Improving inference with matrix elements and machine learning

Johann Brehmer

New York University

IAS Program on High Energy Physics
HK, January 11, 2019

NYU

MSDSE
Moore - Sloan
Data Science Environments

Kyle Cranmer

Gilles Louppe

Juan Pavez

Felix Kling

Irina Espejo

Zubair Bhatti

Markus Stoye

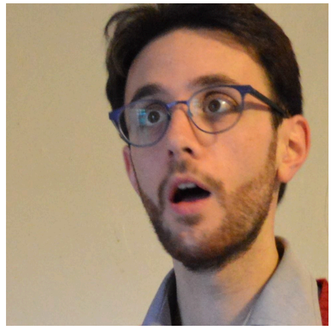
Tilman Plehn

Sally Dawson

Sam Homiller

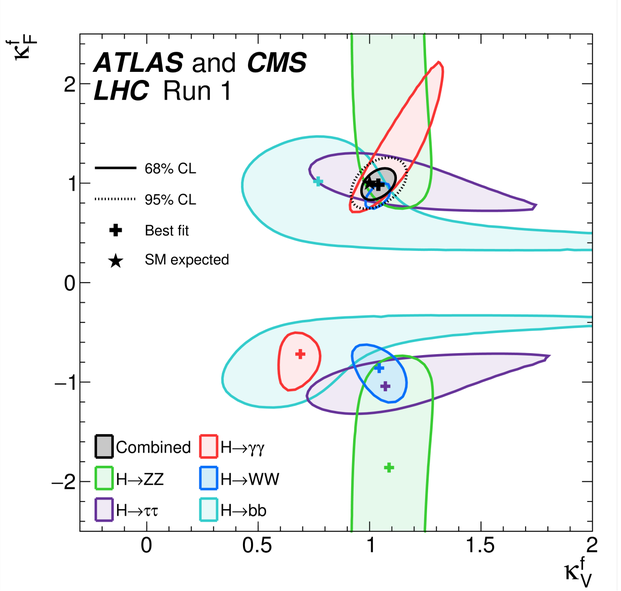
Josh Rudermann

Duccio Pappadopulo



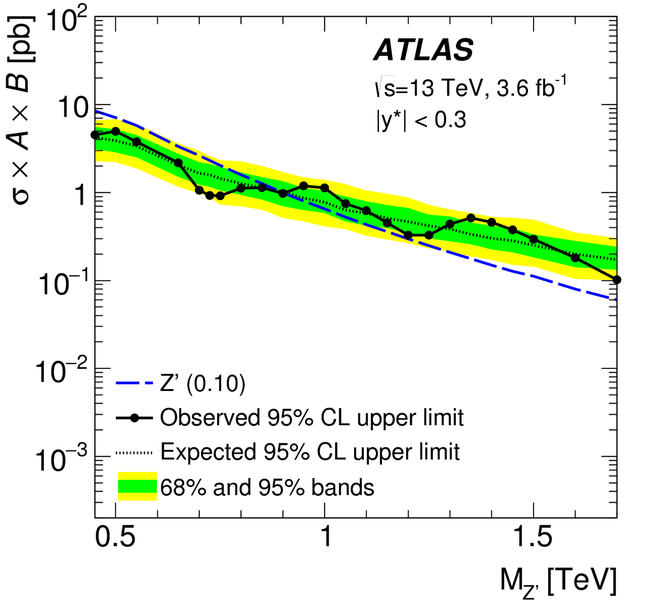
Marco Farina

Inference in particle physics



[ATLAS, CMS 1606.02266]

Measurements: Maximum likelihood estimates + confidence intervals

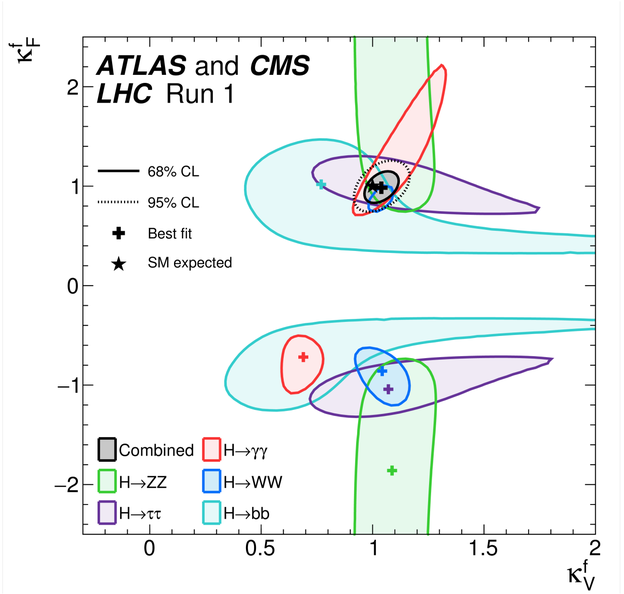


[CMS 1804.03496]

