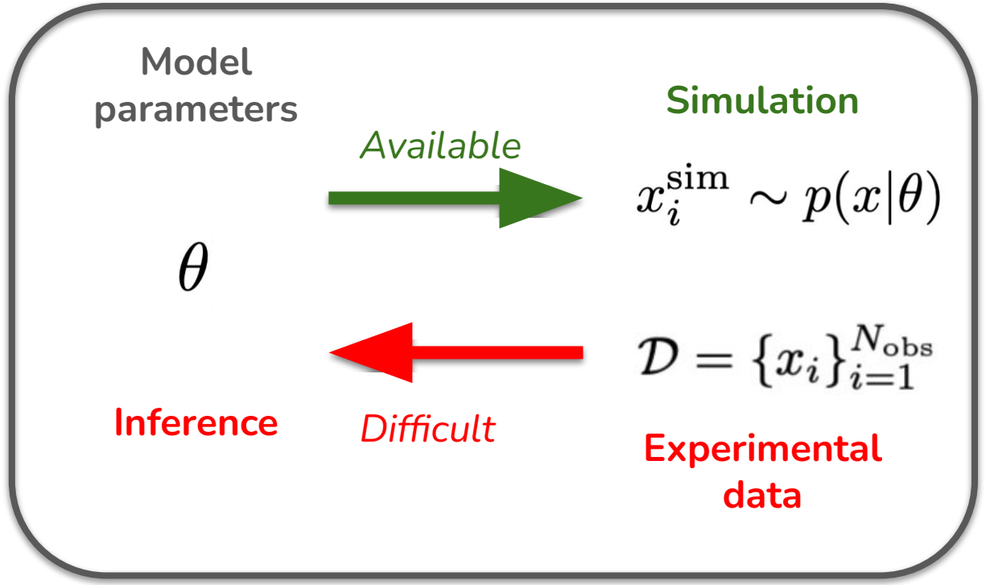


- Incredible complexity across all experimental length scales
  - Impossible to calculate likelihoods (intractable)

$$p(\mathcal{D}|\theta) = \prod_{x_i \in \mathcal{D}}^{N_{obs}} \int dz\, p(x_i|\theta, z)$$

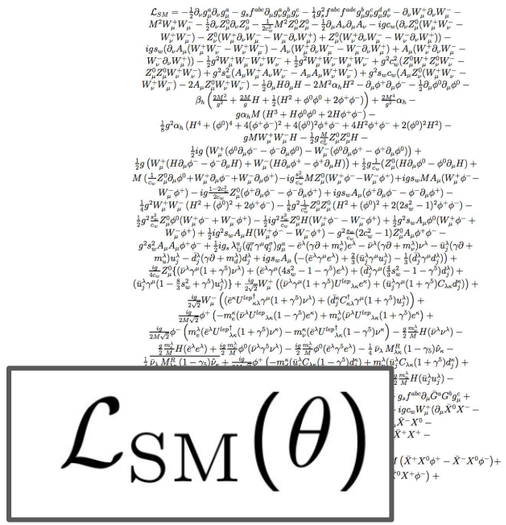
Latent features

- But we have access to **high-fidelity simulations**



Scientific era of simulations

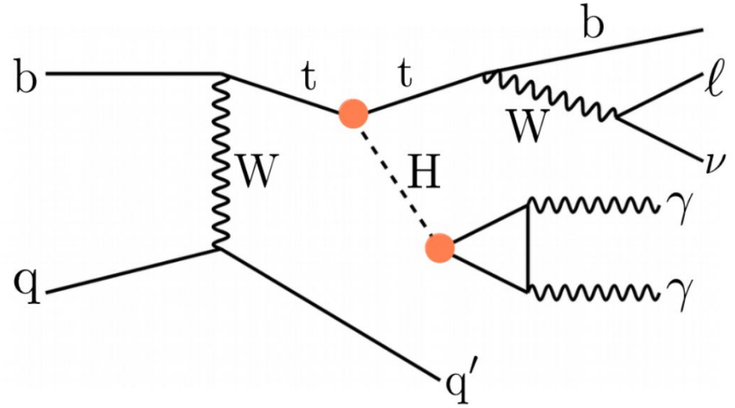
$$p(x\,|\,\theta) = \int \mathrm{d}z_d \int \mathrm{d}z_s \int \mathrm{d}z_p p(x|z_d)p(z_d|z_s)p(z_s|z_p)p(z_p|\theta)$$



$$\mathcal{L}_{\mathrm{SM}}(\theta)$$

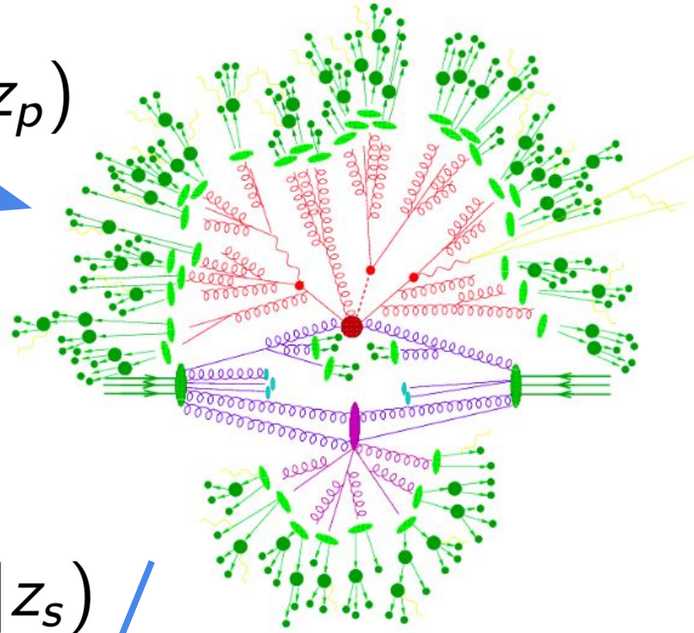
Model Lagrangian
(underlying theory)

$p(z_p|\theta)$



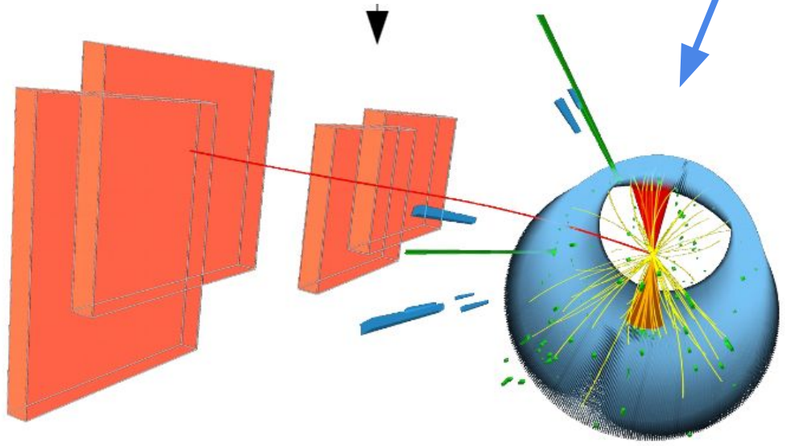
Hard scatter

$p(z_s|z_p)$



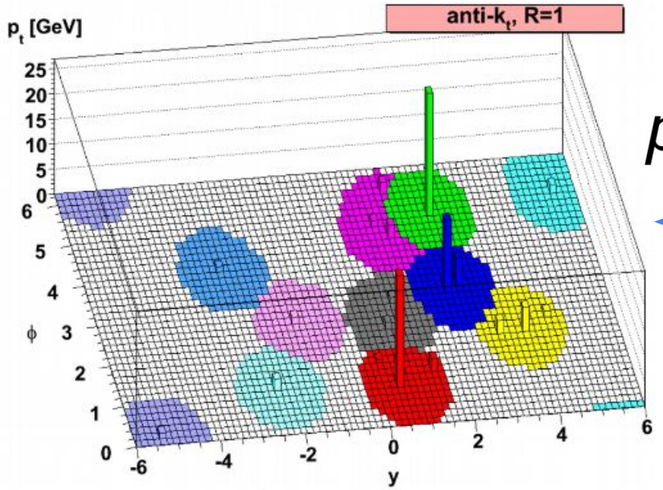
Parton shower and hadronisation

$p(z_d|z_s)$



Interaction of particles with detector material

$p(x|z_d)$

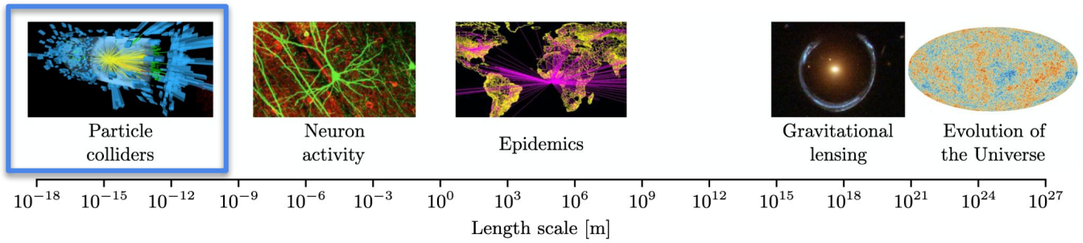


Reconstruction from electronic signals

X

Simulated Data

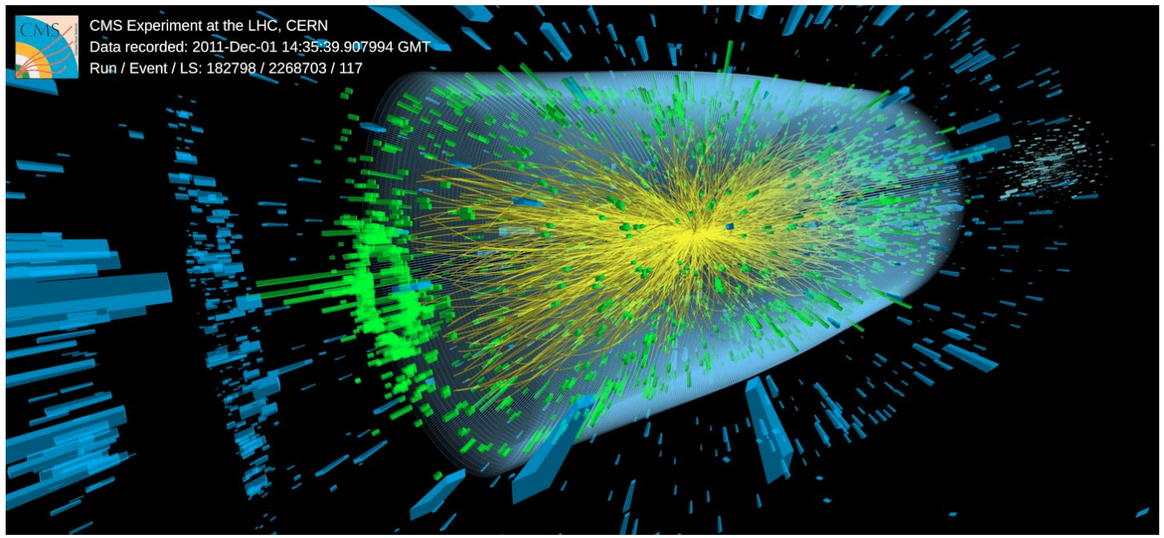
Simulation-based inference (SBI)



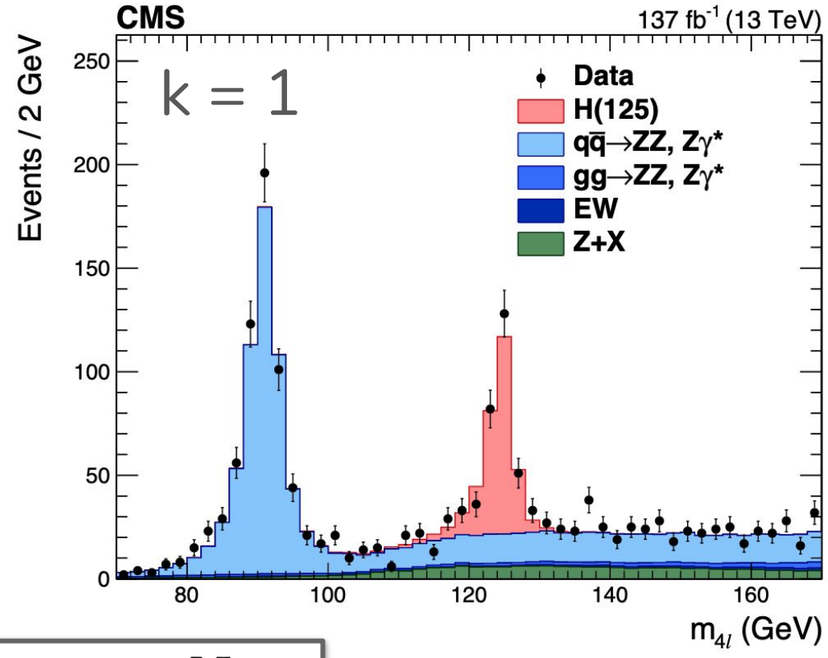
- Traditional approaches to SBI
  - Approximate Bayesian Computation (ABC)
  - **(Probability) density estimation via histogramming/kernels**

$$f(x) \in \mathbb{R}^k, \quad k \ll d$$

- Data is often high-dimensional: $x \in \mathbb{R}^d$ → construct low-dimensional summary statistic for inference



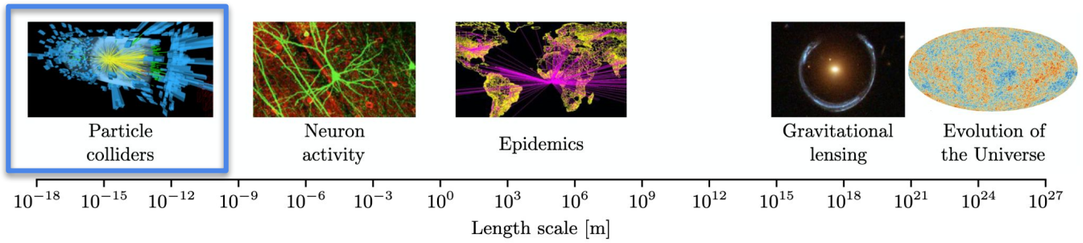
e.g. energy & momentum of O(100) particles produced in each LHC collision



$$\hat{p}(f(x)|\theta) = \frac{N_b}{N_{\text{tot}}\Delta_b}$$

Bin the simulation in summary statistic and estimate the density
→ Compare to observed data

Simulation-based inference (SBI)



- Traditional approaches to SBI
  - Approximate Bayesian Computation (ABC)
  - **(Probability) density estimation via histogramming/kernels**

$$f(x) \in \mathbb{R}^k, \quad k \ll d$$

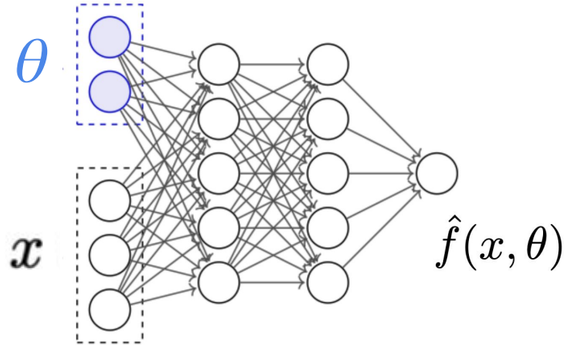
- Data is often high-dimensional: $x \in \mathbb{R}^d$ → construct low-dimensional summary statistic for inference
  - Curse of dimensionality prevents accurate estimator for dimension $k \gtrsim 3$
  - Inevitable loss of information by dimensionality reduction (and binning)
  - Lose statistical power for inferring θ

$$\hat{p}(f(x)|\theta) = \frac{N_b}{N_{\text{tot}}\Delta_b}$$

Q: Can we directly estimate the likelihood as a function of a high-dimensional space, x ?

$$\hat{p}(x|\theta)$$

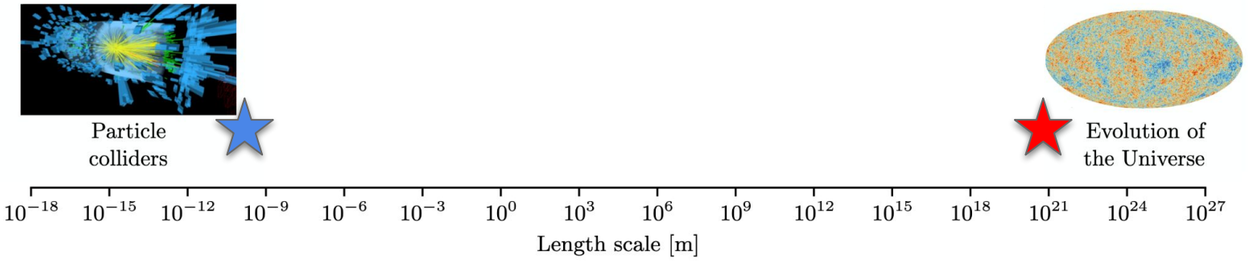
A: use Machine-Learning (ML) → retain maximum sensitivity to θ



- Deep learning models can be effective surrogates for the likelihood (and likelihood-ratios)
- Function of the (full) multivariate input space x
- No need for low-dimensional summaries that lose statistical power

Inference paradigms

- How to use likelihood $p(\mathcal{D}|\theta)$ to infer parameters θ ?



Frequentist inference

$$t_\theta = -2\ln\left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)}\right)$$

Bayesian inference

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

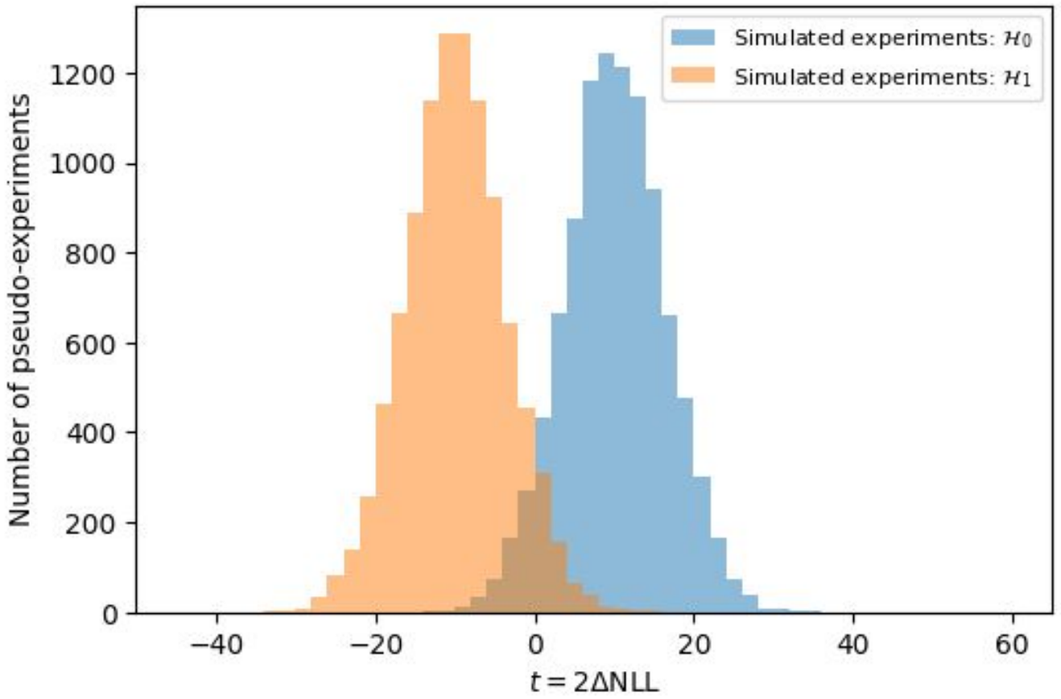
- Interpretation over hypothetical repetitions of the experiment
- **Define a test-statistic** t_θ = a number (score) to summarise **how compatible the data is with a hypothesis**
  - Consider a simple hypothesis test: $t_\theta = -2\ln\left(\frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)}\right)$, where $\mathcal{H}_0 : \theta = \theta_0$ $\mathcal{H}_1 : \theta = \theta_1$
  - Neyman Pearson lemma: (log)-likelihood ratio is the most powerful test-statistic for simple hypothesis testing
  - Calculate test-statistic for hypothetical repetitions of experiment under $\mathcal{H}_0$ and $\mathcal{H}_1$ (using simulation)
  - Compare to observed value t_{obs} to reject/accept hypotheses

Frequentist inference

- How to use likelihood $p(\mathcal{D}|\theta)$ to infer parameters θ ?

Frequentist inference

$$t_\theta = -2\ln\left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)}\right)$$



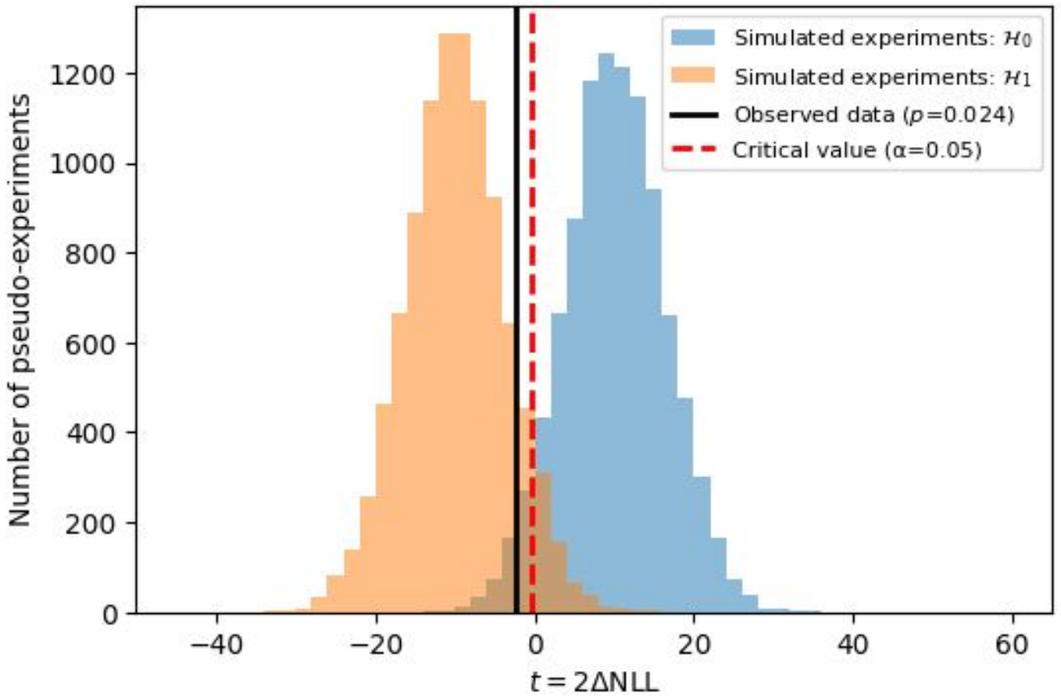
- Interpretation over hypothetical repetitions of the experiment
- **Define a test-statistic** t_θ = a number (score) to summarise **how compatible the data is with a hypothesis**
  - Consider a simple hypothesis test: $t_\theta = -2\ln\left(\frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)}\right)$, where $\mathcal{H}_0 : \theta = \theta_0$ $\mathcal{H}_1 : \theta = \theta_1$
  - Neyman Pearson lemma: (log)-likelihood ratio is the most powerful test-statistic for simple hypothesis testing
  - Calculate test-statistic for hypothetical repetitions of experiment under $\mathcal{H}_0$ and $\mathcal{H}_1$ (using simulation)
  - Compare to observed value t_{obs} to reject/accept hypothesis

Inference paradigms

- How to use likelihood $p(\mathcal{D}|\theta)$ to infer parameters θ ?