Searches: Hypothesis tests

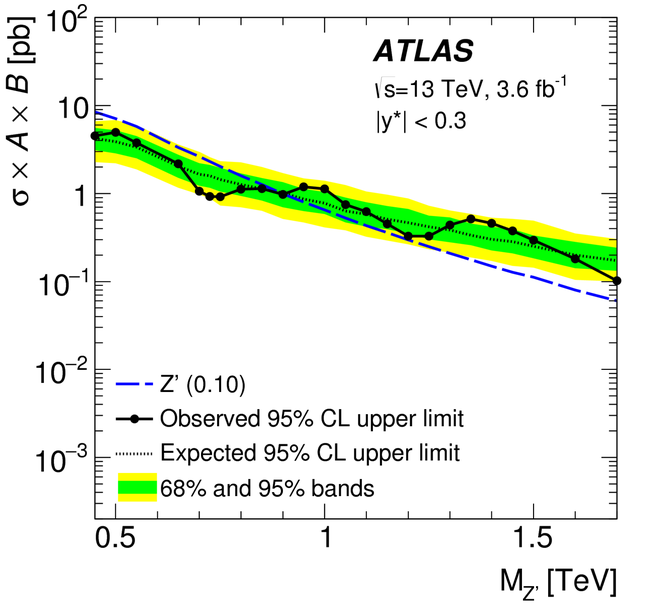
… typically with likelihood-based methods.

Inference in particle physics



[ATLAS, CMS 1606.02266]

Measurements: Maximum likelihood estimates + confidence intervals



[CMS 1804.03496]

Searches: Hypothesis tests

… typically with likelihood-based methods.

But surprisingly, when we want to use high-dimensional observations and have to deal with the detector response, we do not have a good way to calculate the likelihood.

This talk: With matrix elements and machine learning, we can estimate any likelihood and improve inference!