Frequentist inference

$$t_\theta = -2\ln\left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)}\right)$$

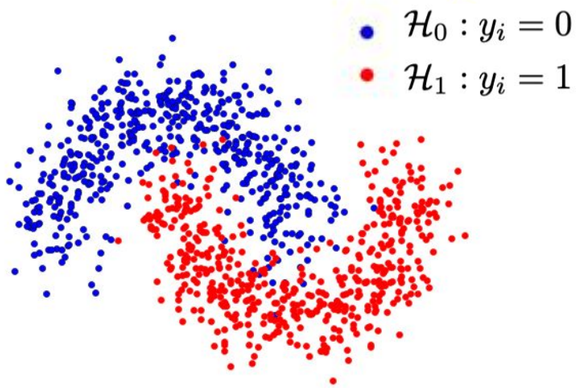


- Interpretation over hypothetical repetitions of the experiment

- **Define a test-statistic** t_θ = a number (score) to summarise **how compatible the data is with a hypothesis**
  - Consider a simple hypothesis test: $t_\theta = -2\ln\left(\frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)}\right)$, where $\mathcal{H}_0 : \theta = \theta_0$ $\mathcal{H}_1 : \theta = \theta_1$
  - Neyman Pearson lemma: (log)-likelihood ratio is the most powerful test-statistic for simple hypothesis testing
  - Calculate test-statistic for hypothetical repetitions of experiment under $\mathcal{H}_0$ and $\mathcal{H}_1$ (using simulation)
  - Compare to observed value t_{obs} to reject/accept hypothesis

Learning the log-likelihood ratio

- **Can we use Machine Learning to estimate the (log)-likelihood ratio test-statistic for inference?**
  - Consider a simple binary classifier to distinguish samples drawn from $p(x|\mathcal{H}_0)$ vs samples drawn from $p(x|\mathcal{H}_1)$
  - **Crucial:** for SBI the samples (x_i) are produced with the simulator
  - Train classifier, f(x), by minimizing the binary cross-entropy (BCE) loss function:



$$\mathcal{L}[f] = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln f(x_i) + (1-y_i)\ln\left(1-f(x_i)\right)$$

  - Assuming "balanced classes" → optimal decision function (i.e. f(x) which minimizes the loss) is...

$$f(x_i) = \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0) + p(x_i|\mathcal{H}_1)}$$

[Maths in Back-Up]

  - In reality (finite training samples, finite architecture), our trained classifier will be an estimator of optimal decision function

$$\hat{f}(x_i) \approx f(x_i) = \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0) + p(x_i|\mathcal{H}_1)}$$

    - *N.B. classification task → mis-labelling of data, but for inference task → bias in our measurement! Requires careful validation*

Learning the log-likelihood ratio

- **Can we use Machine Learning to estimate the (log)-likelihood ratio test-statistic for inference?**
  - With a simple rearranging (known as the likelihood-ratio trick) we arrive at:

$$\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)}$$

  - We have learnt an approximation of the **probability density ratio** between two hypothesis for a single sample x_i
  - **Frequentist inference:** likelihood-ratio over the observed dataset [Neyman-Pearson Lemma]

$$\frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} = \prod_{x_i \in \mathcal{D}}^{N_{obs}} \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)} \approx \prod_{x_i \in \mathcal{D}}^{N_{obs}} \frac{\hat{f}(x_i)}{1-\hat{f}(x_i)}$$

  - Construct test-statistic:

$$t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \right)$$

- **Summary: we can use output of binary classifier trained with simulation to approximate log-likelihood ratio test-statistic**

Hypothesis testing example

- Notebooks: ADD LINKS
- Two-class hypothesis test where analytic likelihood is not known (intractable) but we have a faithful simulator
- **Example:** infer the spin configuration of particle A
  - We perform an experiment and the observed data contains N_{obs} = 10 samples (A decays)
  - Three kinematic observables x = (`x1`, `x2`, `x3`) are measured i.e. 3D data
  - **Task:** we need to decide if A is spin-0 ($\mathcal{H}_0$) or spin-1 ($\mathcal{H}_1$)
  - We can simulate A decays under each spin hypothesis i.e. simulate samples from $p(x|\mathcal{H}_0)$ or $p(x|\mathcal{H}_1)$

	x1	x2	x3
0	-1.137580	2.427345	-1.587226
1	0.193618	-2.112575	-1.114890
2	-0.088694	0.726530	-1.673467
3	-0.390868	-1.162352	-0.849197
4	-0.358286	-0.119108	-2.076547
5	-2.098496	0.677189	2.820093
6	0.851472	0.409719	0.306846
7	2.280547	2.090513	-1.099859
8	-0.482024	-1.209504	0.107297
9	0.330048	-0.113135	0.954608

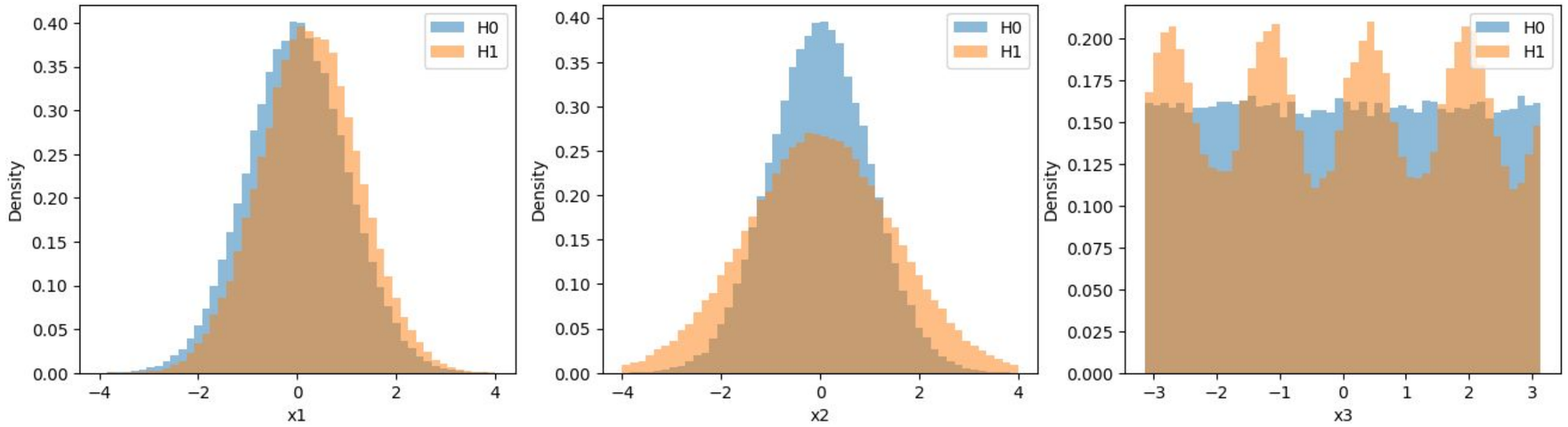
```
from sbi_utils import run_simulation
N_train = 100000
sim_H0 = run_simulation(N_train, hypothesis='H0')
sim_H1 = run_simulation(N_train, hypothesis='H1')
```

[15] ✓ 0.3s

Hypothesis testing example

- Notebooks: ADD LINKS
- Two-class hypothesis test where analytic likelihood is not known (intractable) but we have a faithful simulator
- **Example:** infer the spin configuration of particle A
  - We perform an experiment and the observed data contains N_{obs} = 10 samples (A decays)
  - Three kinematic observables x = (`x1`, `x2`, `x3`) are measured i.e. 3D data
  - **Task:** we need to decide if A is spin-0 ($\mathcal{H}_0$) or spin-1 ($\mathcal{H}_1$)
  - We can simulate A decays under each spin hypothesis i.e. simulate samples from $p(x|\mathcal{H}_0)$ or $p(x|\mathcal{H}_1)$

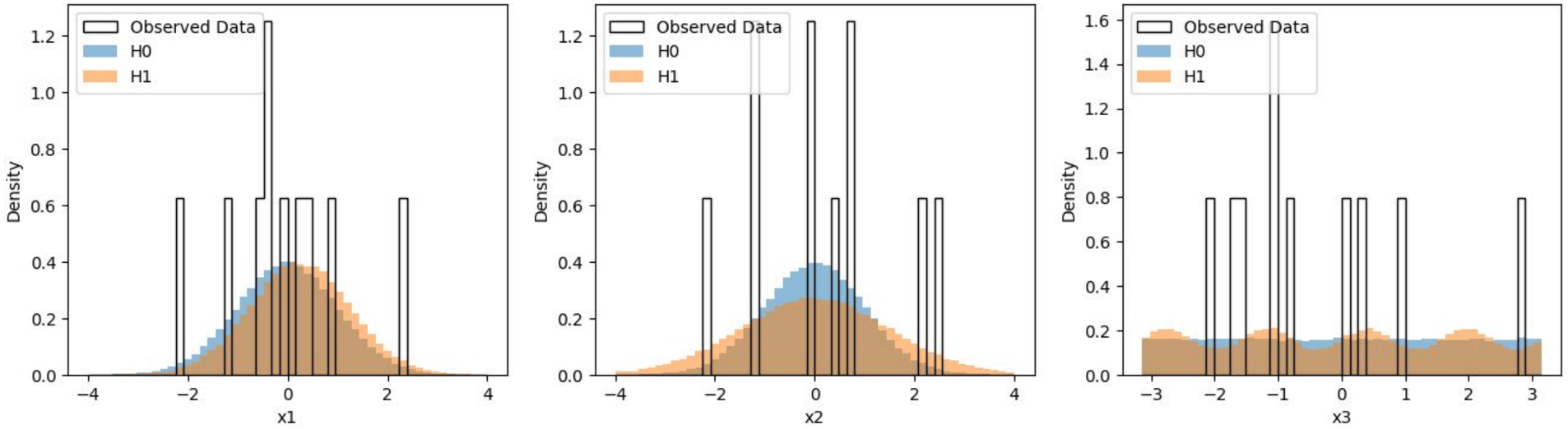
	x1	x2	x3
0	-1.137580	2.427345	-1.587226
1	0.193618	-2.112575	-1.114890
2	-0.088694	0.726530	-1.673467
3	-0.390868	-1.162352	-0.849197
4	-0.358286	-0.119108	-2.076547
5	-2.098496	0.677189	2.820093
6	0.851472	0.409719	0.306846
7	2.280547	2.090513	-1.099859
8	-0.482024	-1.209504	0.107297
9	0.330048	-0.113135	0.954608



Hypothesis testing example

- Notebooks: ADD LINKS
- Two-class hypothesis test where analytic likelihood is not known (intractable) but we have a faithful simulator
- **Example:** infer the spin configuration of particle A
  - We perform an experiment and the observed data contains N_{obs} = 10 samples (A decays)
  - Three kinematic observables x = (`x1`, `x2`, `x3`) are measured i.e. 3D data
  - **Task:** we need to decide if A is spin-0 ($\mathcal{H}_0$) or spin-1 ($\mathcal{H}_1$)
  - We can simulate A decays under each spin hypothesis i.e. simulate samples from $p(x|\mathcal{H}_0)$ or $p(x|\mathcal{H}_1)$

	x1	x2	x3
0	-1.137580	2.427345	-1.587226
1	0.193618	-2.112575	-1.114890
2	-0.088694	0.726530	-1.673467
3	-0.390868	-1.162352	-0.849197
4	-0.358286	-0.119108	-2.076547
5	-2.098496	0.677189	2.820093
6	0.851472	0.409719	0.306846
7	2.280547	2.090513	-1.099859
8	-0.482024	-1.209504	0.107297
9	0.330048	-0.113135	0.954608



Hypothesis testing example

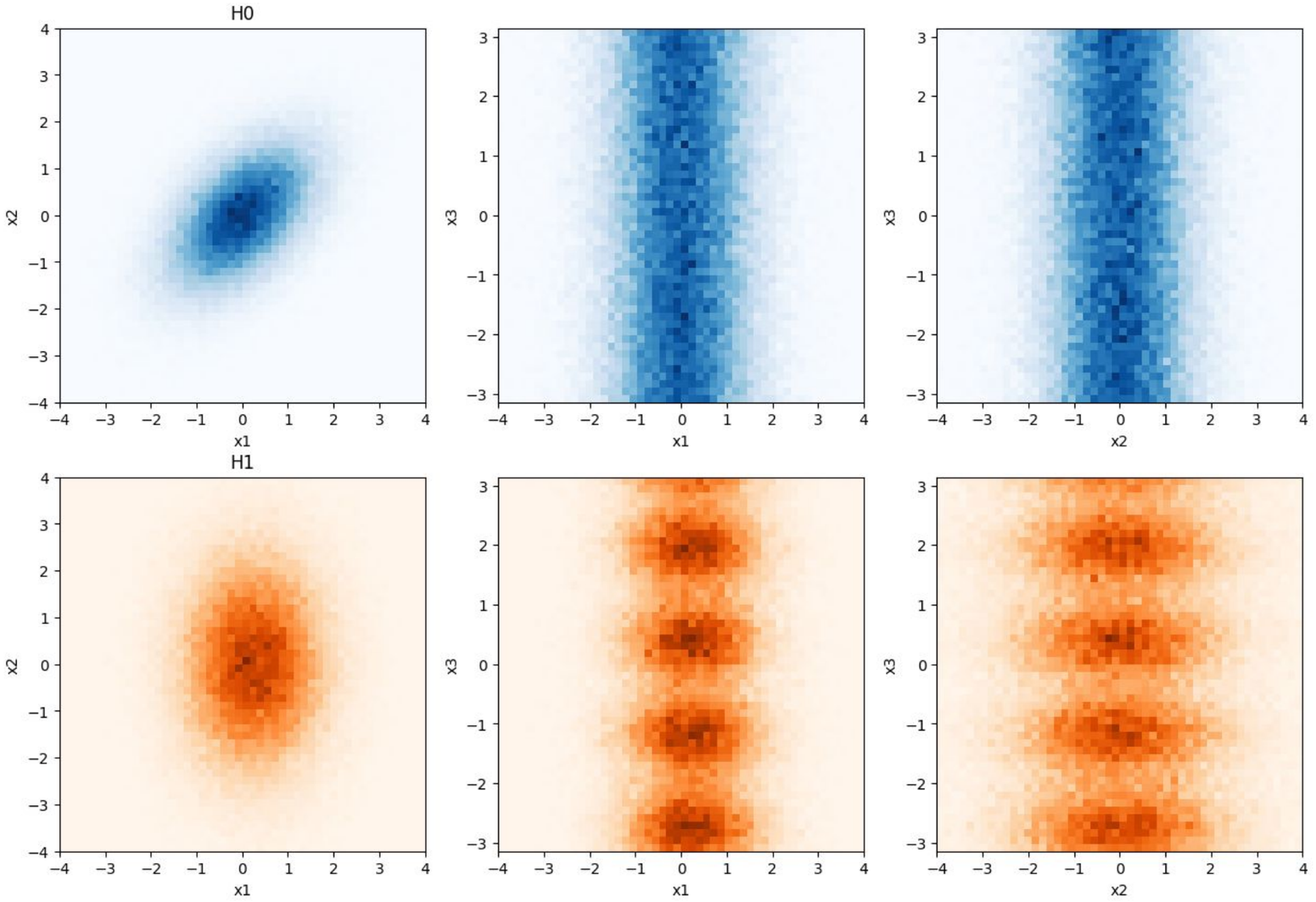
- This is a higher-dimensional problem (e.g. in 2D)

```
from sbi_utils import run_simulation
N_train = 100000
sim_H0 = run_simulation(N_train, hypothesis='H0')
sim_H1 = run_simulation(N_train, hypothesis='H1')
```

[15] ✓ 0.3s

$\mathcal{H}_0$

$\mathcal{H}_1$



Hypothesis testing example

- This is a higher-dimensional problem (e.g. in 2D)

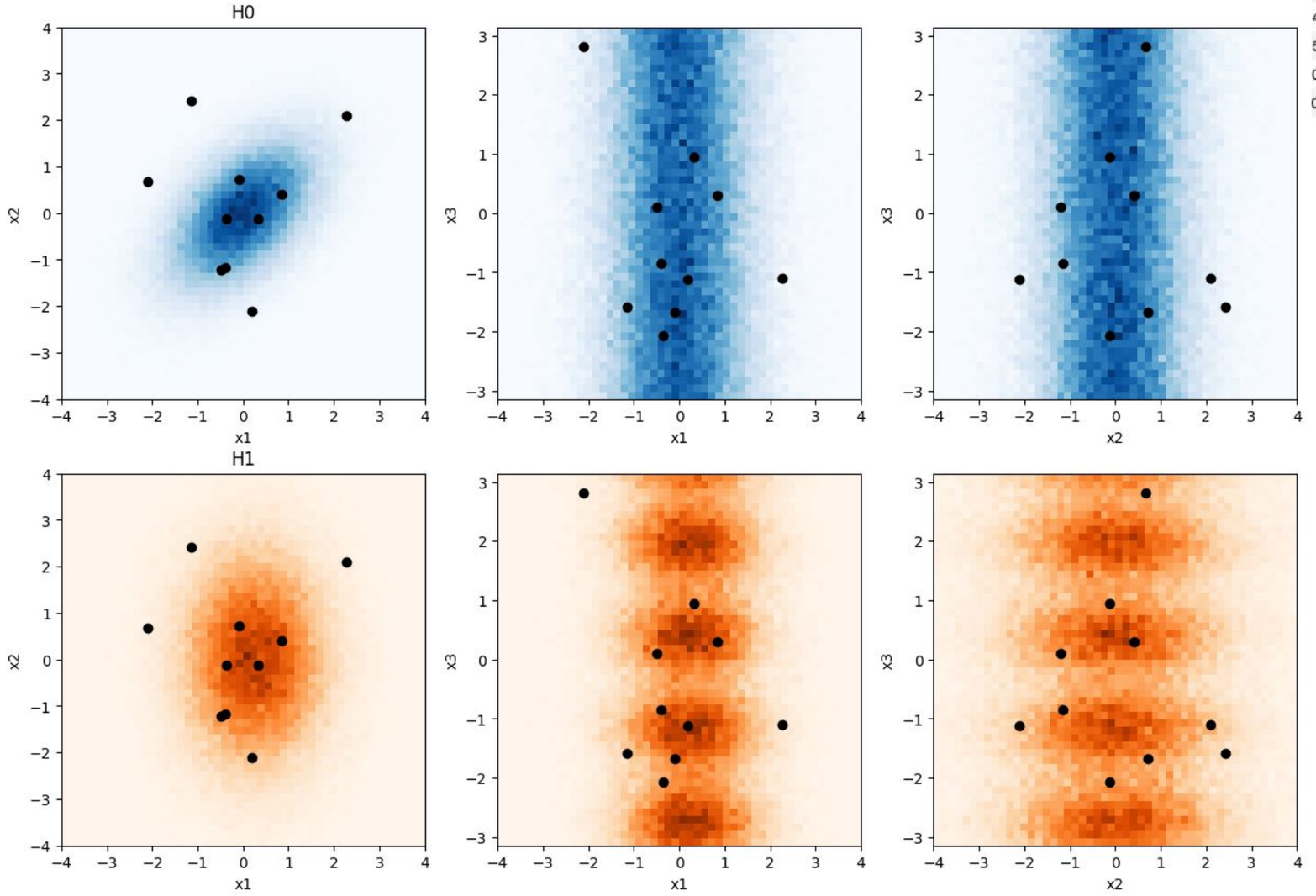
$\mathcal{H}_0$

$\mathcal{H}_1$

```
from sbi_utils import run_simulation
N_train = 100000
sim_H0 = run_simulation(N_train, hypothesis='H0')
sim_H1 = run_simulation(N_train, hypothesis='H1')
```

[15] ✓ 0.3s

	x1	x2	x3
0	-1.137580	2.427345	-1.587226
1	0.193618	-2.112575	-1.114890
2	-0.088694	0.726530	-1.673467
3	-0.390868	-1.162352	-0.849197
4	-0.358286	-0.119108	-2.076547
5	-2.098496	0.677189	2.820093
472	0.409719	0.306846	
547	2.090513	-1.099859	
024	-1.209504	0.107297	
048	-0.113135	0.954608	



Hypothesis testing example

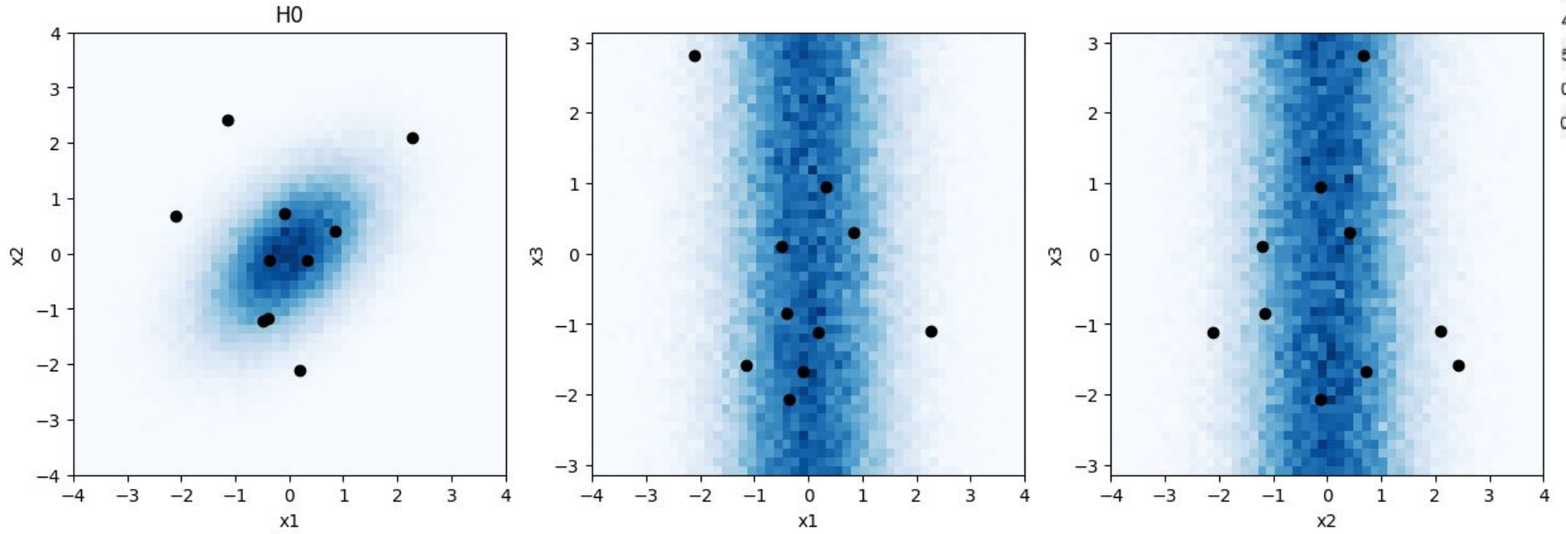
- This is a higher-dimensional problem (e.g. in 2D)

```
from sbi_utils import run_simulation
N_train = 100000
sim_H0 = run_simulation(N_train, hypothesis='H0')
sim_H1 = run_simulation(N_train, hypothesis='H1')
```

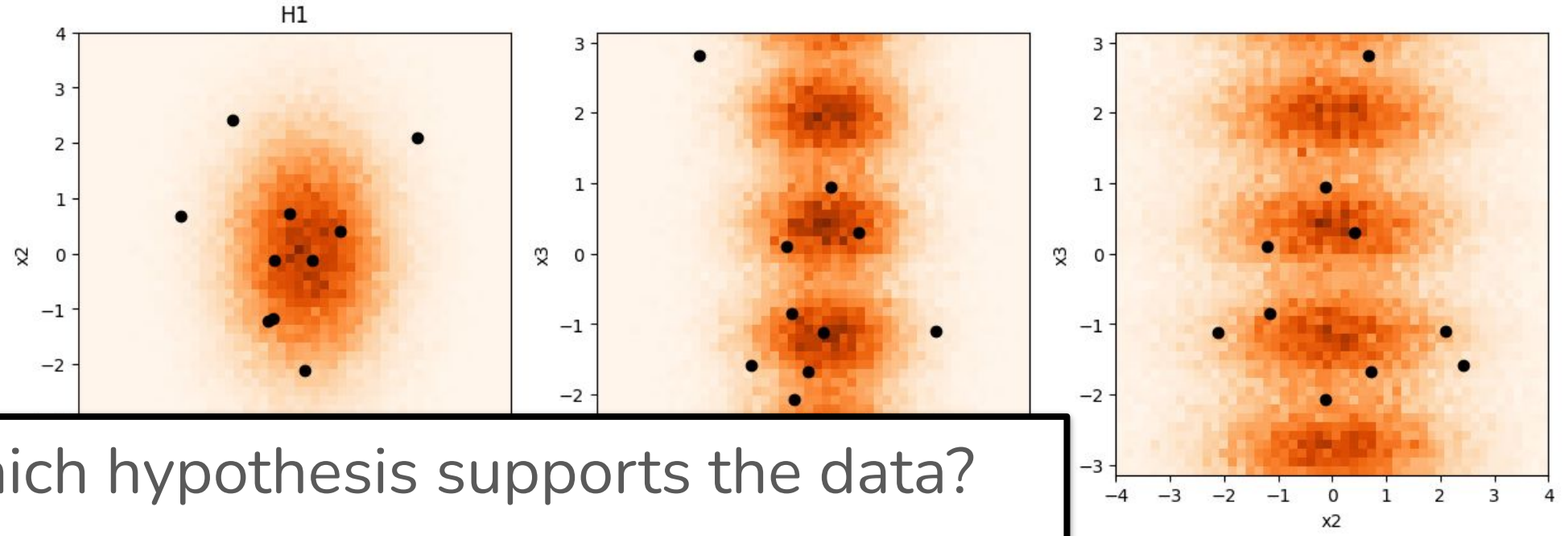
[15] ✓ 0.3s

	x1	x2	x3
0	-1.137580	2.427345	-1.587226
1	0.193618	-2.112575	-1.114890
2	-0.088694	0.726530	-1.673467
3	-0.390868	-1.162352	-0.849197
4	-0.358286	-0.119108	-2.076547
5	-2.098496	0.677189	2.820093
472	0.409719	0.306846	
547	2.090513	-1.099859	
024	-1.209504	0.107297	
048	-0.113135	0.954608	

$\mathcal{H}_0$



$\mathcal{H}_1$

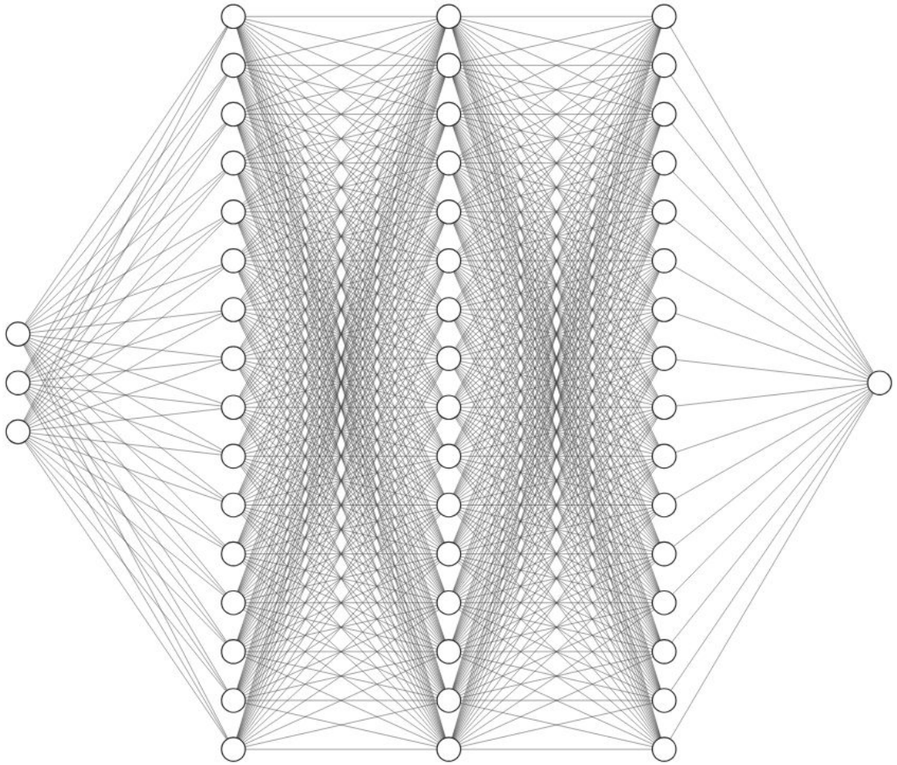


Can you tell by eye which hypothesis supports the data?

Classifier training

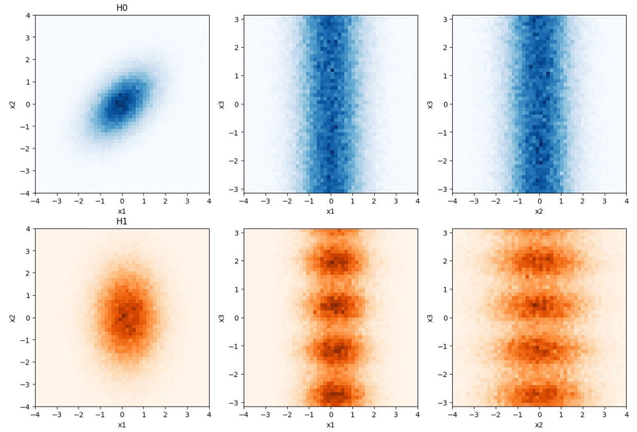
- We train a simple multi-layer perceptron (MLP) with **`torch.nn`** to classify between simulated samples

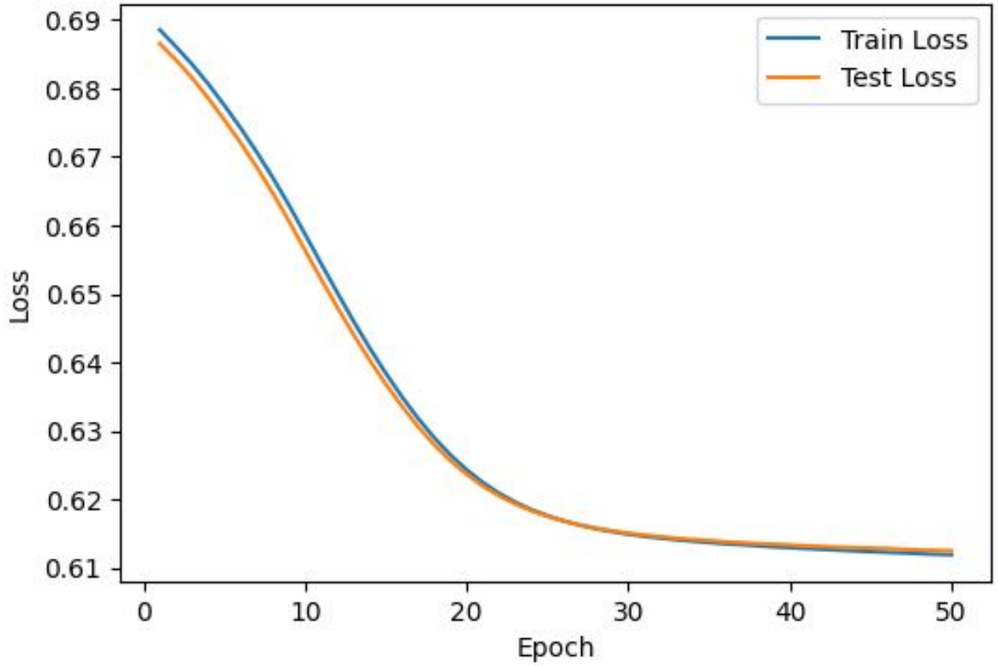
$$x_i = \begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix}_i$$



$$\hat{f}(x_i)$$

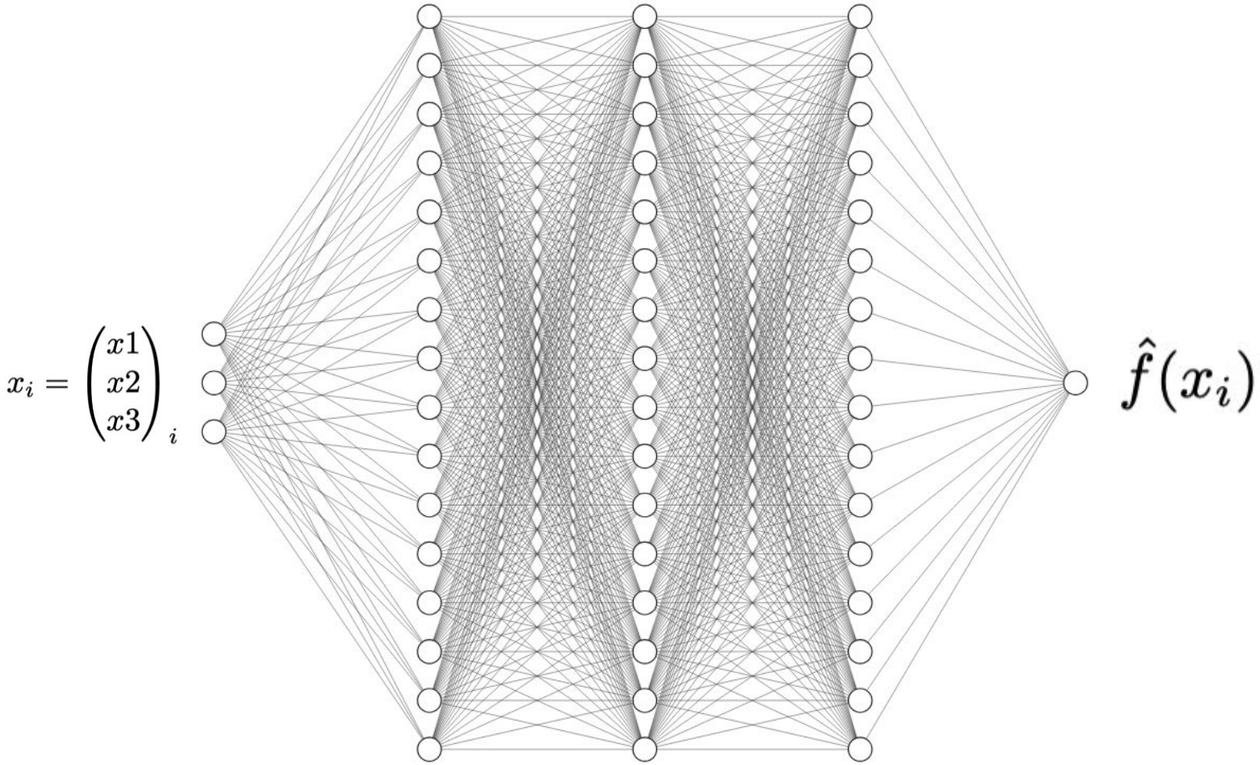
$$\mathcal{L}[f] = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln f(x_i) + (1-y_i)\ln\left(1-f(x_i)\right)$$

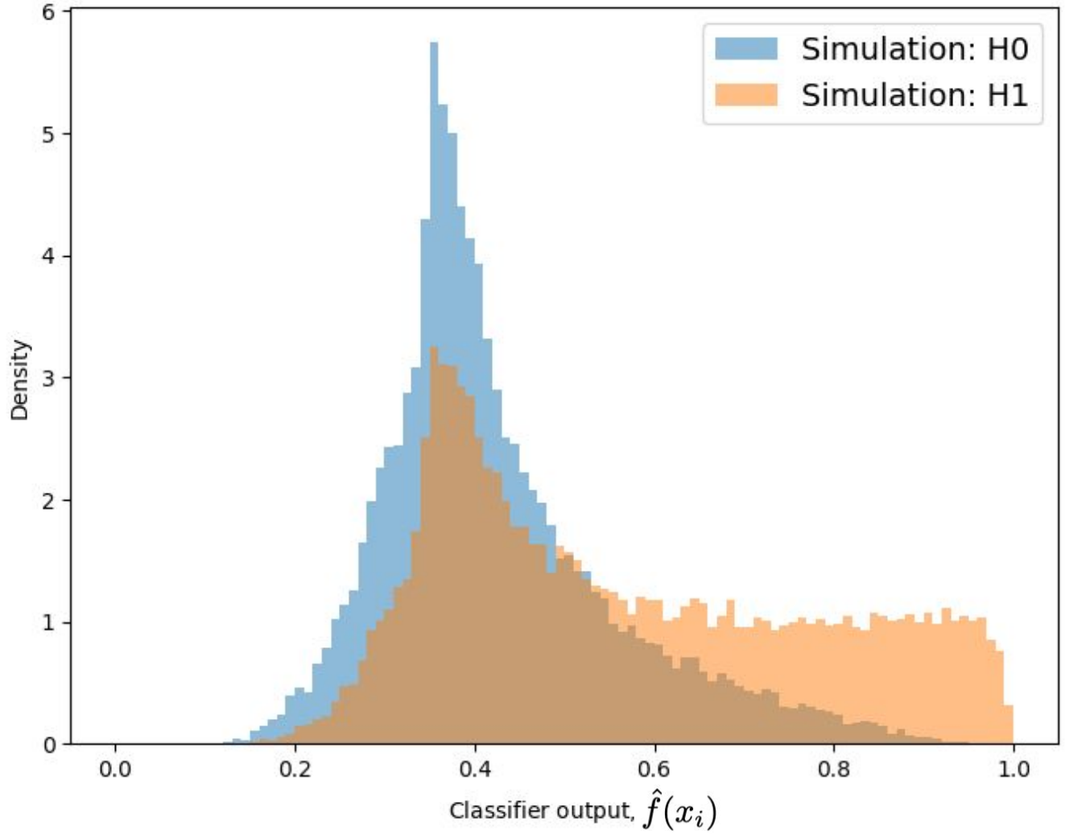




Classifier training

- After training we can evaluate the network on an independent test sample and plot the classifier output $\hat{f}(x_i)$



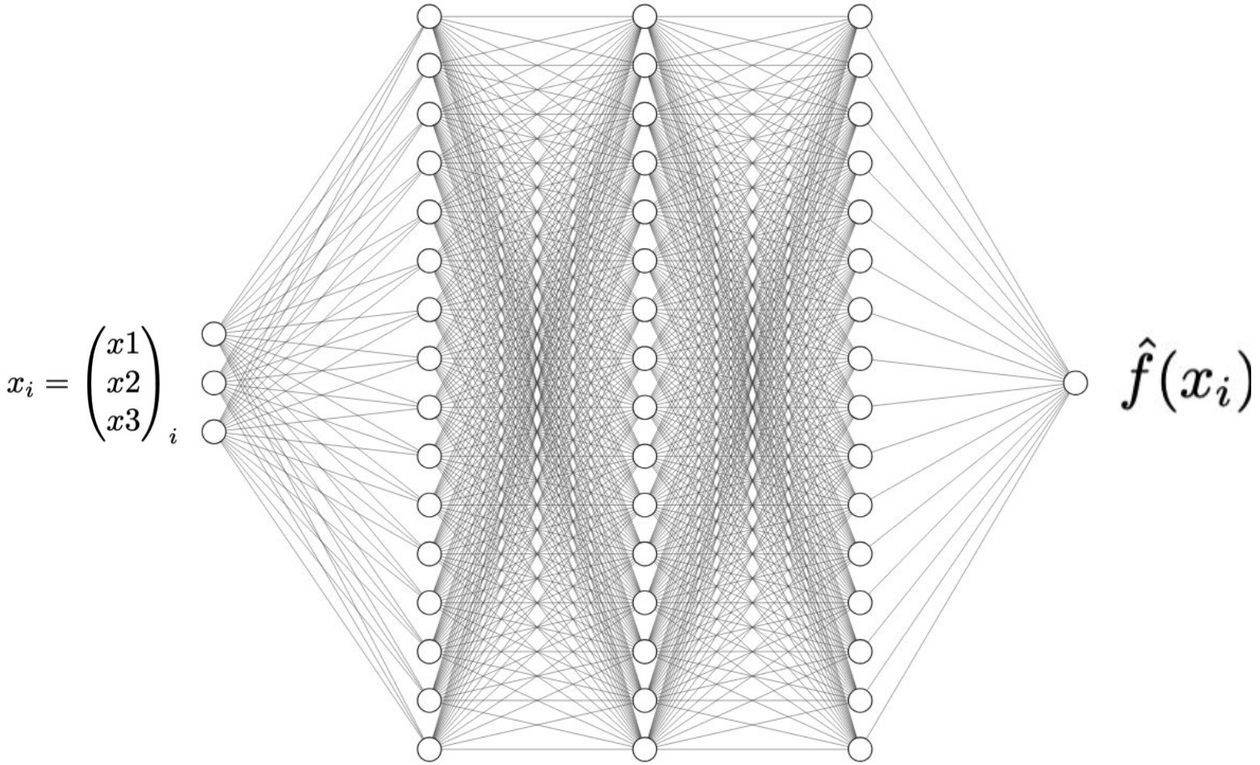


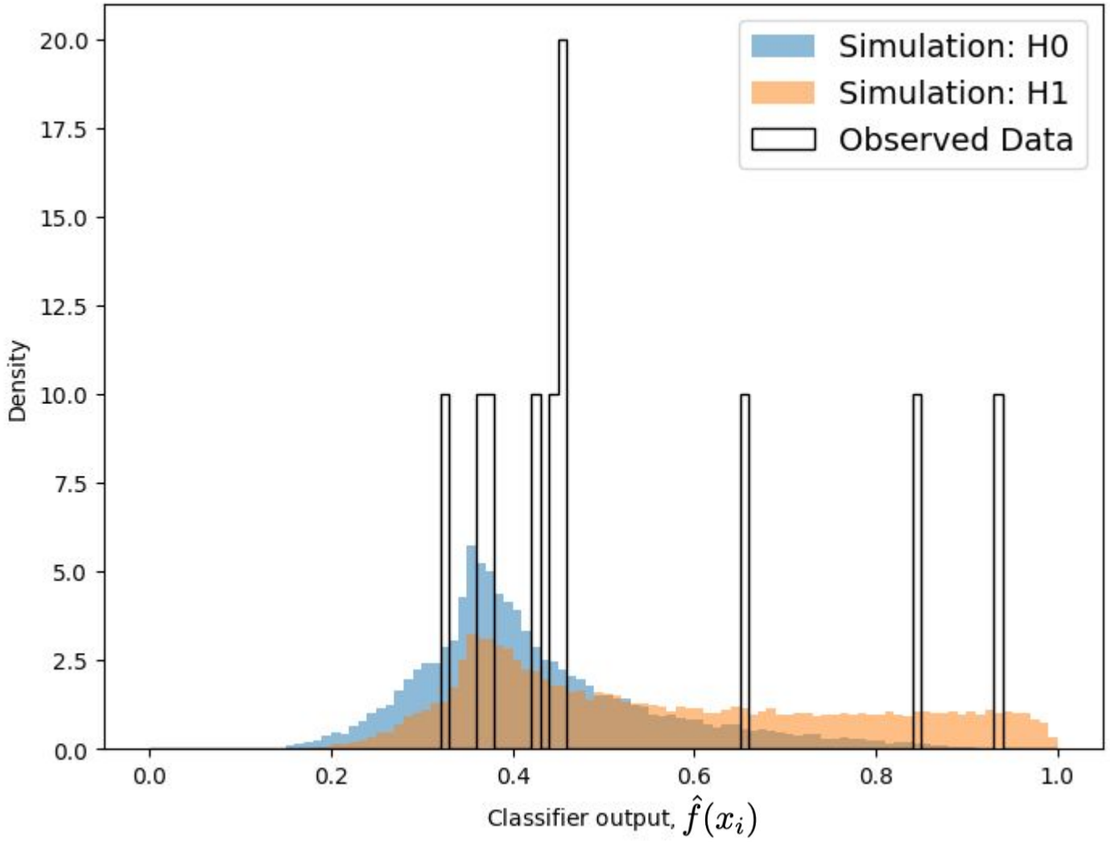
$$\mathcal{L}[f] = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln f(x_i) + (1-y_i)\ln\left(1-f(x_i)\right)$$

- Good discrimination between $\mathcal{H}_0$ and $\mathcal{H}_1$

Classifier training

- After training we can evaluate the network on an independent test sample and plot the classifier output $\hat{f}(x_i)$





$$\mathcal{L}[f] = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln f(x_i) + (1-y_i)\ln\left(1-f(x_i)\right)$$

- Good discrimination between $\mathcal{H}_0$ and $\mathcal{H}_1$
- Observed data (10 samples) shows a preference for $\mathcal{H}_1$

Inference with a classifier

Frequentist inference

$$t_\theta = -2 \ln \left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)} \right)$$

$$\frac{\hat{f}(x_i)}{1 - \hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)}$$

$$t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1 - \hat{f}(x_i)} \right)$$

Frequentist inference is defined over (hypothetical) repetitions of the experiment

Inference with a classifier

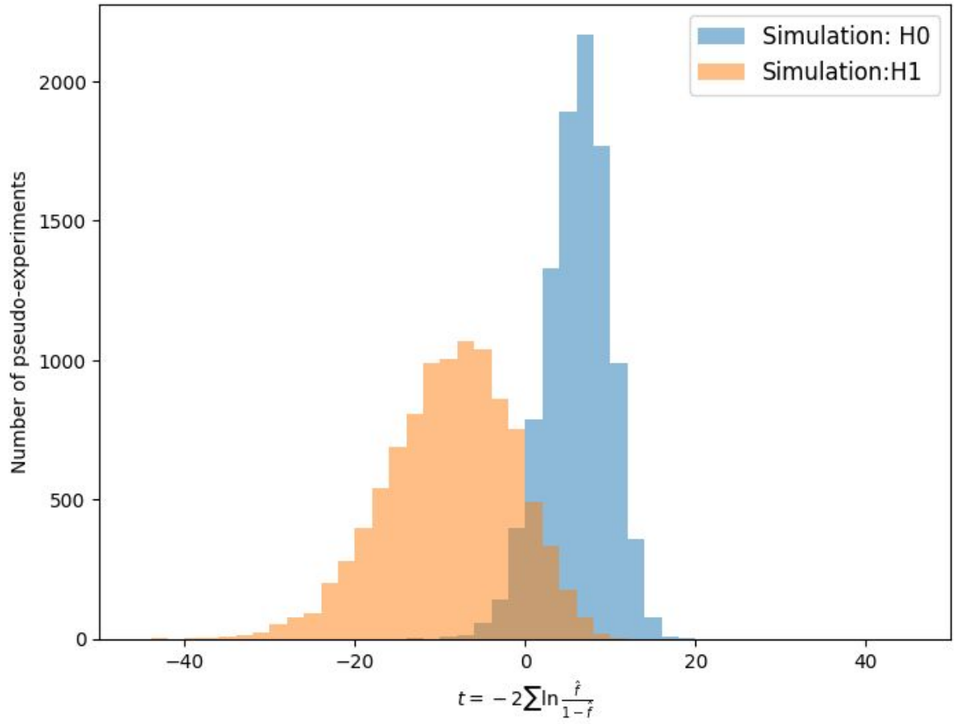
Frequentist inference

$$t_\theta = -2 \ln \left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)} \right)$$

$$\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)} \rightarrow t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \right)$$