Likelihood-free inference

Particle physics processes

Theory
parameters
θ

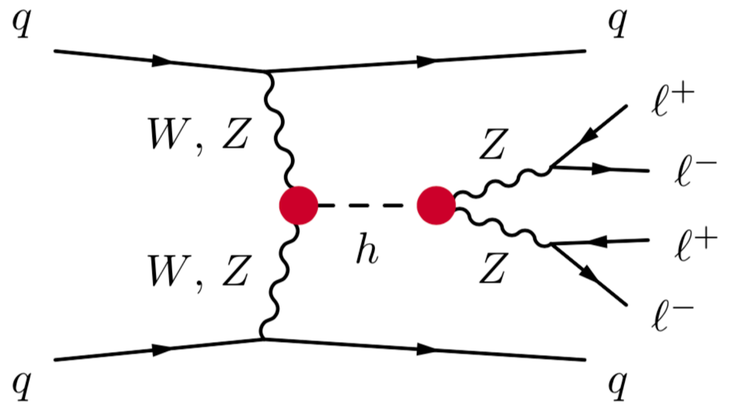
Evolution

Particle physics processes

Latent variables

Parton-level momenta z_p

Theory parameters θ

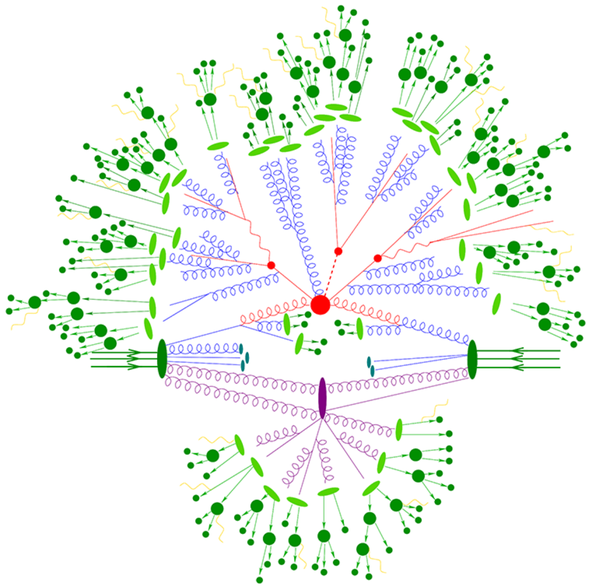


Evolution

Particle physics processes

Latent variables

Shower splittings z_s ← Parton-level momenta z_p ← Theory parameters θ



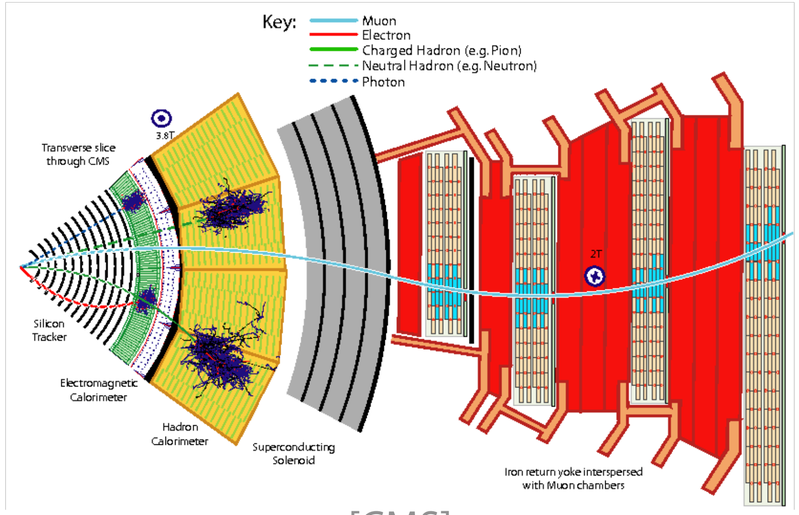
[F. Krauss]

Evolution

Particle physics processes

Latent variables

Detector interactions	Shower splittings	Parton-level momenta	Theory parameters
z_d	z_s	z_p	θ



[CMS]

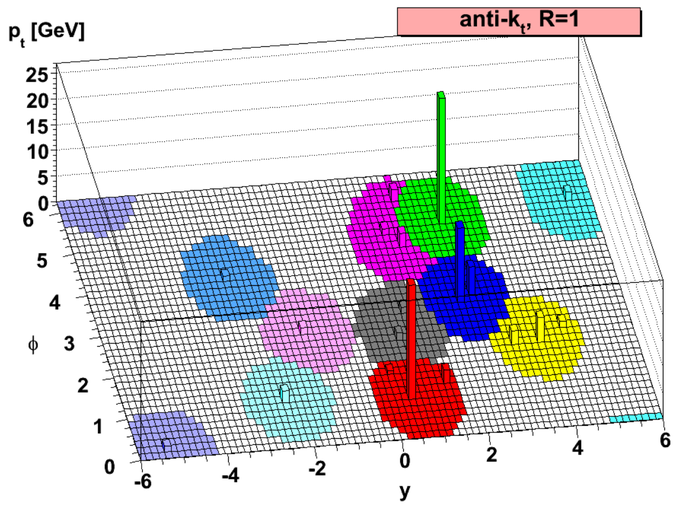
Evolution

Particle physics processes

Latent variables

Observables | Detector interactions | Shower splittings | Parton-level momenta | Theory parameters

$x \leftarrow z_d \leftarrow z_s \leftarrow z_p \leftarrow \theta$



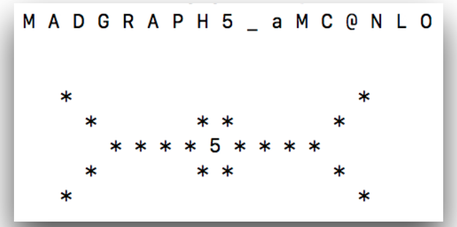
[M. Cacciari, G. Salam, G. Soyez 0802.1189]