Frequentist inference is defined over (hypothetical) repetitions of the experiment

1) Use simulator to generate “pseudo-experiments” under each hypothesis (N_{obs}=10). Repeat and build up test-statistic distribution under each hypothesis



Inference with a classifier

Frequentist inference

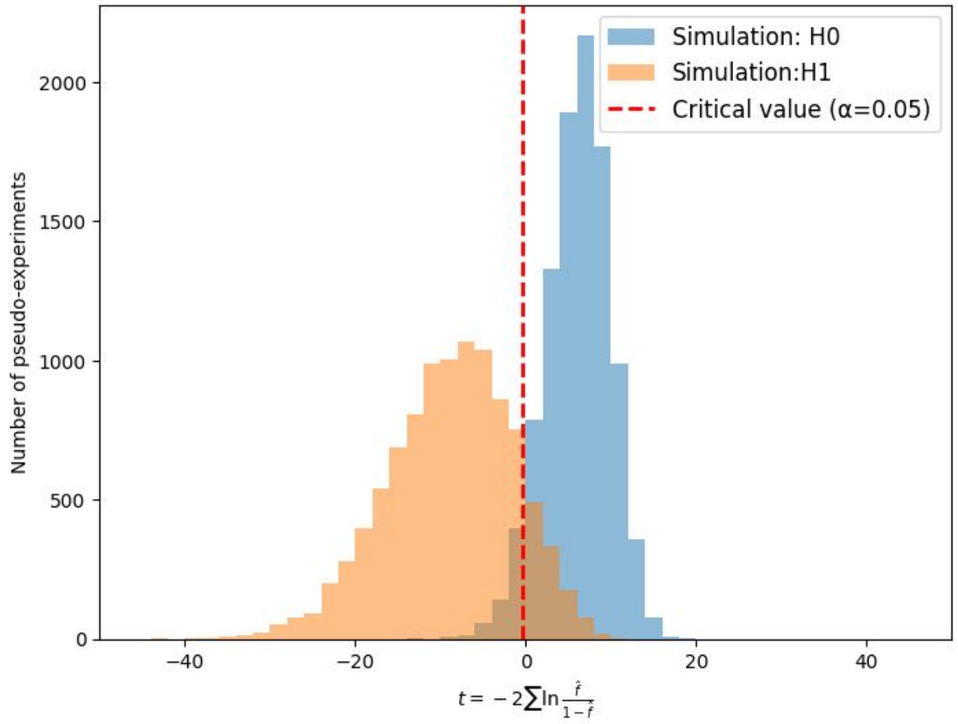
$$t_\theta = -2 \ln \left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)} \right)$$

$$\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)} \rightarrow t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \right)$$

Frequentist inference is defined over (hypothetical) repetitions of the experiment

1) Use simulator to generate "pseudo-experiments" under each hypothesis (N_{obs}=10). Repeat and build up test-statistic distribution under each hypothesis

2) Define critical threshold (e.g. α = 0.05) for rejecting $\mathcal{H}_0$



Inference with a classifier

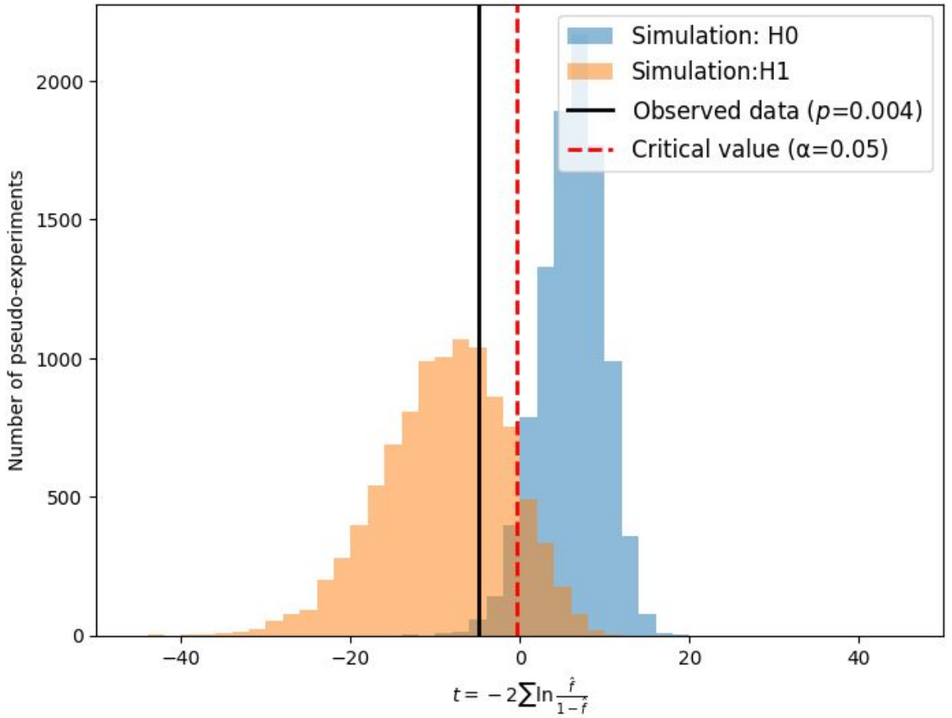
Frequentist inference

$$t_\theta = -2 \ln \left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)} \right)$$

$$\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)} \rightarrow t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \right)$$

Frequentist inference is defined over (hypothetical) repetitions of the experiment

1) Use simulator to generate “pseudo-experiments” under each hypothesis (N_{obs}=10). Repeat and build up test-statistic distribution under each hypothesis
2) Define critical threshold (e.g. α = 0.05) for rejecting $\mathcal{H}_0$
3) Calculate the test-statistic for the observed data, t_{obs}



Inference with a classifier

Frequentist inference

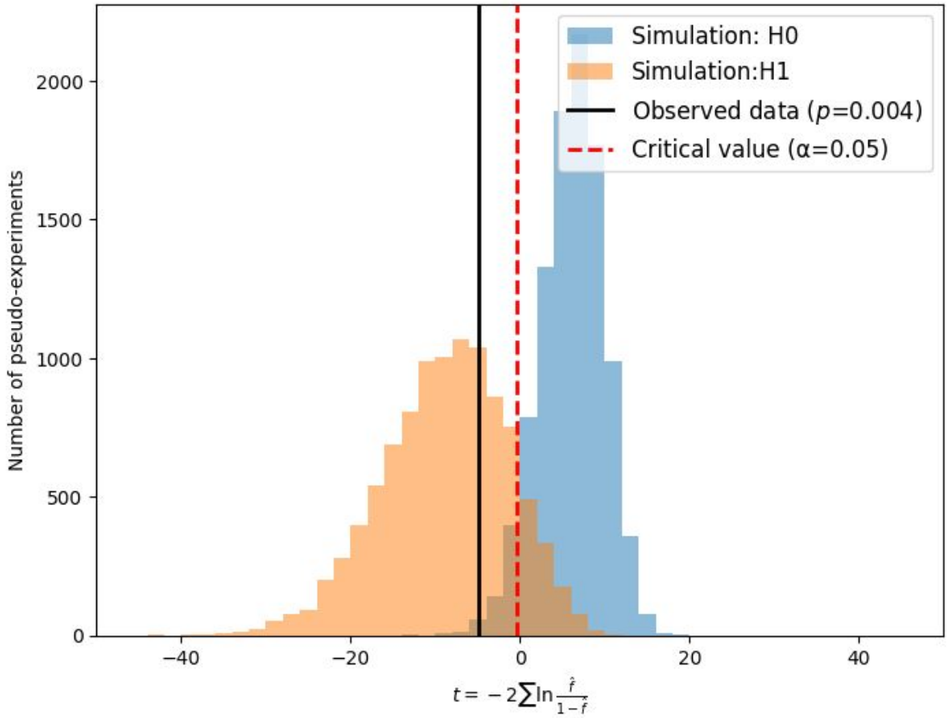
$$t_\theta = -2 \ln \left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)} \right)$$

$$\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)} \rightarrow t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \right)$$

Frequentist inference is defined over (hypothetical) repetitions of the experiment

1) Use simulator to generate "pseudo-experiments" under each hypothesis (N_{obs}=10). Repeat and build up test-statistic distribution under each hypothesis
2) Define critical threshold (e.g. α = 0.05) for rejecting $\mathcal{H}_0$
3) Calculate the test-statistic for the observed data, t_{obs}

p-value (with respect to $\mathcal{H}_0$): is the probability of obtaining data at least as extreme as what was observed, assuming $\mathcal{H}_0$ is true, over a long run of repeated identical experiments



Inference with a classifier

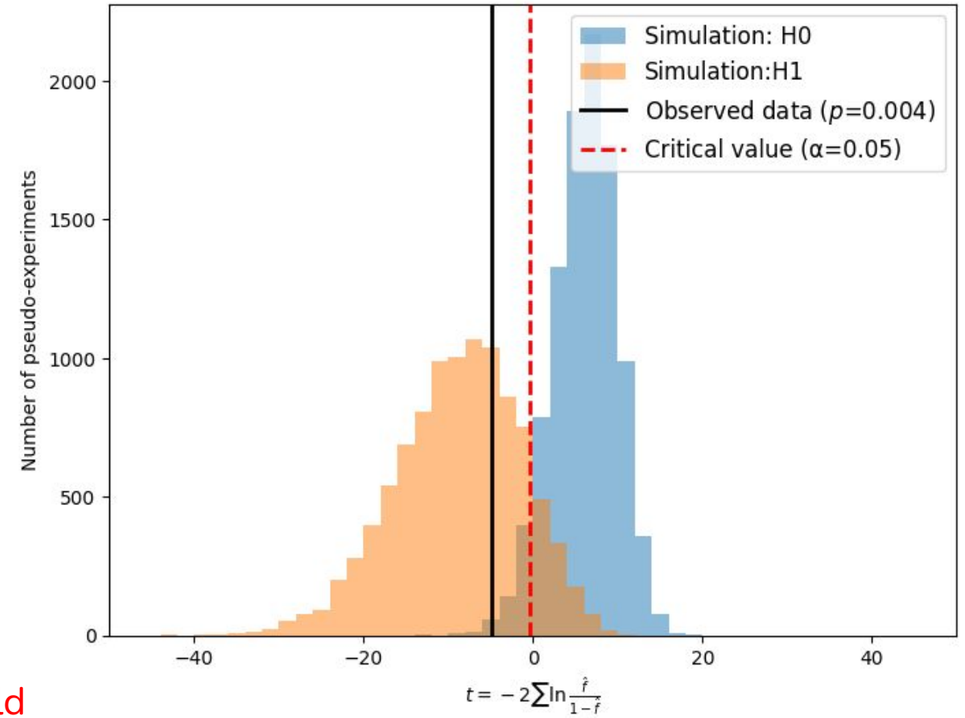
Frequentist inference

$$t_\theta = -2 \ln \left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)} \right)$$

$$\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)} \rightarrow t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \right)$$

Frequentist inference is defined over (hypothetical) repetitions of the experiment

1) Use simulator to generate "pseudo-experiments" under each hypothesis (N_{obs}=10). Repeat and build up test-statistic distribution under each hypothesis
2) Define critical threshold (e.g. α = 0.05) for rejecting $\mathcal{H}_0$
3) Calculate the test-statistic for the observed data, t_{obs}



p-value (with respect to $\mathcal{H}_0$): is the probability of obtaining data at least as extreme as what was observed, assuming $\mathcal{H}_0$ is true, over a long run of repeated identical experiments

- Calculate by integrating left-hand tail of $\mathcal{H}_0$ distribution up to t_{obs}
- In this case we find p = 0.004 (i.e. very unlikely!). This is beyond critical threshold
- We reject $\mathcal{H}_0$ at the 1-α = 95% confidence level
- The observed data is very consistent with $\mathcal{H}_1$

Inference with a classifier

Frequentist inference

$$t_\theta = -2 \ln \left(\frac{p(\mathcal{D}|\theta)}{p(\mathcal{D}|\theta_0)} \right)$$

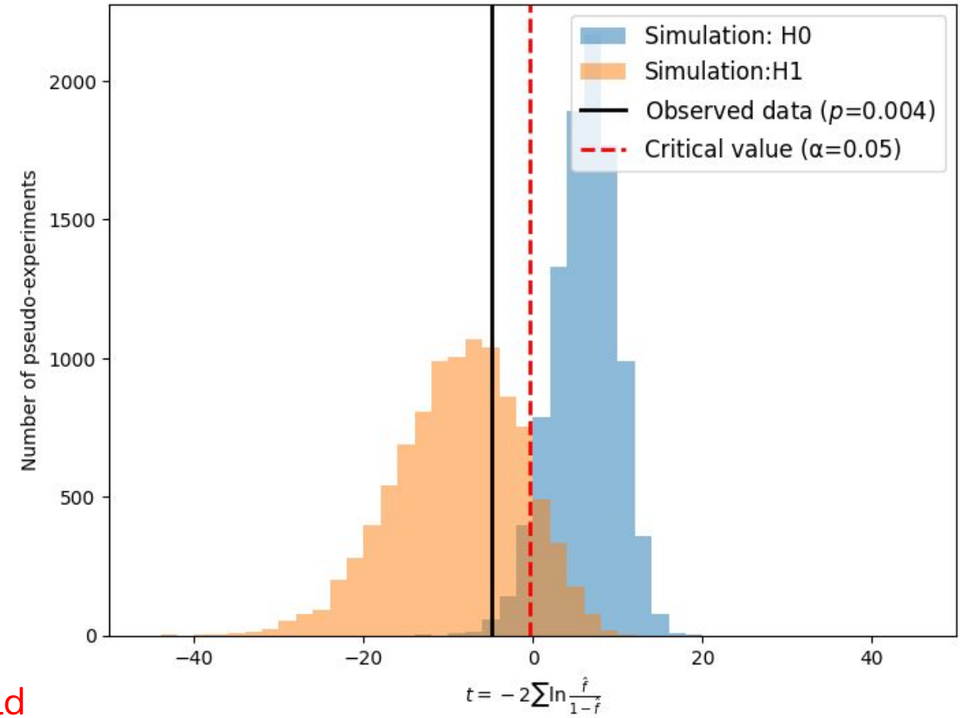
$$\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \approx \frac{p(x_i|\mathcal{H}_1)}{p(x_i|\mathcal{H}_0)} \rightarrow t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)} \right)$$

Frequentist inference is defined over (hypothetical) repetitions of the experiment

1) Use simulator to generate "pseudo-experiments" under each hypothesis (N_{obs}=10). Repeat and build up test-statistic distribution under each hypothesis
2) Define critical threshold (e.g. α = 0.05) for rejecting $\mathcal{H}_0$
3) Calculate the test-statistic for the observed data, t_{obs}

p-value (with respect to $\mathcal{H}_0$): is the probability of obtaining data at least as extreme as what was observed, assuming $\mathcal{H}_0$ is true, over a long run of repeated identical experiments

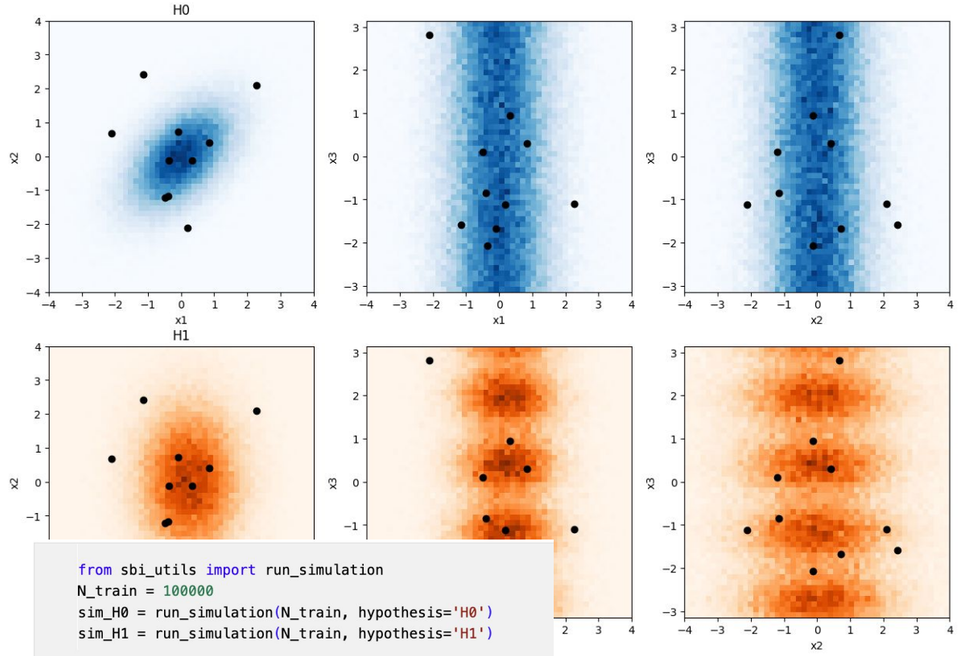
- Calculate by integrating left-hand tail of $\mathcal{H}_0$ distribution up to t_{obs}
- In this case we find p = 0.004 (i.e. very unlikely!). This is beyond critical threshold
- We reject $\mathcal{H}_0$ at the 1-α = 95% confidence level
- The observed data is very consistent with $\mathcal{H}_1$



Particle A is spin-1!

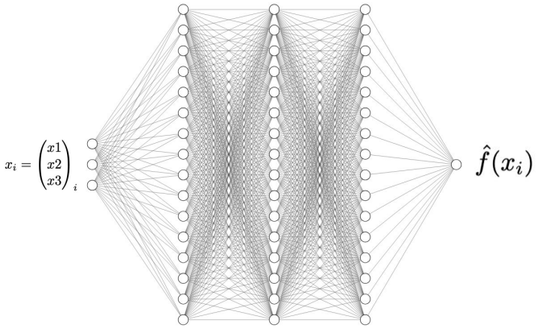
Summary: hypothesis testing

- We were presented with a research problem with an unknown (intractable) likelihood: $p(x|\theta)$
- Nevertheless, we had access to a faithful simulation of the data: $x_i^{\text{sim}} \sim p(x|\theta)$
- We trained a binary classifier over full (3D) feature space to distinguish $\mathcal{H}_0$ (spin-0) from $\mathcal{H}_1$ (spin-1)
- Use output of classifier to approximate log-likelihood-ratio test-statistic → compare observed data to pseudo-experiments
- Our data was inconsistent with $\mathcal{H}_0$ with a p-value of 0.004 → We reject the null hypothesis and conclude A is spin-one!

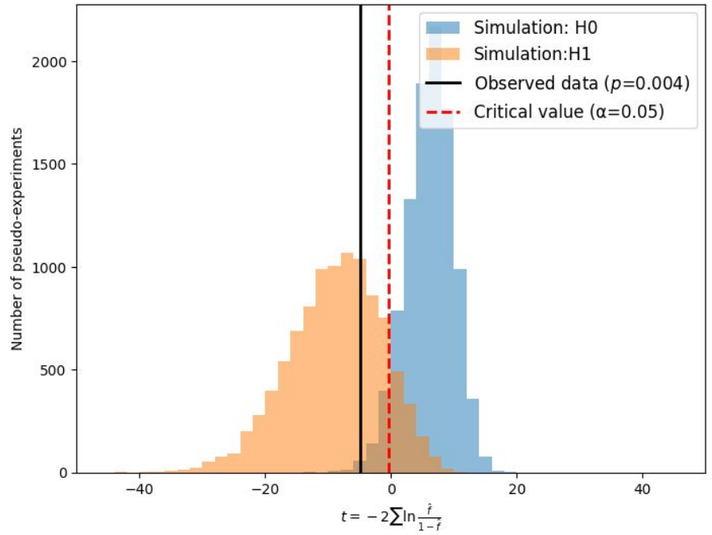



```
from sbi_utils import run_simulation
N_train = 100000
sim_H0 = run_simulation(N_train, hypothesis='H0')
sim_H1 = run_simulation(N_train, hypothesis='H1')
```

[15] ✓ 0.3s

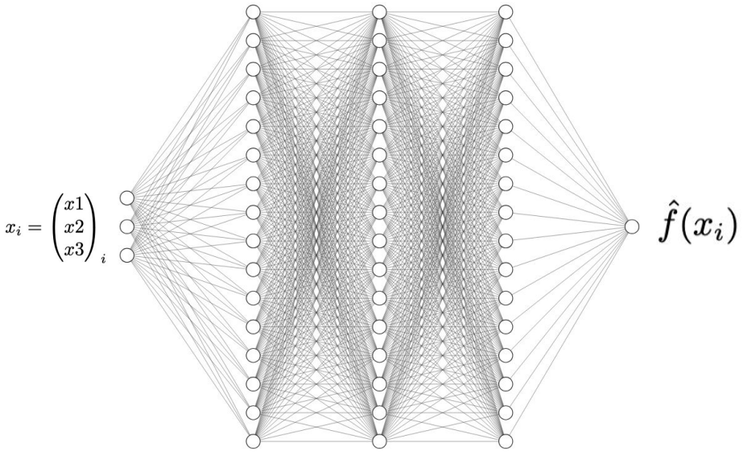


$$t = -2 \ln \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_0)} \approx -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \left(\frac{\hat{f}(x_i)}{1 - \hat{f}(x_i)} \right)$$



- End-to-end example of using a ML classifier for SBI (hypothesis testing)!
  - **This is not architecture specific and generalises to more complex data**

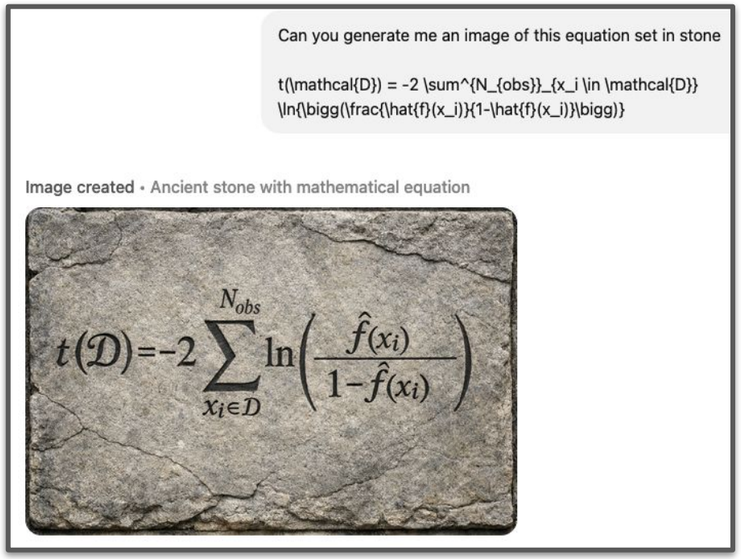
Amortized inference



- Classifier output is just a function defined over the feature space: $\hat{f}(x_i)$

- Assuming that the experimental conditions remain the same (i.e. the probability density does not change), we can simply re-calculate the test-statistic for any new observations:

$$t \equiv t(\mathcal{D}) = -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln\left(\frac{\hat{f}(x_i)}{1-\hat{f}(x_i)}\right)$$

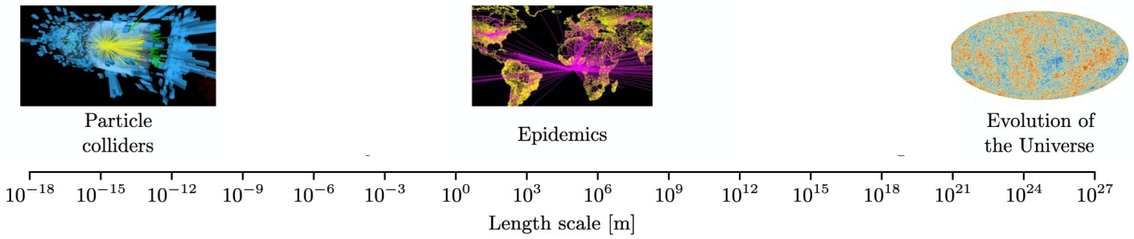


Credit: ChatGPT

- **Inference procedure is "amortized" for future experiments:**
  - We do not need to retrain the classifier
  - Very useful when dealing with problems with extremely complicated likelihoods

- See extension in [ADD LINK] to investigate amortized inference for particle spin example with 50 Y decays

Parameter estimation



- We have shown how to use a classifier for a simple hypothesis test
- More common problem: **parameter estimation**
  - We have a model $p(x|\theta)$ that describes data, where θ is a parameter of the model
  - Use observed data to infer θ i.e. extract best fit value and confidence intervals
  - θ is e.g. mass of a new particle produced at LHC, R_0 of a new infectious disease, Hubble constant governing Universe expansion
- For this inference task, we need to learn the conditional probability density ratio: $\frac{p(x|\theta)}{p(x|\theta_0)}$
- *N.B. extending previous idea to "composite hypothesis testing" over ensemble of hypotheses* $\frac{p(x|\mathcal{H}_1)}{p(x|\mathcal{H}_0)} \longrightarrow \frac{p(x|\{\mathcal{H}_1\}_\theta)}{p(x|\mathcal{H}_0)}$
- Log-likelihood ratio test-statistic becomes conditional on the parameter θ: $t(\mathcal{D}|\theta) = -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \frac{p(x_i|\theta)}{p(x_i|\theta_0)}$
  - This is what we will use for inference!

Aside: frequentist parameter estimation

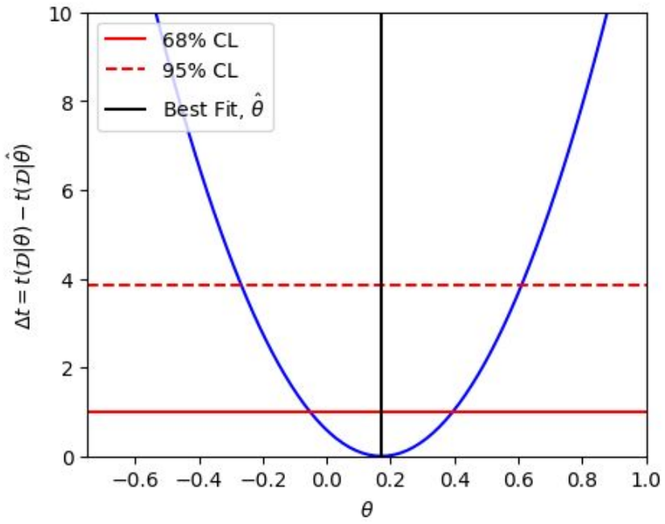
$$t(\mathcal{D}|\hat{\theta}) = \min[t(\mathcal{D}|\theta)] \equiv t_{\min}$$

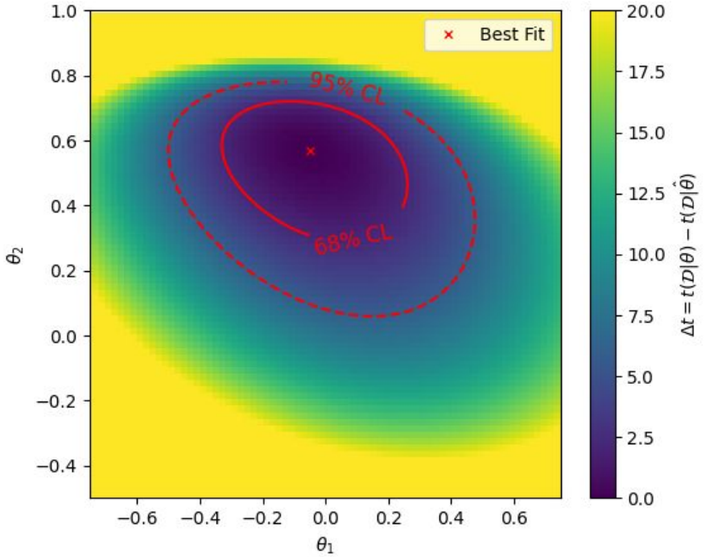
- For a fixed dataset, $\mathcal{D}$, the point in parameter space $\hat{\theta}$ which minimizes the test-statistic is the best-fit value

- What about confidence interval (CI) estimation?
  - Rigorous frequentist CI estimation requires Neyman construction
  - Most cases we make use of Wilks' Theorem:
    - (Delta) Log-likelihood ratio test statistic asymptotically ($N_{obs} \to \infty$) approaches the χ^2 distribution with *n* degrees of freedom, where *n* is the dimensionality of θ (under the null hypothesis)
    - In a nutshell: calculate $\Delta t(\mathcal{D}|\theta) = t(\mathcal{D}|\theta) - t(\mathcal{D}|\hat{\theta})$ and use properties of the χ^2 distribution to infer confidence intervals

$$t(\mathcal{D}|\theta) = -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \frac{p(x_i|\theta)}{p(x_i|\theta_0)}$$

1D parameter vector example

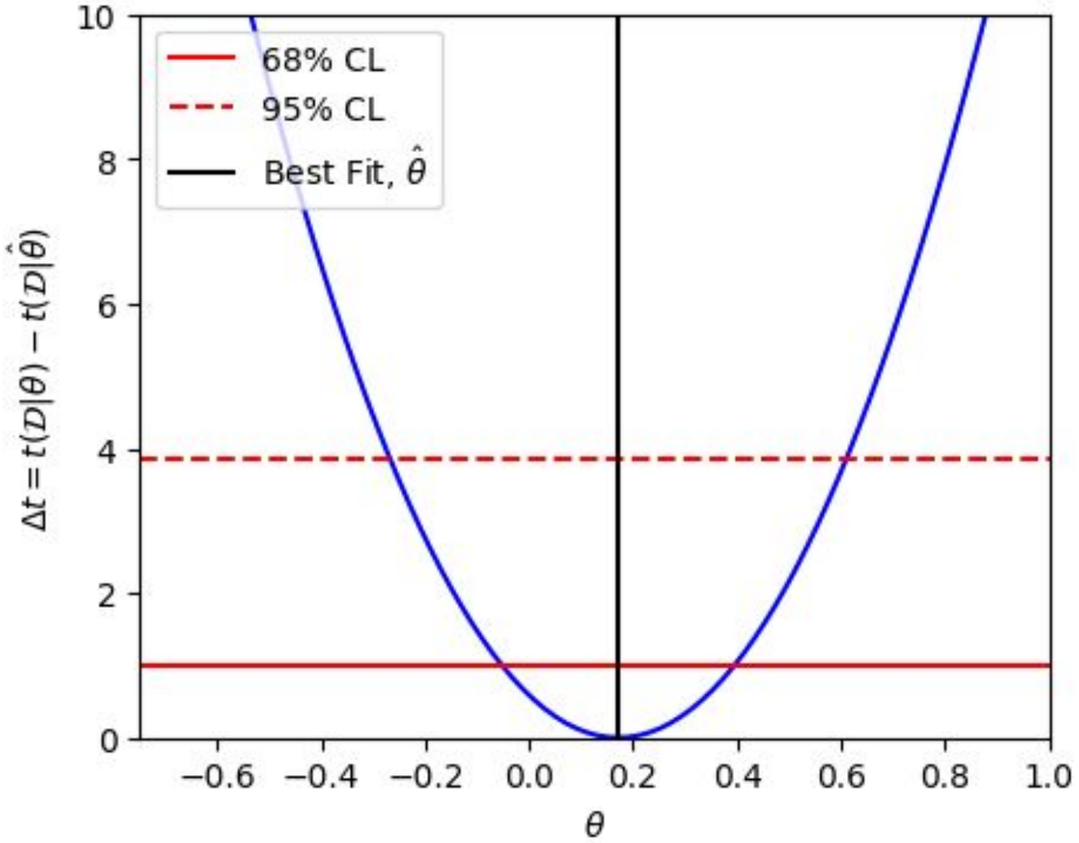




2D parameter vector example

Aside: frequentist parameter estimation

- In a nutshell: calculate $\Delta t(\mathcal{D}|\theta) = t(\mathcal{D}|\theta) - t(\mathcal{D}|\hat{\theta})$ and use properties of the χ^2 distribution to infer confidence intervals
  - Best-fit where $\Delta t(\mathcal{D}|\theta)$ equals zero (by construction)

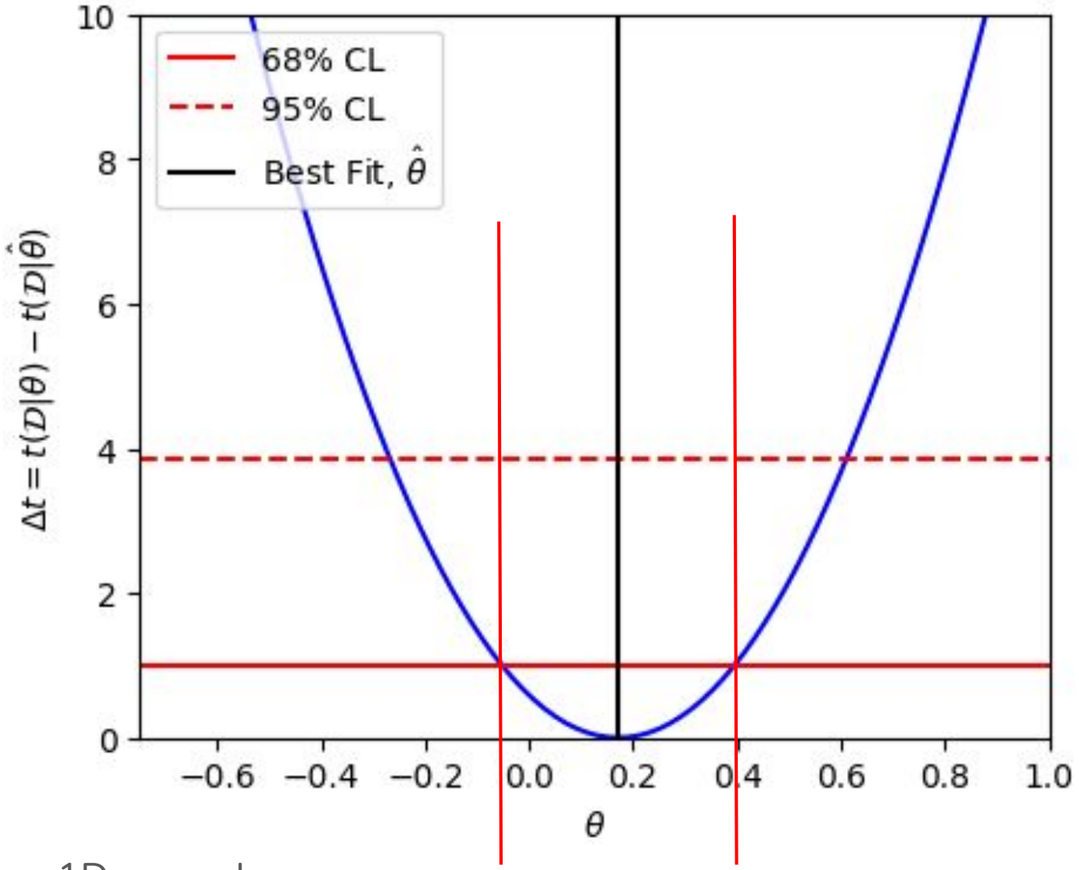


1D example:

- 68% CL is region defined by $\Delta t < 1$
- 95% CL is region defined by $\Delta t < 3.84$

Aside: frequentist parameter estimation

- In a nutshell: calculate $\Delta t(\mathcal{D}|\theta) = t(\mathcal{D}|\theta) - t(\mathcal{D}|\hat{\theta})$ and use properties of the χ^2 distribution to infer confidence intervals
  - Best-fit where $\Delta t(\mathcal{D}|\theta)$ equals zero (by construction)

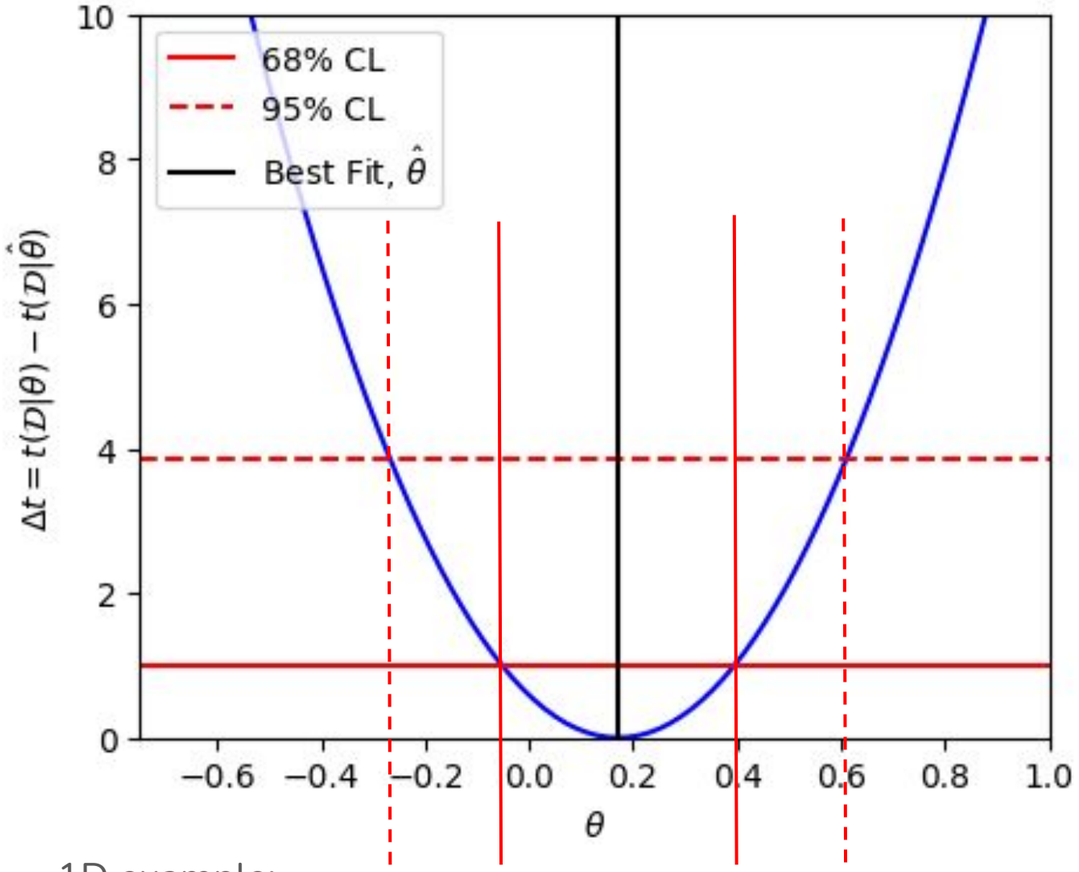


1D example:

- 68% CL is region defined by $\Delta t < 1$
- 95% CL is region defined by $\Delta t < 3.84$

Aside: frequentist parameter estimation

- In a nutshell: calculate $\Delta t(\mathcal{D}|\theta) = t(\mathcal{D}|\theta) - t(\mathcal{D}|\hat{\theta})$ and use properties of the χ^2 distribution to infer confidence intervals
  - Best-fit where $\Delta t(\mathcal{D}|\theta)$ equals zero (by construction)

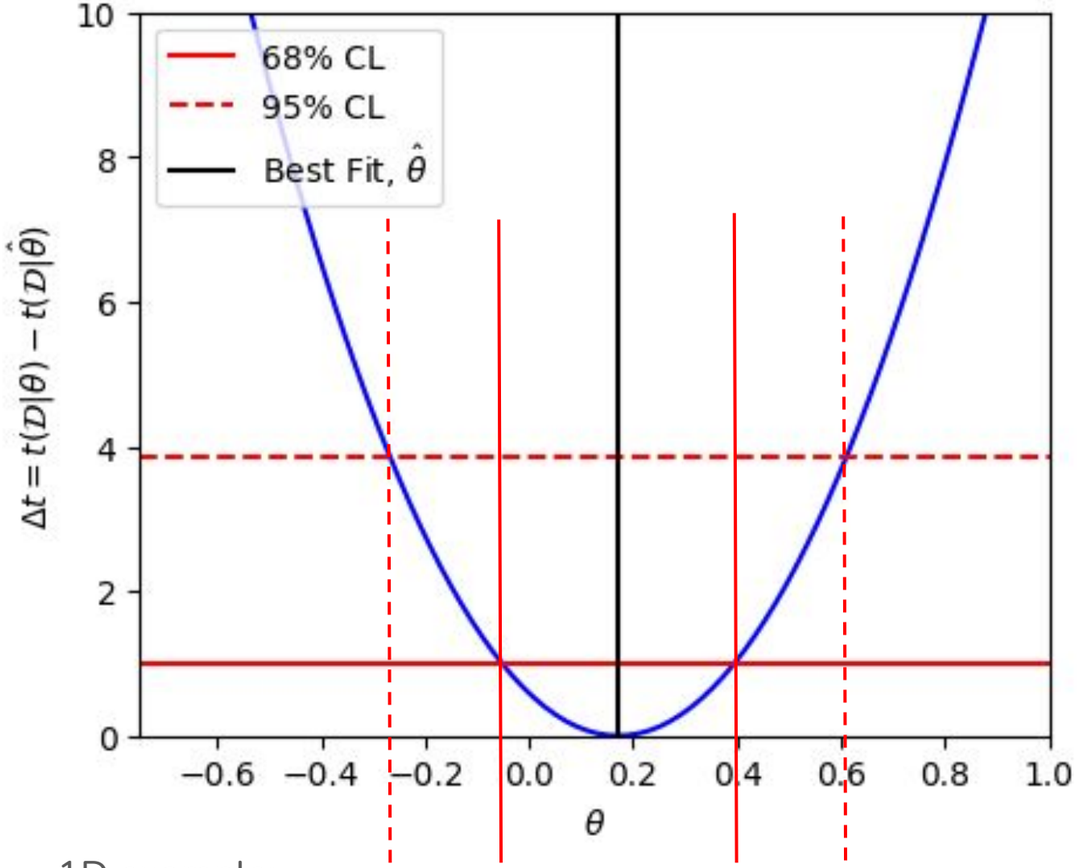


1D example:

- 68% CL is region defined by $\Delta t < 1$
- 95% CL is region defined by $\Delta t < 3.84$

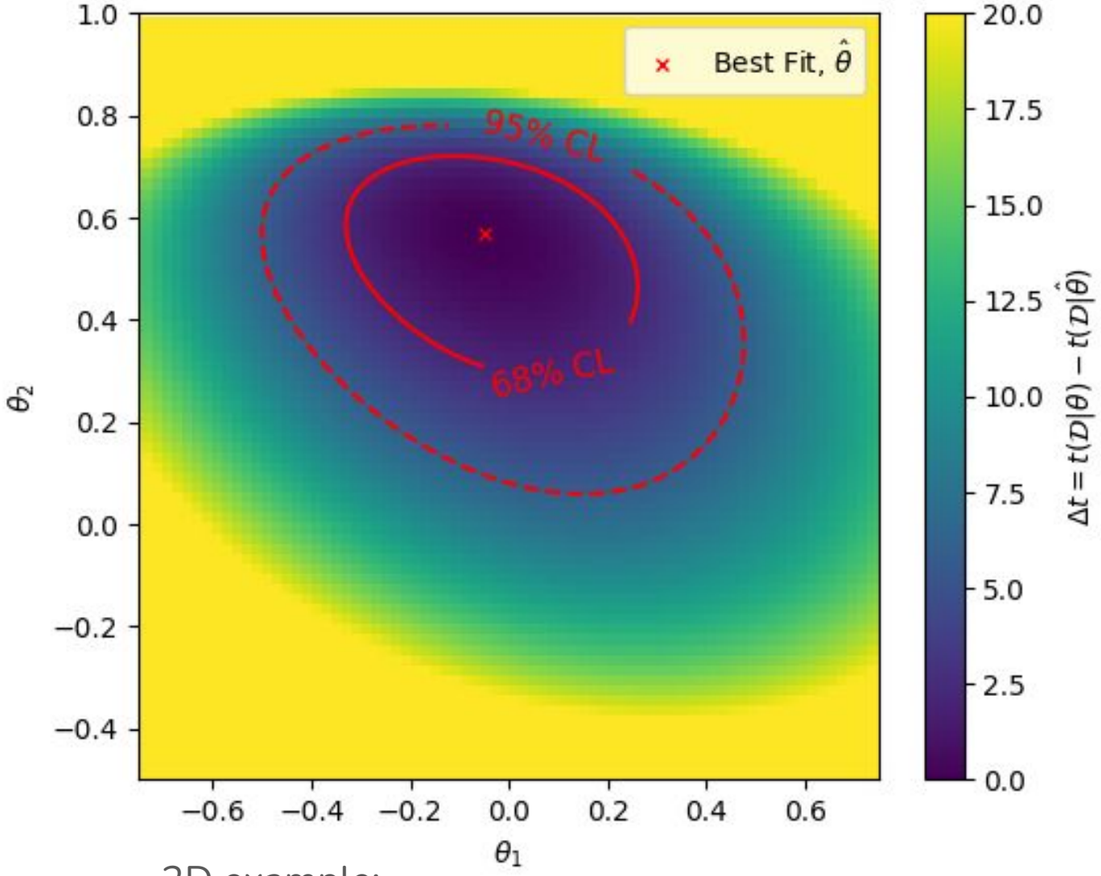
Aside: frequentist parameter estimation

- In a nutshell: calculate $\Delta t(\mathcal{D}|\theta) = t(\mathcal{D}|\theta) - t(\mathcal{D}|\hat{\theta})$ and use properties of the χ^2 distribution to infer confidence intervals
  - Best-fit where $\Delta t(\mathcal{D}|\theta)$ equals zero (by construction)



1D example:

- 68% CL is region defined by $\Delta t < 1$
- 95% CL is region defined by $\Delta t < 3.84$



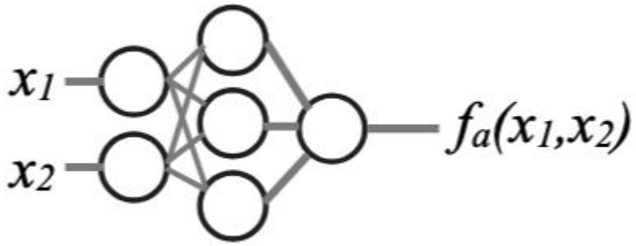
2D example:

- 68% CL is region defined by $\Delta t < 2.30$
- 95% CL is region defined by $\Delta t < 5.99$

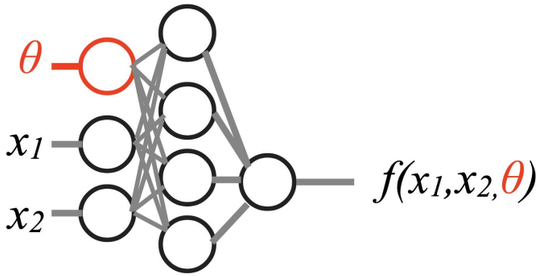
Parameter estimation with classifiers

$$\frac{p(x_i|\theta)}{p(x_i|\theta_0)}$$

- **Can we use Machine Learning to estimate the conditional log-likelihood ratio test-statistic for inference?**
  - Use a parametric classifier. Simple idea: extend input feature space from $x \longrightarrow x, \theta$
    - Standard classifier is a function of the input features: $\hat{f}(x_i)$



      - Evaluates to a (single) real number
    - Parametric classifier is a function of both input features and parameters: $\hat{f}(x_i, \theta_i)$



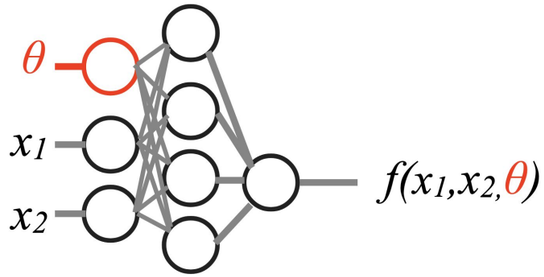
      - Result that is parameterized in terms of θ
      - Yields different output values for different values of the parameters θ_i

- **But how do we train these classifiers for inferences?**

Parameter estimation with classifiers

$$\frac{p(x_i|\theta)}{p(x_i|\theta_0)}$$

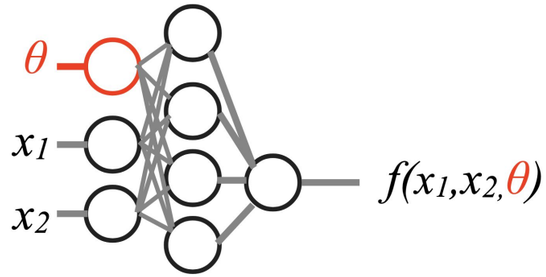
- **Can we use Machine Learning to estimate the conditional log-likelihood ratio test-statistic for inference?**
  - Use a parametric classifier. Simple idea: extend input feature space from $x \longrightarrow x, \theta$
  - All we need is a simulator that can generate data for any value of θ: $x_i^{\text{sim}} \sim p(x|\theta)$

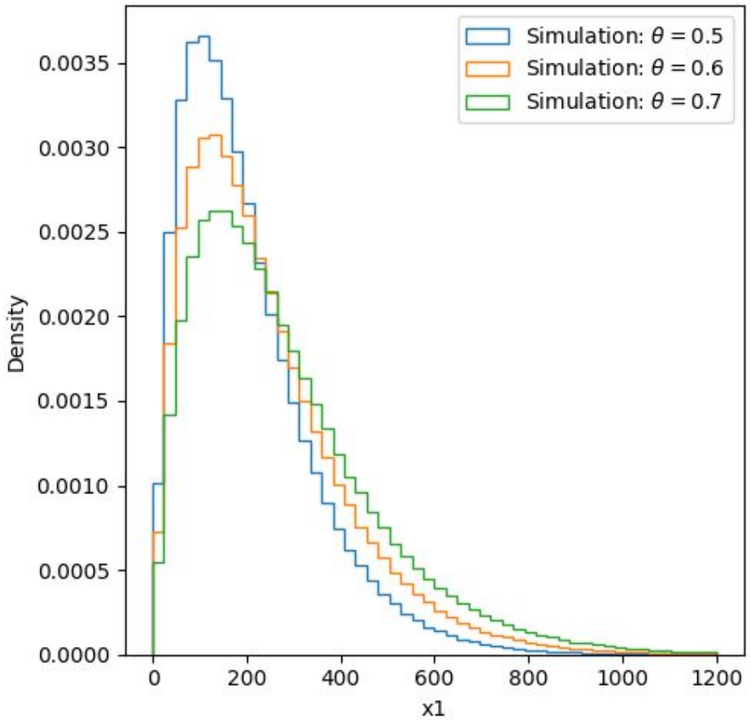


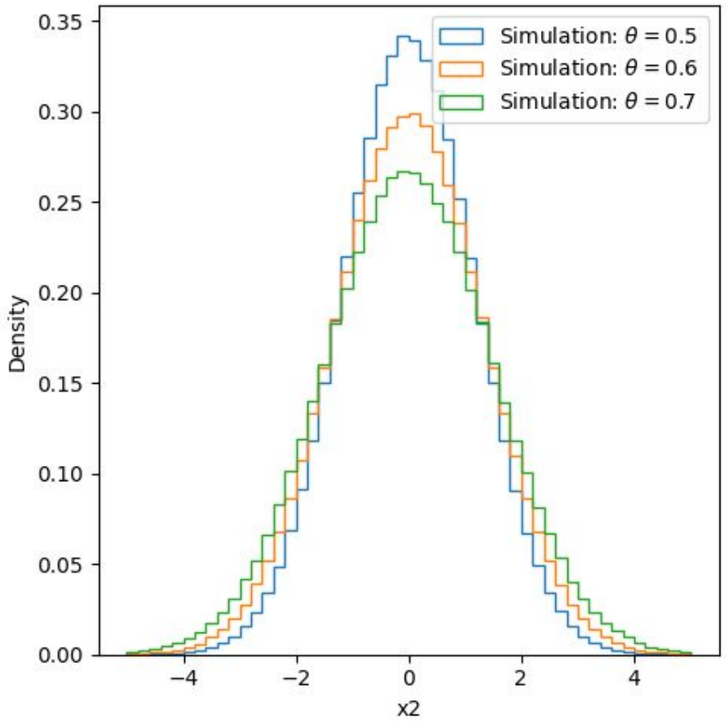
Parameter estimation with classifiers

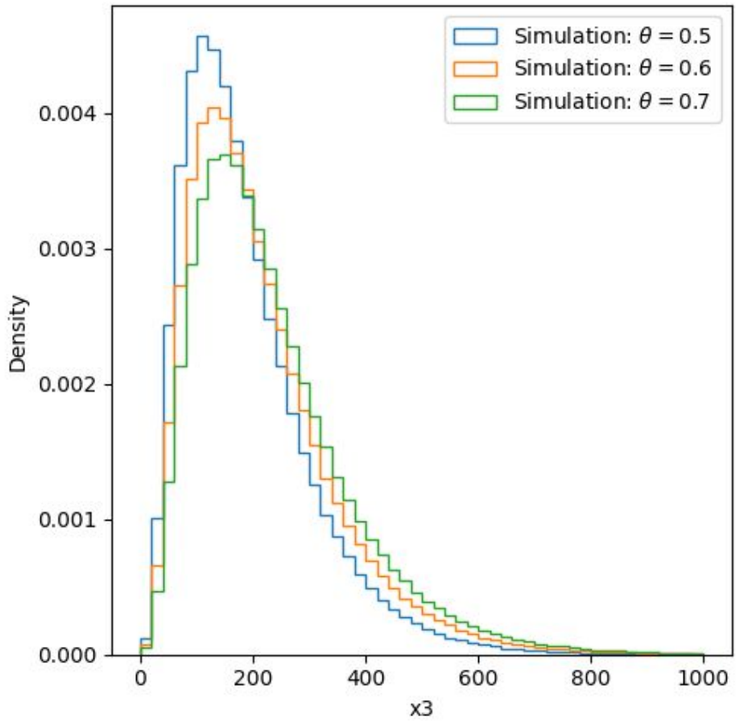
$$\frac{p(x_i|\theta)}{p(x_i|\theta_0)}$$

- **Can we use Machine Learning to estimate the conditional log-likelihood ratio test-statistic for inference?**
  - Use a parametric classifier. Simple idea: extend input feature space from $x \longrightarrow x, \theta$
  - All we need is a simulator that can generate data for any value of θ: $x_i^{\text{sim}} \sim p(x|\theta)$
    - e.g. 3D input feature space (`x1`, `x2`, `x3`) with one parameter θ





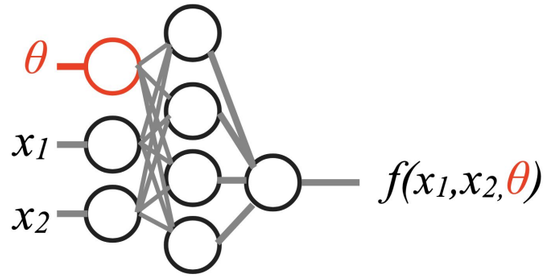




Parameter estimation with classifiers

$$\frac{p(x_i|\theta)}{p(x_i|\theta_0)}$$

- **Can we use Machine Learning to estimate the conditional log-likelihood ratio test-statistic for inference?**
  - Use a parametric classifier. Simple idea: extend input feature space from $x \longrightarrow x, \theta$
  - All we need is a simulator that can generate data for any value of θ: $x_i^{\text{sim}} \sim p(x|\theta)$
  - Use simulation to train a binary classifier, $f(x_i, \theta_i)$, by minimizing BCE loss:



$$\mathcal{L}[f] = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln f(x_i, \theta_i) + (1 - y_i) \ln\left(1 - f(x_i, \theta_i)\right)$$

$$\mathcal{H}_0 : y_i = 0$$
$$\mathcal{H}_1 : y_i = 1$$

  - Class 0 ($\mathcal{H}_0$): draw samples from simulator with fixed $\theta = \theta_0$: $\boxed{x_i^{\mathcal{H}_0} \sim p(x|\theta_0)}$
    - θ_0 = reference hypothesis. Final parameter estimation is independent of this value, but it helps to pick something sensible
  - Class 1 ($\{\mathcal{H}_1\}_\theta$): draw samples from simulator with various θ values: $\boxed{x_i^{\mathcal{H}_1}, \theta_i^{\mathcal{H}_1} \sim p(x|\theta)}$
    - If possible, generate samples to be continuous in θ e.g. sampled from uniform distribution over *sensible* range
    - In practice, often simpler to generate sub-samples with discrete steps in θ (interpolates if steps are *sufficiently fine-grained*)
  - Assuming "balanced classes", our trained classifier approximates:

$$\hat{f}(x_i, \theta_i) \approx f(x_i, \theta_i) = \frac{p_{\mathcal{H}_1}(x_i, \theta_i)}{p_{\mathcal{H}_0}(x_i, \theta_0) + p_{\mathcal{H}_1}(x_i, \theta_i)}$$

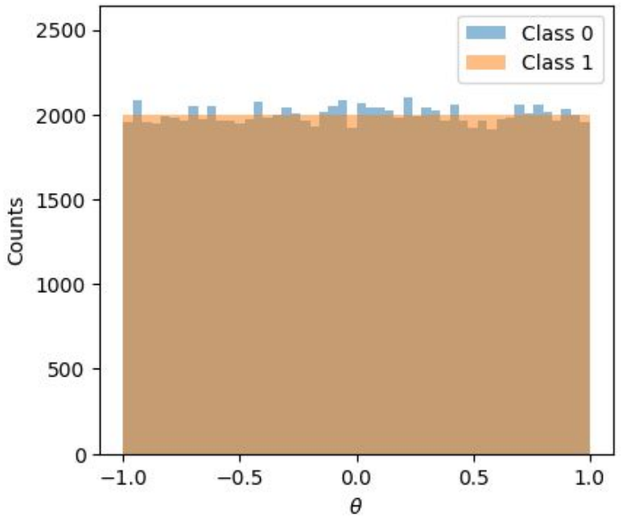
Likelihood-ratio trick

$$\boxed{\frac{\hat{f}(x_i, \theta_i)}{1 - \hat{f}(x_i, \theta_i)} \approx \frac{p_{\mathcal{H}_1}(x_i, \theta_i)}{p_{\mathcal{H}_0}(x_i, \theta_0)}}$$

Parameter estimation with classifiers

- **Can we use Machine Learning to estimate the conditional log-likelihood ratio test-statistic for inference?**
  - We have arrived at an estimator for the "joint" probability density: $$\frac{\hat{f}(x_i,\theta_i)}{1-\hat{f}(x_i,\theta_i)} \approx \frac{p_{\mathcal{H}_1}(x_i,\theta_i)}{p_{\mathcal{H}_0}(x_i,\theta_0)}$$
  - **Probability theory:** *joint = conditional x marginal*
    $$p(A,B) = p(A|B) \times p(B) \qquad p(B) = \int p(A,B)\,\mathrm{d}A$$
    $$\frac{\hat{f}(x_i,\theta_i)}{1-\hat{f}(x_i,\theta_i)} \approx \frac{p(x_i|\theta_i)p_{\mathcal{H}_1}(\theta_i)}{p(x_i|\theta_0)p_{\mathcal{H}_0}(\theta_0)}$$
  - Now comes the key training "trick" - align the marginal of $\mathcal{H}_0$ with $\{\mathcal{H}_1\}_\theta$
    - When training the network, rather than using $\theta = \theta_0$ for Class 0 samples, you must pair each $x_i^{\mathcal{H}_0}$ with a randomly sampled value of $\theta_i^{\mathcal{H}_0} \sim p_{\mathcal{H}_1}(\theta)$
    - Enforces $p(\theta)$ to be the same between Class 0 and Class 1
    - Interpret classifier output as the conditional density ratio

$$\frac{\hat{f}(x_i,\theta_i)}{1-\hat{f}(x_i,\theta_i)} \approx \frac{p(x_i|\theta_i)}{p(x_i|\theta_0)}$$



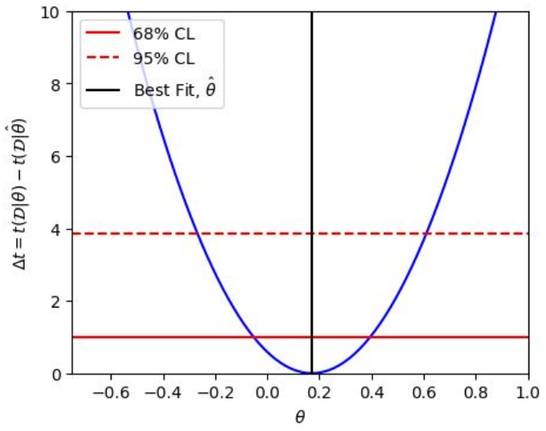
e.g. simple 1D parameter vector example

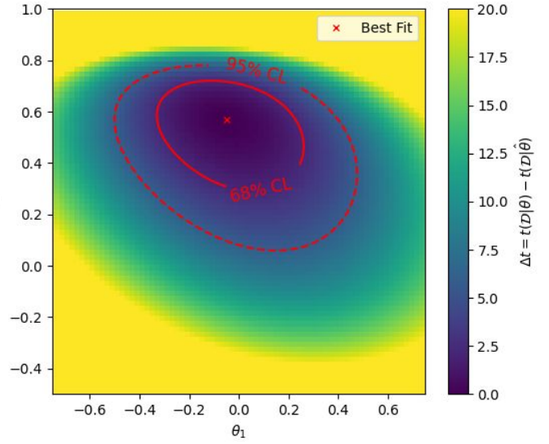
Parameter estimation with classifiers

- **Can we use Machine Learning to estimate the conditional log-likelihood ratio test-statistic for inference?**
  - We have arrived at an estimator for the "joint" probability density: $$\frac{\hat{f}(x_i,\theta_i)}{1-\hat{f}(x_i,\theta_i)} \approx \frac{p_{\mathcal{H}_1}(x_i,\theta_i)}{p_{\mathcal{H}_0}(x_i,\theta_0)}$$
  - **Probability theory:** *joint = conditional x marginal*
    $$p(A,B) = p(A|B) \times p(B) \qquad p(B) = \int p(A,B)\,\mathrm{d}A$$
  - $$\frac{\hat{f}(x_i,\theta_i)}{1-\hat{f}(x_i,\theta_i)} \approx \frac{p(x_i|\theta_i)\cancel{p_{\mathcal{H}_1}(\theta_i)}}{p(x_i|\theta_0)\cancel{p_{\mathcal{H}_0}(\theta_0)}}$$
  - Now comes the key training "trick" - align the marginal of $\mathcal{H}_0$ with $\{\mathcal{H}_1\}_\theta$
- **We can use the output of a parametric (binary) classifier trained with simulation to approximate the conditional test-statistic**

$$t(\mathcal{D}|\theta) = -2\sum_{x_i\in\mathcal{D}}^{N_{obs}} \ln\frac{p(x_i|\theta)}{p(x_i|\theta_0)} \approx -2\sum_{x_i\in\mathcal{D}}^{N_{obs}} \ln\frac{\hat{f}(x_i,\theta_i)}{1-\hat{f}(x_i,\theta_i)}$$

For a given dataset $\mathcal{D}$, we evaluate the classifier (and subsequently the approximate test-statistic) over θ parameter space





Use $\Delta t(\mathcal{D}|\theta)$ to infer the best-fit value and confidence intervals

Parameter estimation example

- Notebook: ADD LINK
- 2D parameter estimation task: in this example the analytic likelihood **is** known (**2D Gaussian**)
  - We will use a simulator to learn the test-statistic and then compare to the analytic solution
  - Note in most real-world scenarios, you will not have access to the analytic likelihood
- **Example:** infer parameters $\theta = (\mu_2, \rho_{12})$ and their confidence intervals from observed data (N_{obs}=20)
  - Two observables x = (`x1`, `x2`) are measured
  - Mean μ_1 and stdevs σ_1, σ_2 of the 2D Gaussian are already known (= 0, 1, 1)
  - Analytic probability density:

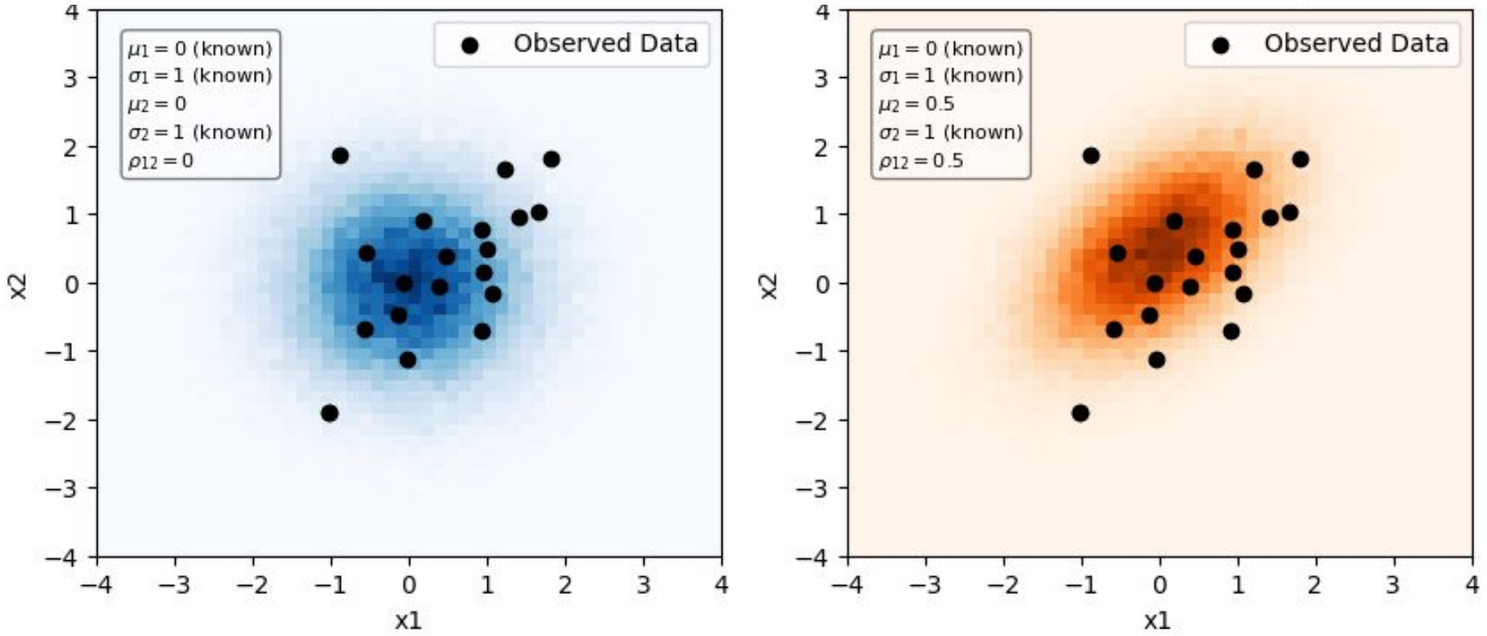
$$p(x|\mu_2, \rho_{12}) = \frac{1}{2\pi\sqrt{1-\rho_{12}^2}} \exp\left(-\frac{1}{2(1-\rho_{12}^2)}\left[x_1^2 + (x_2-\mu_2)^2 - 2\rho_{12}x_1(x_2-\mu_2)\right]\right)$$

  - We can simulate data for any values of $\theta = (\mu_2, \rho_{12})$: $x_i^{\text{sim}} \sim p(x|\mu_2, \rho_{12})$
  - We will use the simulation to approximate the conditional log-likelihood ratio test-statistic for inference

	x1	x2
0	-1.010586	-1.910270
1	1.808325	1.808219
2	-0.873411	1.869261
3	0.471487	0.392625
4	1.220651	1.658730
5	1.411832	0.942164
6	0.185869	0.893841
7	0.949642	0.160108
8	1.068799	-0.175526
9	0.933053	0.773940
10	1.666441	1.042463
11	0.922885	-0.706301
12	-0.058121	-0.011644
13	-0.539066	0.443757
14	-1.013156	-1.897984
15	0.998487	0.492797
16	0.392428	-0.049023
17	-0.131662	-0.484124
18	-0.032107	-1.135443
19	-0.572507	-0.682939

Parameter estimation example

- Notebook: ADD LINK
- 2D parameter estimation task: in this example the analytic likelihood **is** known (**2D Gaussian**)
  - We will use a simulator to learn the test-statistic and then compare to the analytic solution
  - Note in most real-world scenarios, you will not have access to the analytic likelihood
- **Example:** infer parameters $\theta = (\mu_2, \rho_{12})$ and their confidence intervals from observed data (N_{obs}=20)
  - e.g. comparing simulation to observed data for two points in $\theta = (\mu_2, \rho_{12})$ space...



	x1	x2
0	-1.010586	-1.910270
1	1.808325	1.808219
2	-0.873411	1.869261
3	0.471487	0.392625
4	1.220651	1.658730
5	1.411832	0.942164
6	0.185869	0.893841
7	0.949642	0.160108
8	1.068799	-0.175526
9	0.933053	0.773940
10	1.666441	1.042463
11	0.922885	-0.706301
12	-0.058121	-0.011644
13	-0.539066	0.443757
14	-1.013156	-1.897984
15	0.998487	0.492797
16	0.392428	-0.049023
17	-0.131662	-0.484124
18	-0.032107	-1.135443
19	-0.572507	-0.682939