Evolution

Particle physics processes

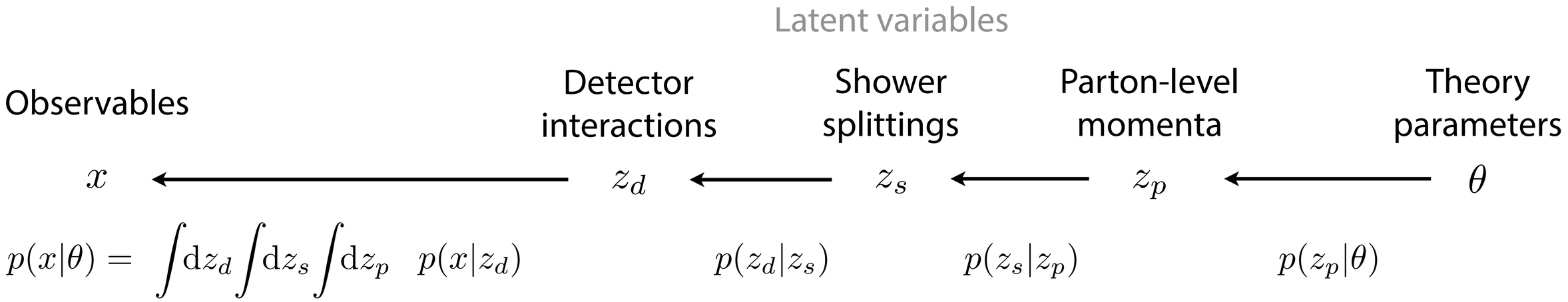
Latent variables

Observables	Detector interactions	Shower splittings	Parton-level momenta	Theory parameters
x	z_d	z_s	z_p	θ
Sample from	$p(x\|z_d)$	$p(z_d\|z_s)$	$p(z_s\|z_p)$	$p(z_p\|\theta)$



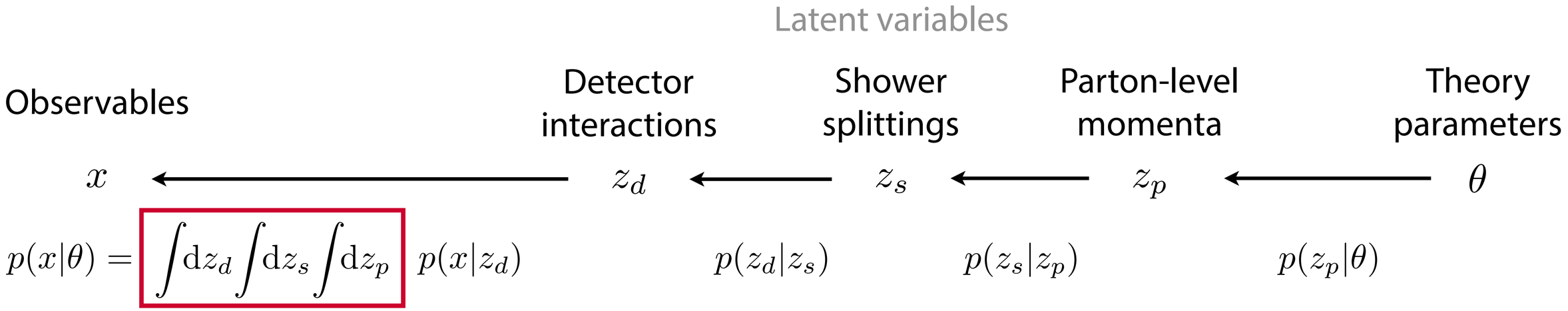
Prediction (simulation)

Particle physics processes



Inference

Particle physics processes

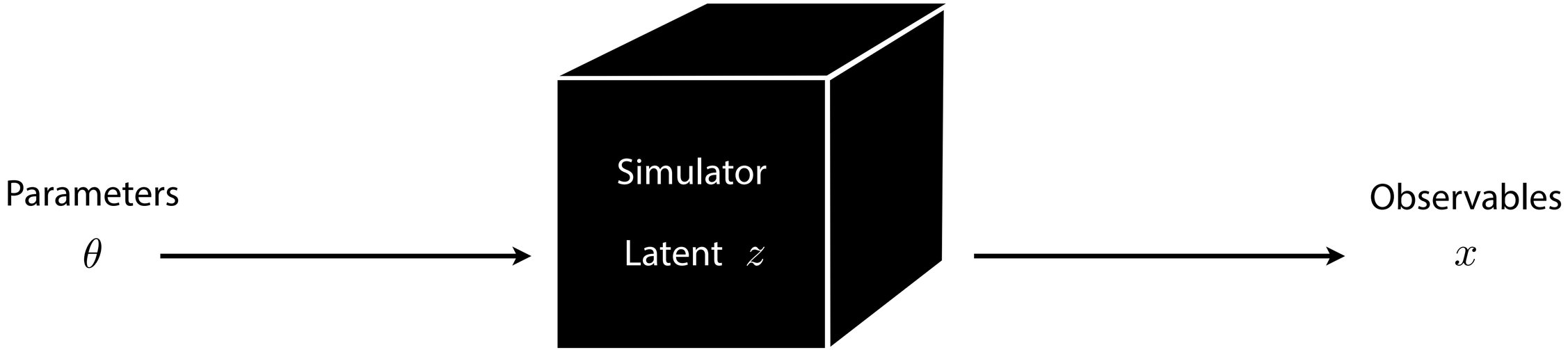


It's infeasible to calculate the integral over this enormous space!