Analytic solution

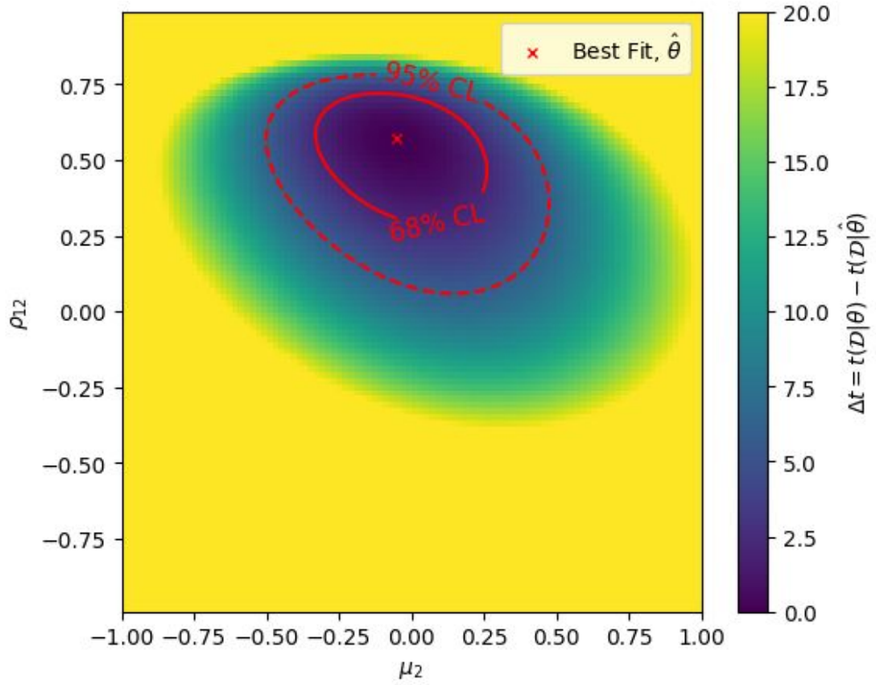
- Analytic probability density: $$p(x|\mu_2,\rho_{12}) = \frac{1}{2\pi\sqrt{1-\rho_{12}^2}} \exp\left(-\frac{1}{2(1-\rho_{12}^2)}\left[x_1^2 + (x_2-\mu_2)^2 - 2\rho_{12}x_1(x_2-\mu_2)\right]\right)$$

- Define test-statistic with respect to some reference hypothesis: $\theta_0 = (0,0)$

$$t(\mathcal{D}|\mu_2,\rho_{12}) = -2\sum_{x_i\in\mathcal{D}}^{N_{obs}} \ln\frac{p(x_i|\mu_2,\rho_{12})}{p(x_i|0,0)} \longrightarrow \Delta t(\mathcal{D}|\mu_2,\rho_{12}) = t(\mathcal{D}|\mu_2,\rho_{12}) - t(\mathcal{D}|\hat{\mu}_2,\hat{\rho}_{12})$$

	x1	x2
0	-1.010586	-1.910270
1	1.808325	1.808219
2	-0.873411	1.869261
3	0.471487	0.392625
4	1.220651	1.658730
5	1.411832	0.942164
6	0.185869	0.893841
7	0.949642	0.160108
8	1.068799	-0.175526
9	0.933053	0.773940
10	1.666441	1.042463
11	0.922885	-0.706301
12	-0.058121	-0.011644
13	-0.539066	0.443757
14	-1.013156	-1.897984
15	0.998487	0.492797
16	0.392428	-0.049023
17	-0.131662	-0.484124
18	-0.032107	-1.135443
19	-0.572507	-0.682939

- Scan over $\theta = (\mu_2, \rho_{12})$ parameter space and calculate $\Delta t(\mathcal{D}|\mu_2,\rho_{12})$



Use Wilks' Theorem to estimate confidence intervals:

- 68% CL is region defined by $\Delta t < 2.30$
- 95% CL is region defined by $\Delta t < 5.99$

Neyman-Pearson lemma tells us this is the most powerful test-statistic

Can we approximate the test-statistic using our SBI methods?

Parametric classifier training

1) Simulate training data

```
sim_H0 = run_simulation(num_train_per_class, mu1=0, sigma1=1, mu2=0, sigma2=1, rho12=0)
```

- Class 0: simulate N samples from reference $\theta_0 = (0, 0)$
- Class 1: simulate N samples with different $\theta = (\mu_2, \rho_{12})$ values
  - Define grid of points in $\theta = (\mu_2, \rho_{12})$ space (50x50 *over sensible range*)
  - Simulate N/2500 samples at each point and concatenate

```
N = 100000
sim_H1_ensemble = []
mu2_train_vals = np.linspace(-1,1,50)
rho12_train_vals = np.linspace(-0.9999,0.9999,50)

# Calculate the number of training samples per parameter point to keep total = num_train_per_class
num_train_per_subsample = N // (len(mu2_train_vals) * len(rho12_train_vals))
for mu2 in mu2_train_vals:
    for rho12 in rho12_train_vals:
        sim_subsample = run_simulation(num_train_per_subsample, mu1=0, sigma1=1, mu2=mu2, sigma2=1, rho12=rho12)
        sim_subsample['mu2'] = mu2
        sim_subsample['rho12'] = rho12
        sim_H1_ensemble.append(sim_subsample)

# Concatenate all subsamples into a single dataframe and add label
sim_H1 = pd.concat(sim_H1_ensemble, ignore_index=True)
sim_H1['label'] = 1
```

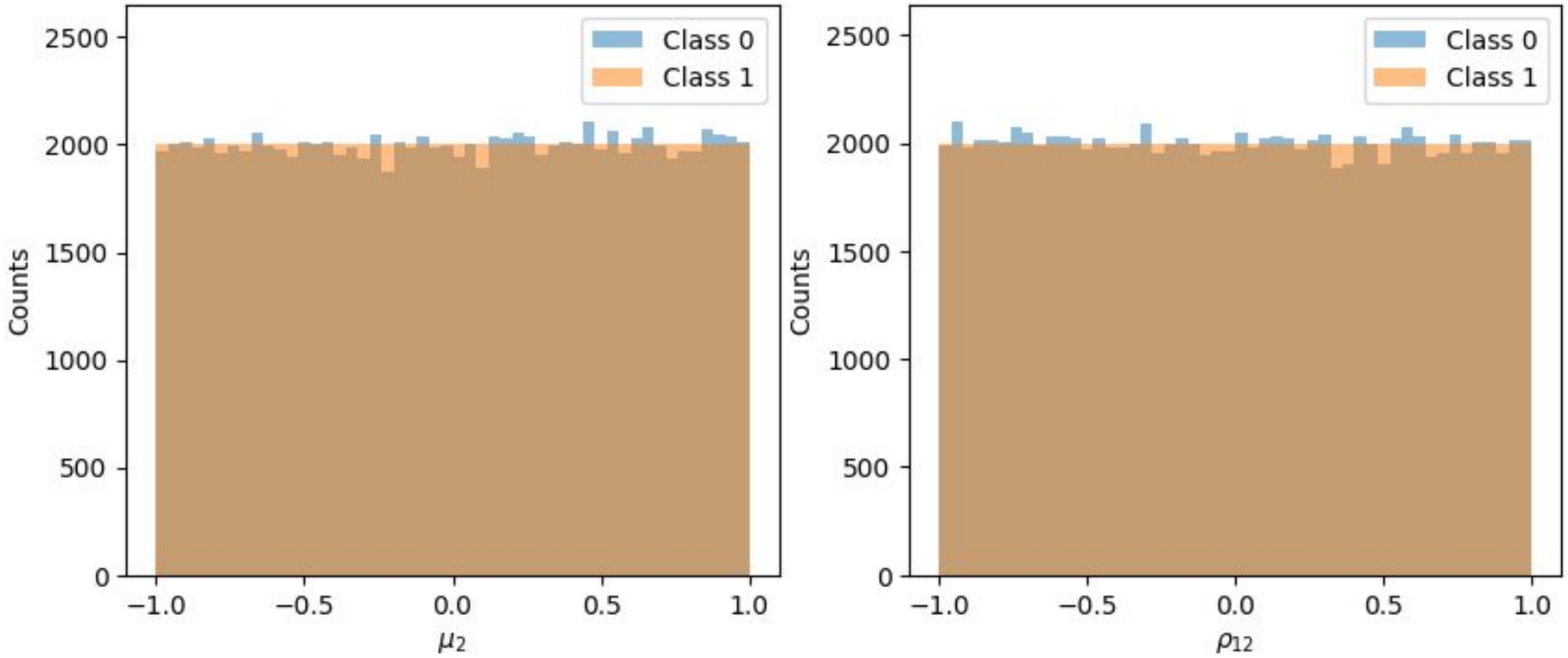
Parametric classifier training

1) Simulate training data

2) Align marginal distribution of Class 0 with Class 1 i.e. pair each $x_i^{\mathcal{H}_0}$ with randomly sampled $\mu_2^{\mathcal{H}_0}, \rho_{12}^{\mathcal{H}_0} \sim p_{\mathcal{H}_1}(\mu_2, \rho_{12})$

```
sim_H0['mu2'] = np.random.choice(mu2_train_vals, size=len(sim_H0))
sim_H0['rho12'] = np.random.choice(rho12_train_vals, size=len(sim_H0))

# Add label for H0
sim_H0['label'] = 0
```

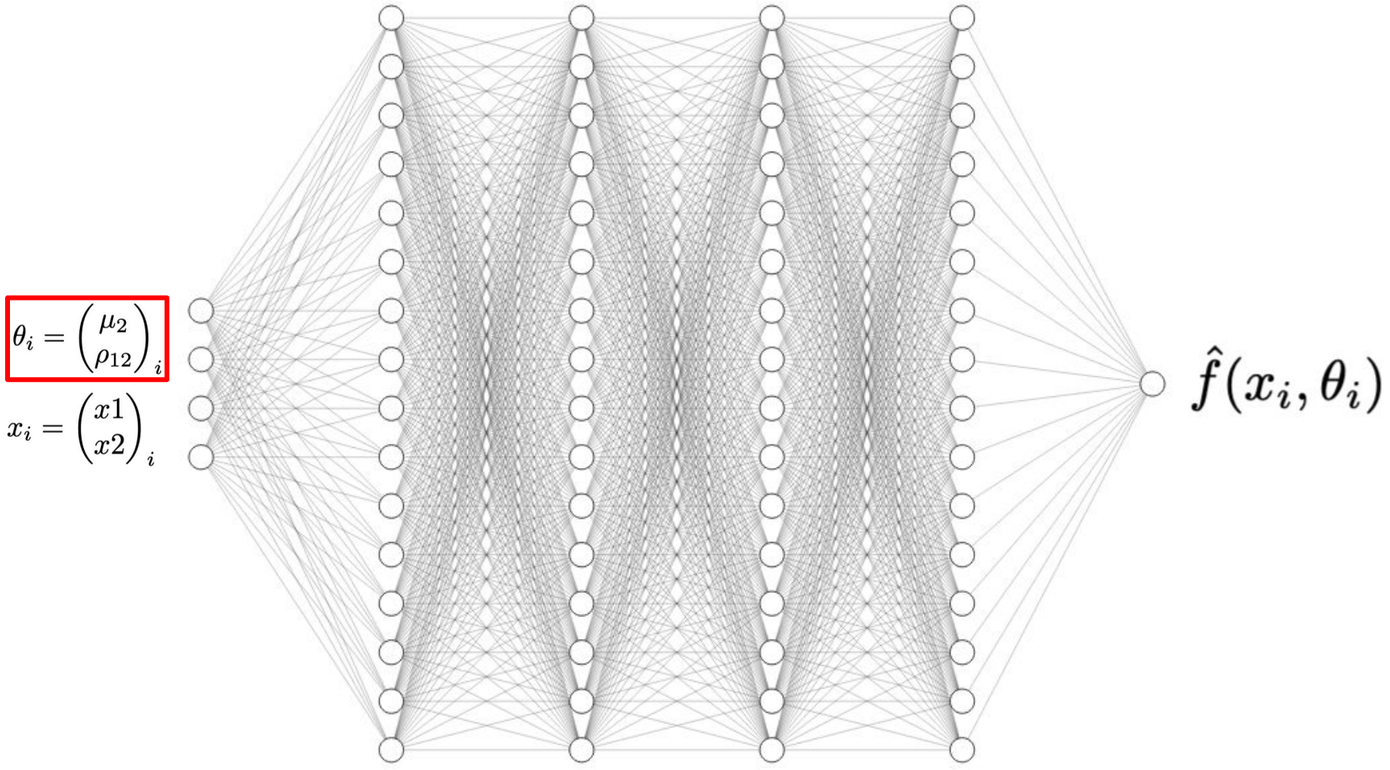


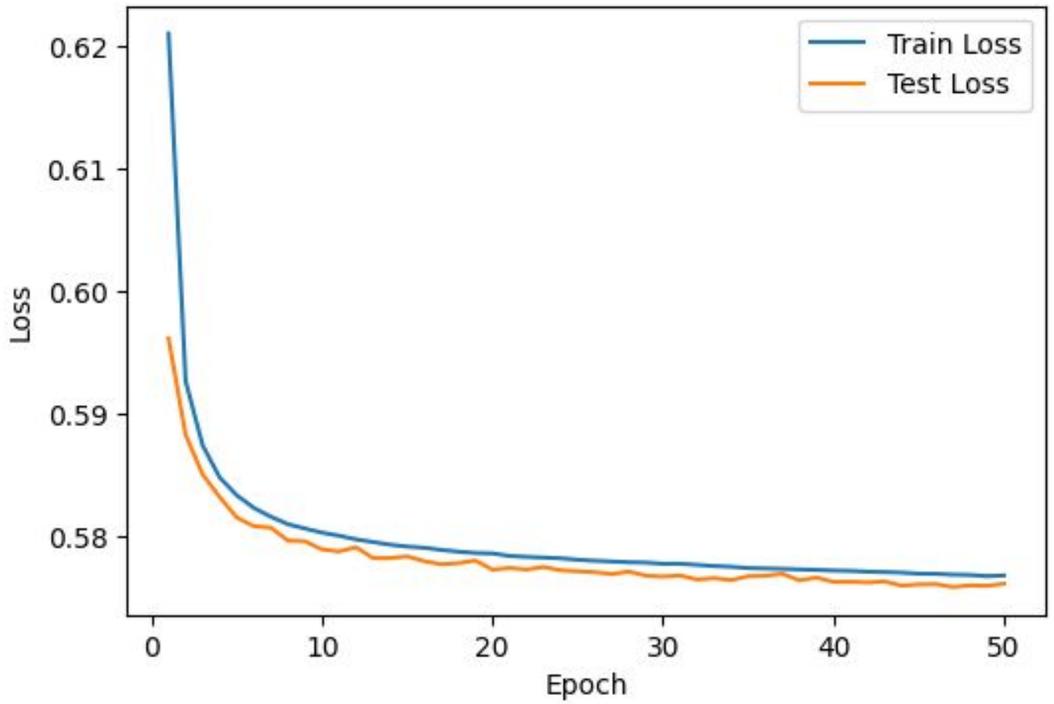
Parametric classifier training

1) Simulate training data

2) Align marginal distribution of Class 0 with Class 1 i.e. pair each $x_i^{\mathcal{H}_0}$ with randomly sampled $\mu_2^{\mathcal{H}_0}, \rho_{12}^{\mathcal{H}_0} \sim p_{\mathcal{H}_1}(\mu_2, \rho_{12})$

3) Train parametric classifier with **`torch.nn`** to classify between Class 0 (reference) and Class 1 (ensemble)



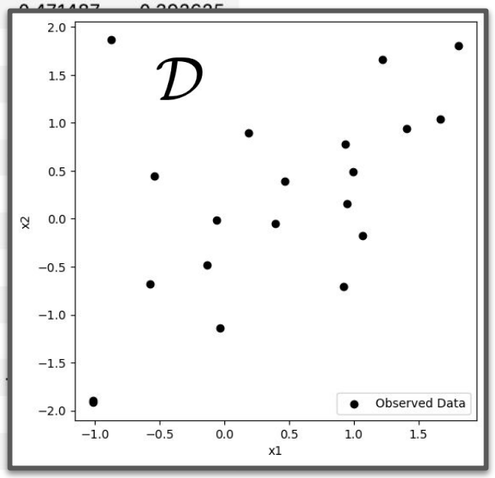


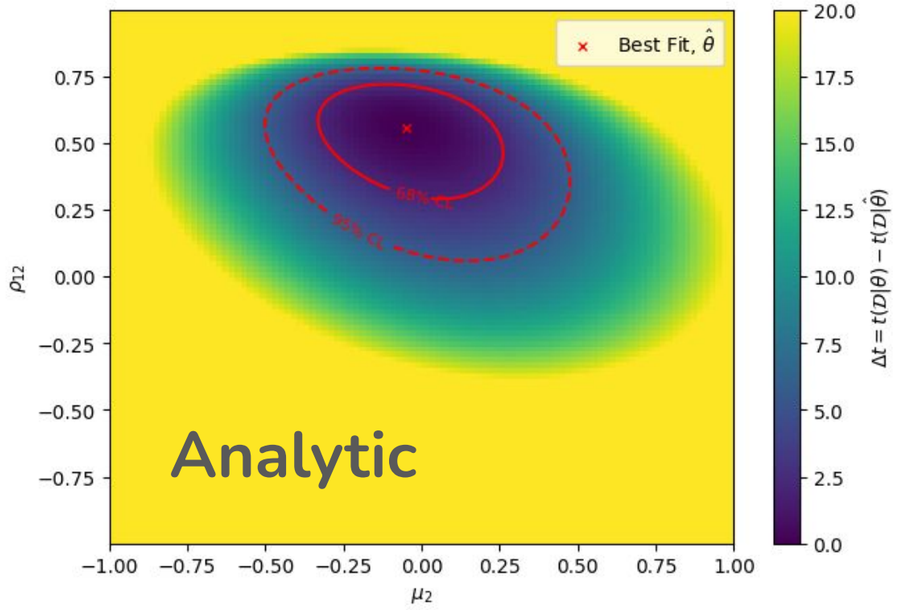
$$\mathcal{L}[f] = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln f(x_i, \theta_i) + (1 - y_i) \ln \left(1 - f(x_i, \theta_i)\right)$$

Parameter estimation with classifier

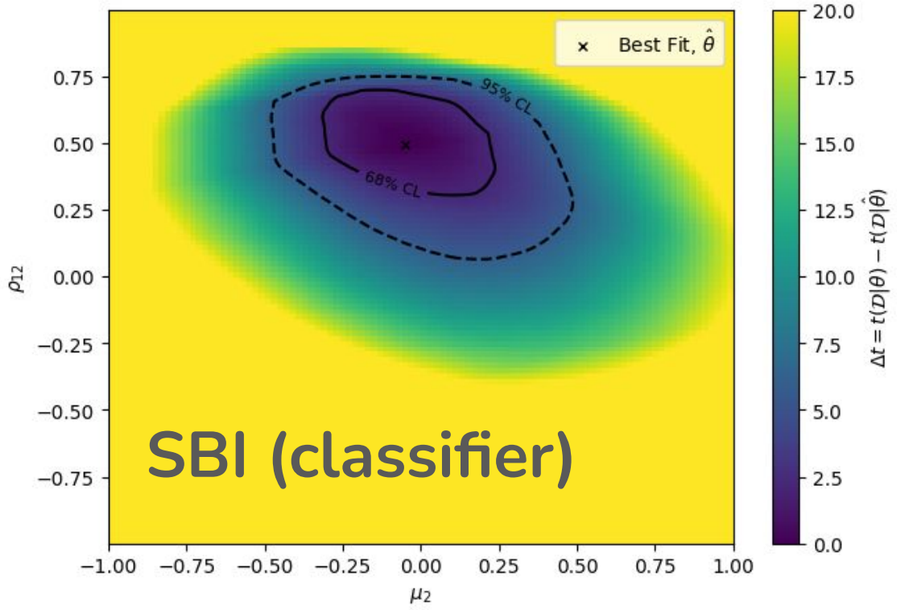
1) Simulate training data
2) Align marginal distribution of Class 0 with Class 1 i.e. pair each $x_i^{\mathcal{H}_0}$ with randomly sampled $\mu_2^{\mathcal{H}_0}, \rho_{12}^{\mathcal{H}_0} \sim p_{\mathcal{H}_1}(\mu_2, \rho_{12})$
3) Train parametric classifier with **`torch.nn`** to classify between Class 0 (reference) and Class 1 (ensemble)
4) Evaluate classifier for observed data $\mathcal{D}$ over $\theta = (\mu_2, \rho_{12})$ parameter space → calculate learned test-statistic

	x1	x2
0	-1.010586	-1.910270
1	1.808325	1.808219
2	-0.873411	1.869261
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16	0.392428	-0.049023
17	-0.131662	-0.484124
18	-0.032107	-1.135443
19	-0.572507	-0.682939





$$t(\mathcal{D}|\mu_2, \rho_{12}) = -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \frac{p(x_i|\mu_2, \rho_{12})}{p(x_i|0,0)}$$

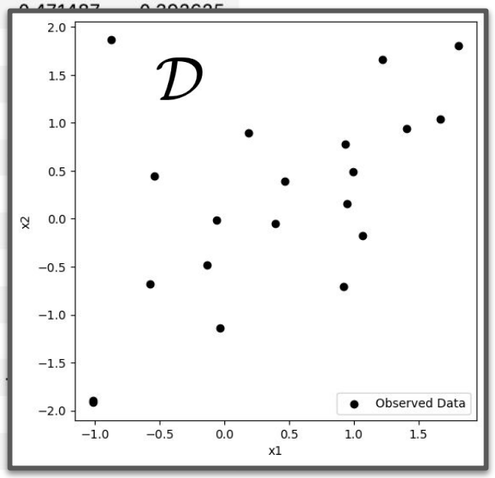


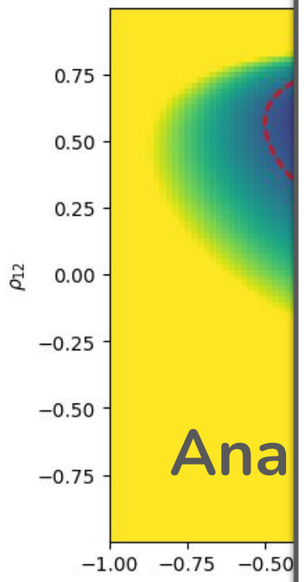
$$t(\mathcal{D}|\mu_2, \rho_{12}) = -2 \sum_{x_i \in \mathcal{D}}^{N_{obs}} \ln \frac{\hat{f}(x_i, \mu_2, \rho_{12})}{1 - \hat{f}(x_i, \mu_2, \rho_{12})}$$

Parameter estimation with classifier

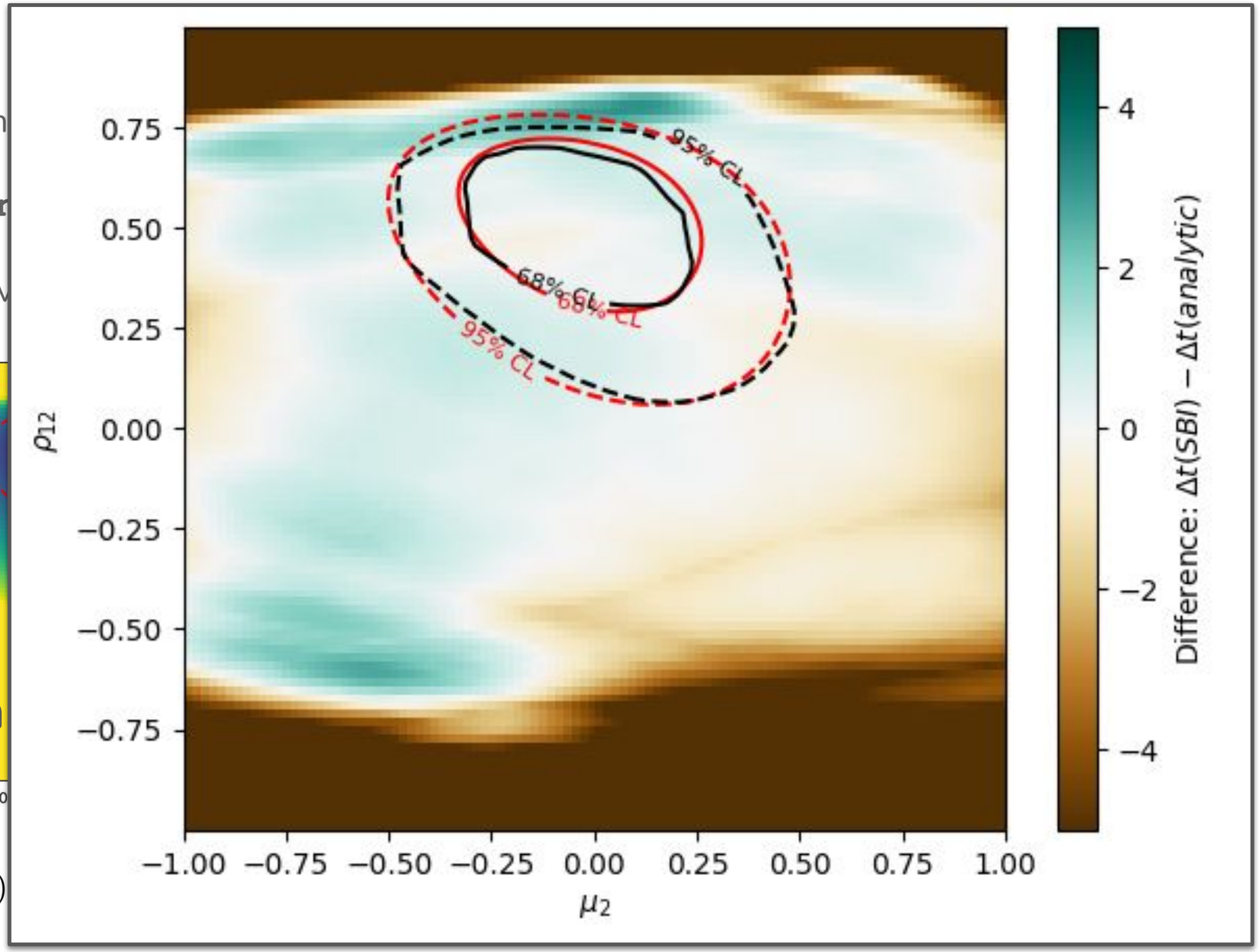
1) Simulate training data
2) Align marginal distribution of Class 0 with
3) Train parametric classifier with **`torch.nn`**
4) Evaluate classifier for observed data $\mathcal{D}$ ov

	x1	x2
0	-1.010586	-1.910270
1	1.808325	1.808219
2	-0.873411	1.869261
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16	0.392428	-0.049023
17	-0.131662	-0.484124
18	-0.032107	-1.135443
19	-0.572507	-0.682939

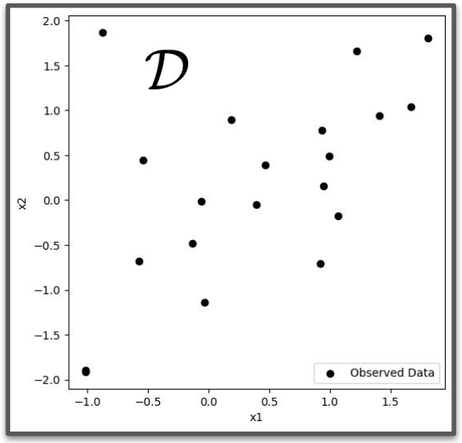




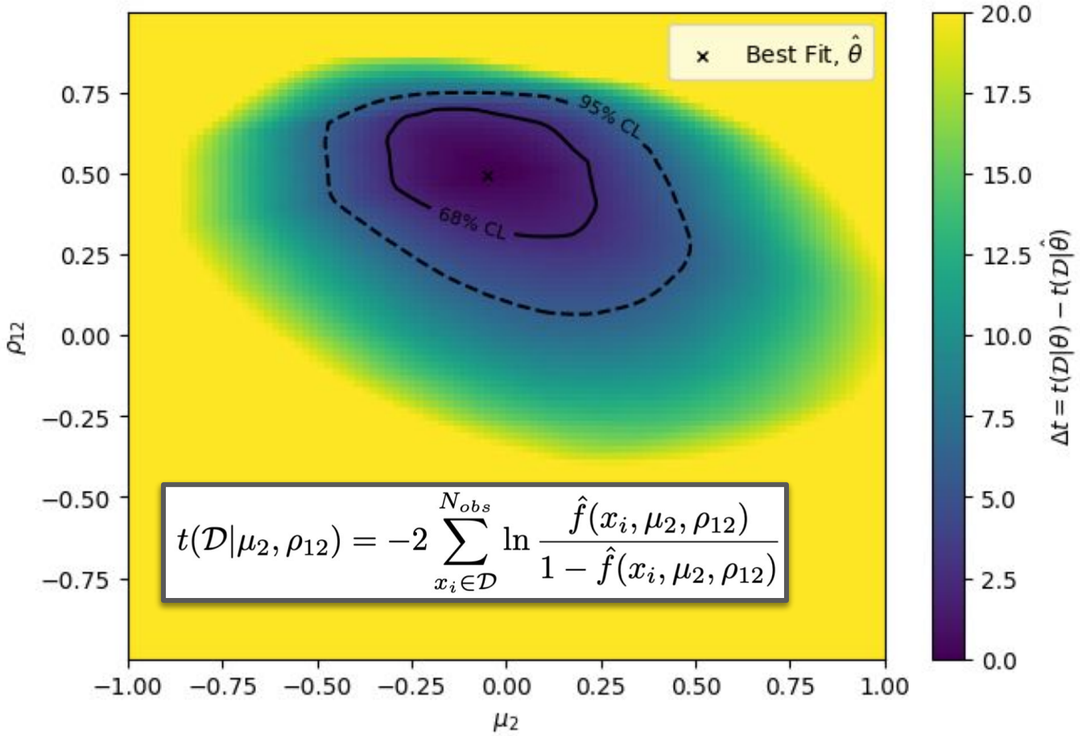
$t(\mathcal{D}|\mu_2, \rho_{12})$

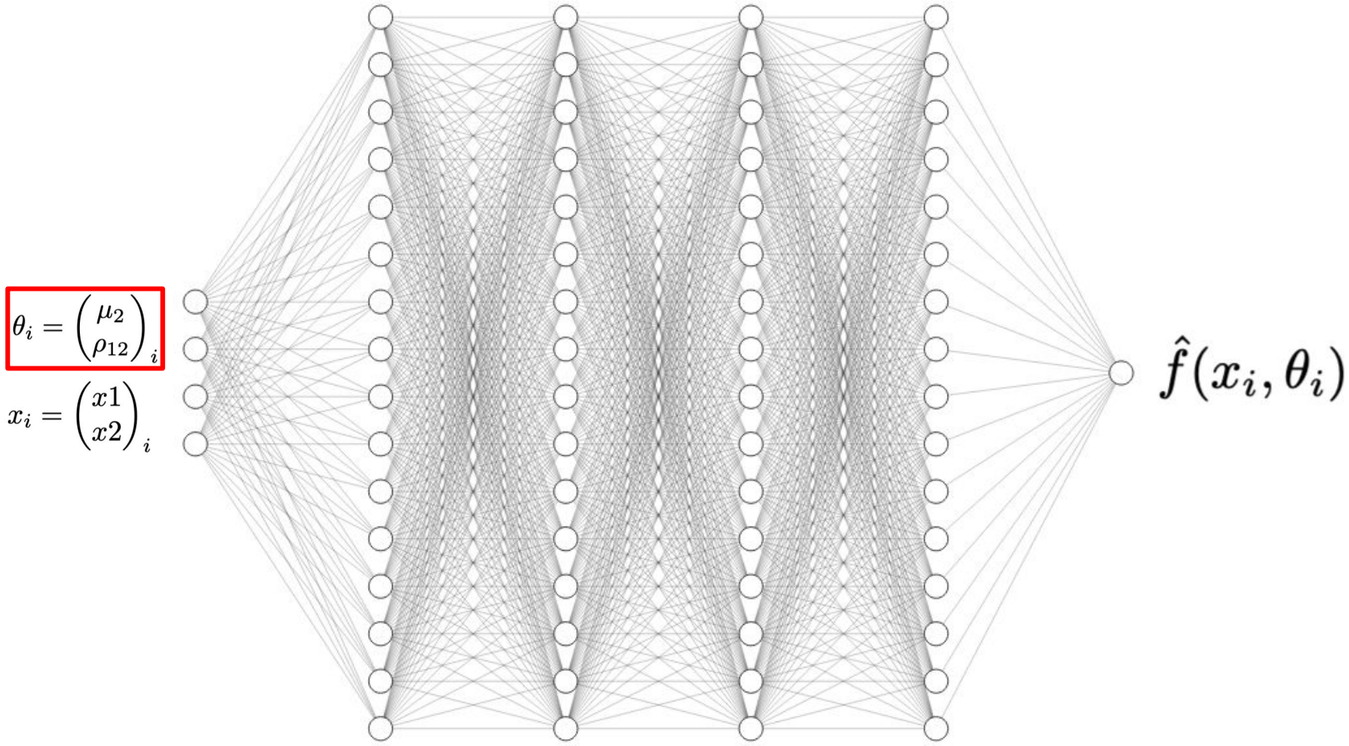


Summary: parameter estimation

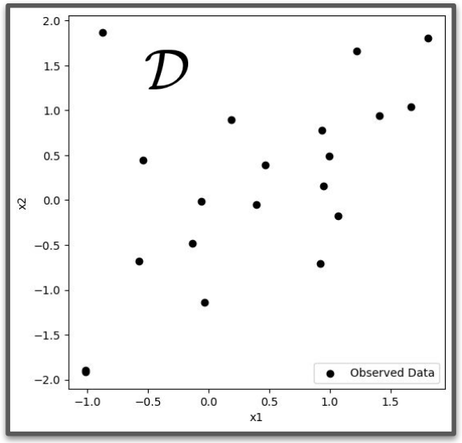


- We were presented with a research problem to estimate two parameters for a dataset (N_{obs} = 20)
- Our faithful simulator could generate data for different parameter values: $x_i, \theta_i \sim p(x|\theta)$
- We trained a binary **parametric classifier** to distinguish Class 0 (reference) from Class 1 (ensemble)
- Use output of classifier to approximate **<u>conditional</u> log-likelihood-ratio test-statistic**

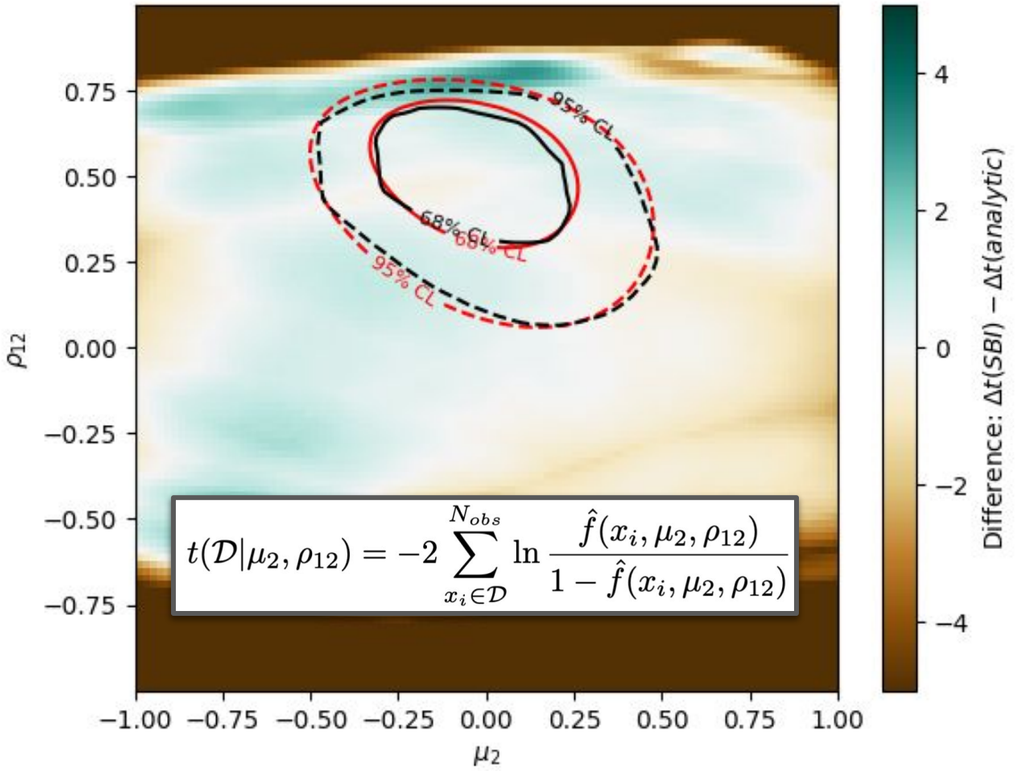




Summary: parameter estimation



- We were presented with a research problem to estimate two parameters for a dataset (N_{obs} = 20)
- Our faithful simulator could generate data for different parameter values: $x_i, \theta_i \sim p(x|\theta)$
- We trained a binary **parametric classifier** to distinguish Class 0 (reference) from Class 1 (ensemble)
- Use output of classifier to approximate **<u>conditional</u> log-likelihood-ratio test-statistic**



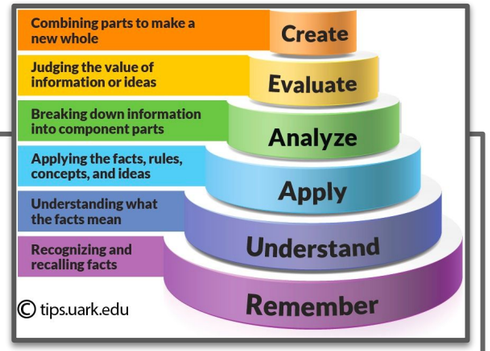
- We were able to accurately reproduce the analytic solution!
  - Use Wilk's theorem to infer the confidence intervals
  - Scope for improvement: slight bias in the estimator
    - Increasing amount of training data?
    - Increasing granularity of $\theta = (\mu_2, \rho_{12})$ training grid?
    - More complex classifier architecture?
- Inference is amortized (same classifier for new data)
- End-to-end example of using a ML classifier for SBI (parameter est.)
  - **Extends to real-world problems with no analytic solution!**

Further reading

- **General overview of SBI with ML:** [arXiv:1911.01429]

- **What if we do not have a "faithful" simulator?**
  - i.e. systematic differences between observed data and synthetic (simulated) data
  - Leads to systematic uncertainties in the measurements
  - Incorporate "nuisance parameters" into the classifier training → profile/marginalize in the inference step
  - [arXiv:1506.02169], [arXiv:2105.08742]
- **Validating and calibrating the classifier**
  - Sub-optimal training may lead to a biased estimator for the likelihood ratio, which could (in principle) lead to wrong conclusions!
  - Crucial to perform diagnostic checks on SBI model to validate output is what we expect
  - Without the analytic solution we must use (independent) simulation samples for validation
  - [arXiv:2412.01600] (Section 4)
- **Learning the likelihood**
  - Using ML classifiers is just one (simple) approach for SBI, where we learn the likelihood-ratio
  - Modern ML techniques can be used to learn the likelihood directly e.g. conditional normalising flows [JMLR 22 (2021)]
- **Bayesian approach to SBI: neural posterior estimation**
  - This lecture presented SBI in a frequentist paradigm but of course this extends to Bayesian inference
  - Many applications look at learning the posterior directly
  - [arXiv:2011.05991], [arXiv:2106.12594]

Lecture summary

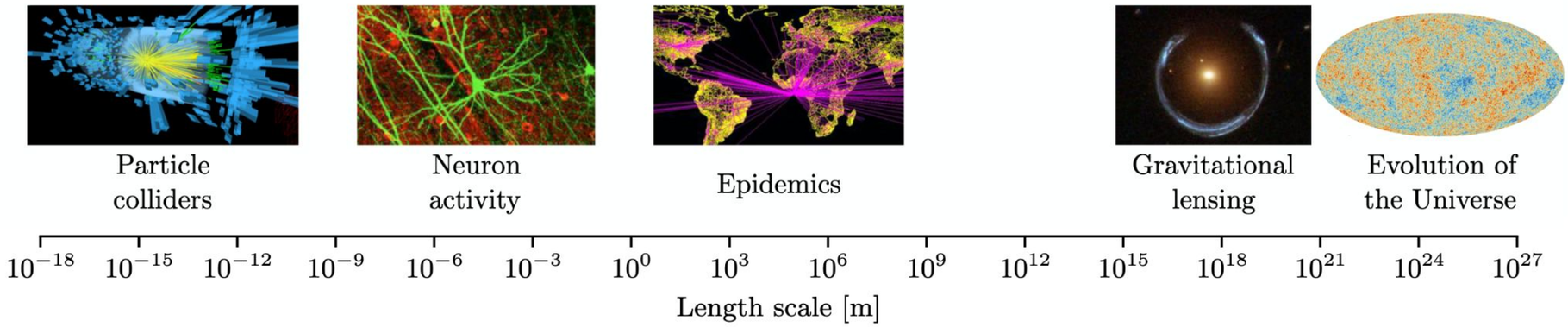


Intended Learning Outcomes

- **Understand** the need for simulation-based inference in the modern era of science
- **Understand** how to use a simple ML classifier to learn the likelihood-ratio and **apply** this knowledge to perform a hypothesis test on a research problem with an unknown likelihood
- Extend the approach to learning the conditional likelihood-ratio via a parametric classifier and **apply** this to a parameter estimation problem. **Evaluate** the performance by comparing to the analytic solution

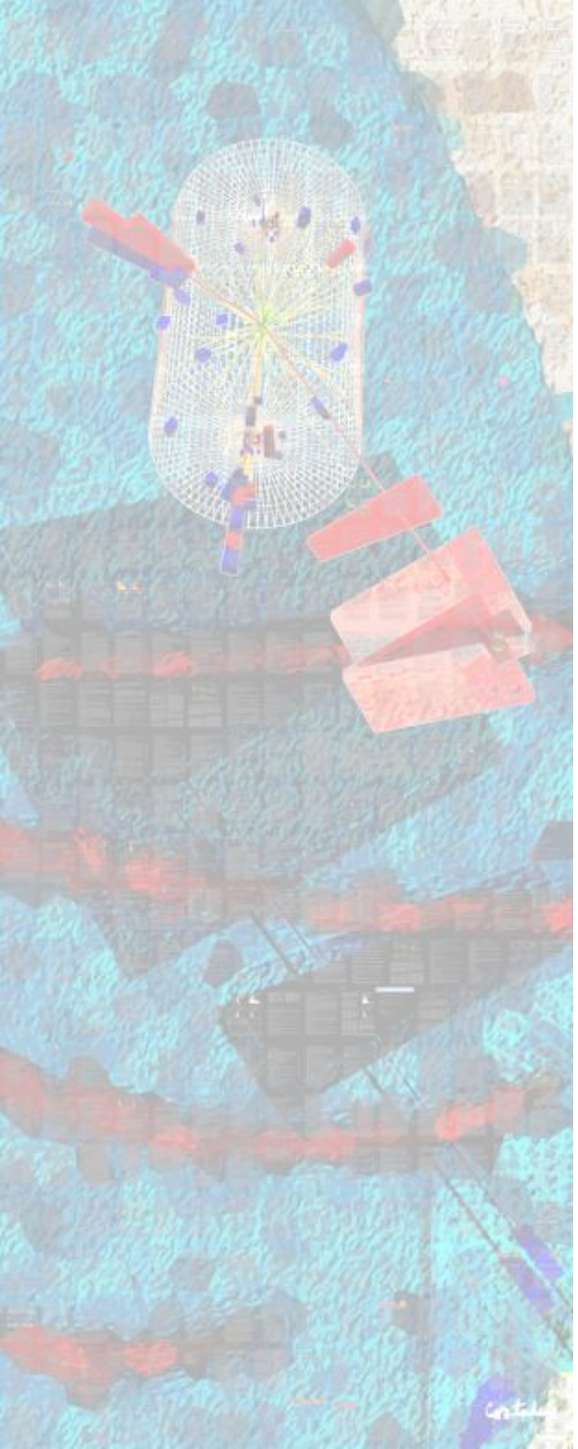
We hope this lecture has helped bridge the gap between ML and some core concepts in statistics!

Remember, these techniques generalise to complex data → Use ML classifiers to perform SBI for more interesting problems!



The project

- Add details
- X’ → lnu example
- Hypothesis test: spin of particle X’
- Parameter estimation: mass of X’



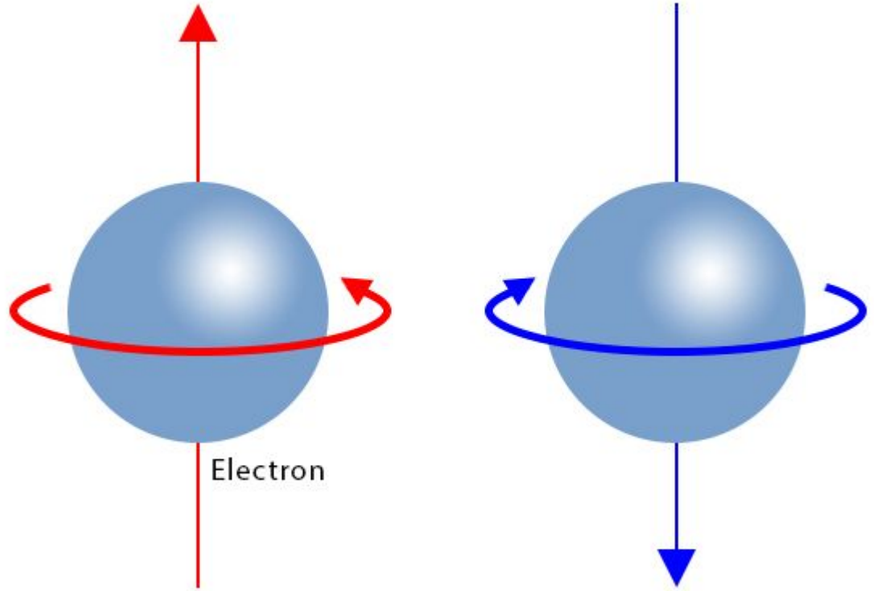
Back-Up

Optimal decision function for BCE loss

- Add maths

What is a particle's spin?

- Spin is a purely quantum mechanical property of an elementary particle
- It is an intrinsic form of angular momentum
  - Unlike everyday (classical) spinning objects
  - Does not correspond to any literal rotation in space



- Instead quantum spin influences how a particle behaves, decays, and how it interacts with forces
- This makes is possible to distinguish between different types of particles by their spin values
  - Bosons - integer spins
  - Fermions - spin-½ particles
  - These have very different properties...