(More subtle: We cannot sample from $p(z|x,\theta)$ efficiently.)

Inference

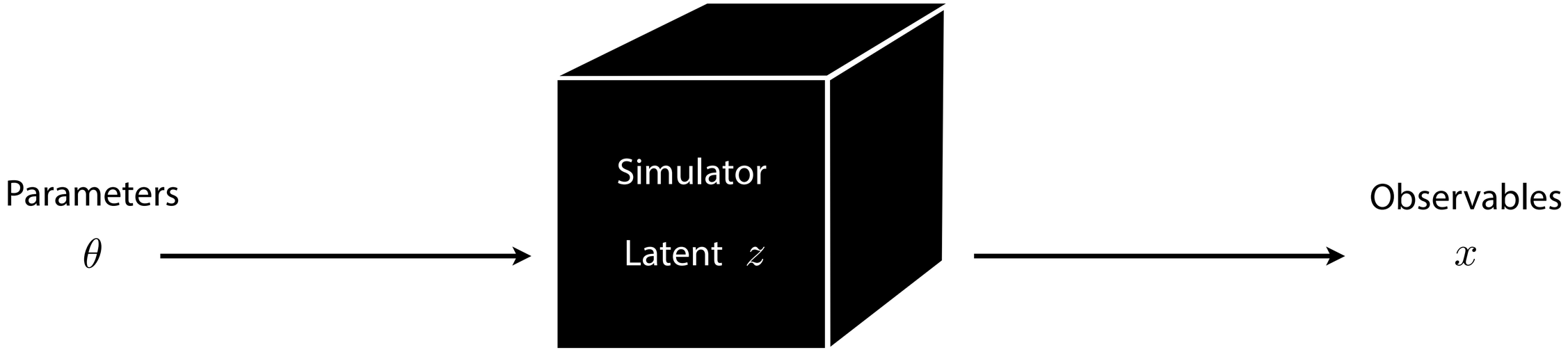
“Likelihood-free inference”



Prediction:

- Well-understood mechanistic model
- Simulator can generate samples

“Likelihood-free inference”



Prediction:
- Well-understood mechanistic model
- Simulator can generate samples

Inference:
- Likelihood function $p(x|\theta)$ is intractable
- Inference needs estimator $\hat{p}(x|\theta)$

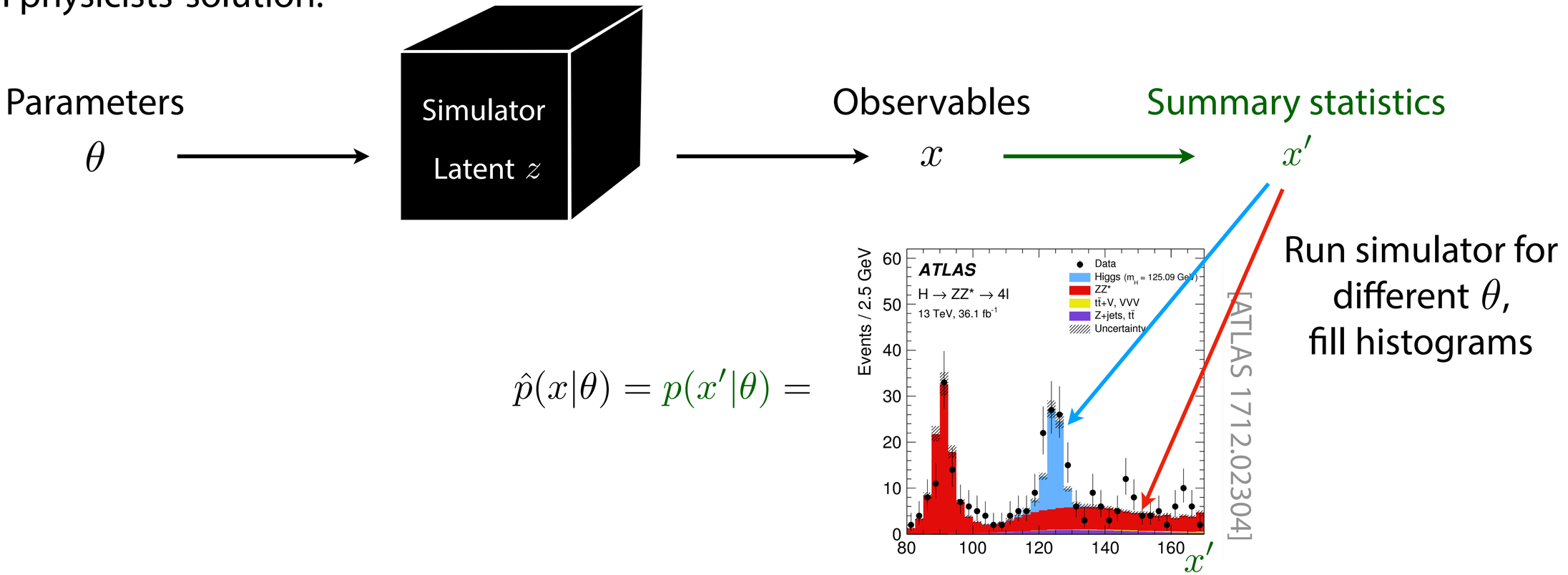
Why has that not stopped us so far?

Solve it by histogramming summary statistics

- Typical physicists' solution:

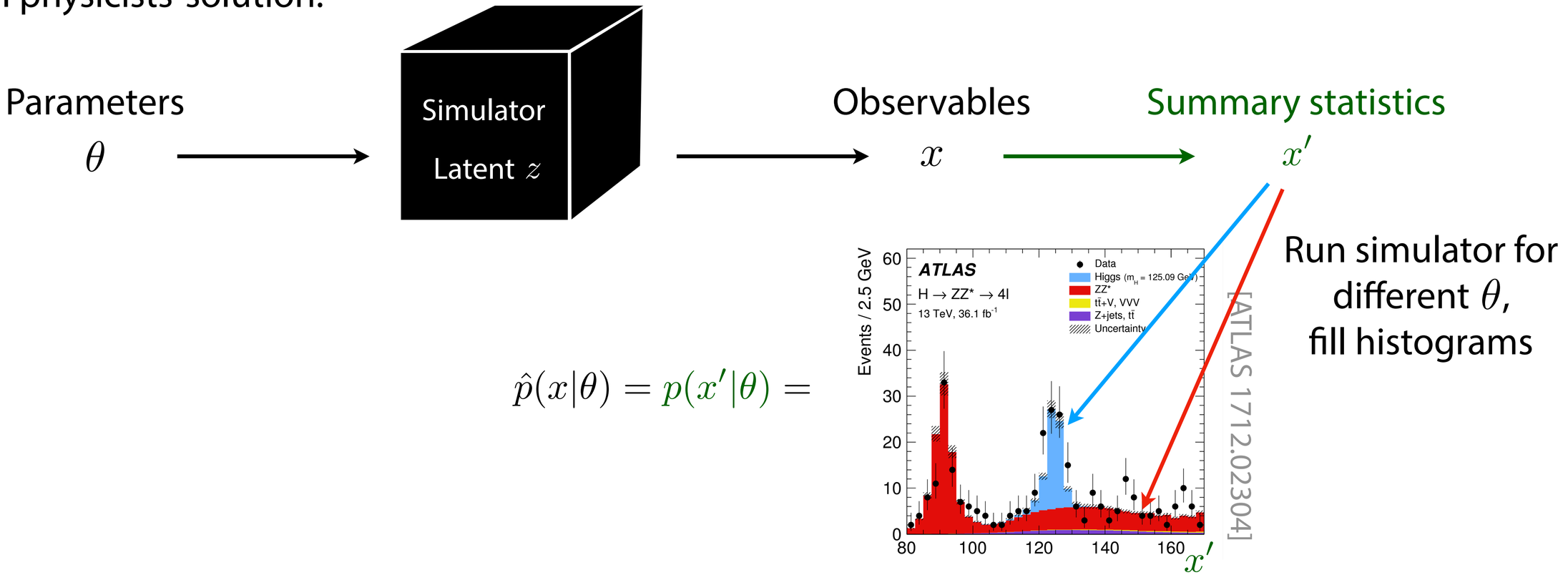
Solve it by histogramming summary statistics

- Typical physicists' solution:



Solve it by histogramming summary statistics

- Typical physicists' solution:



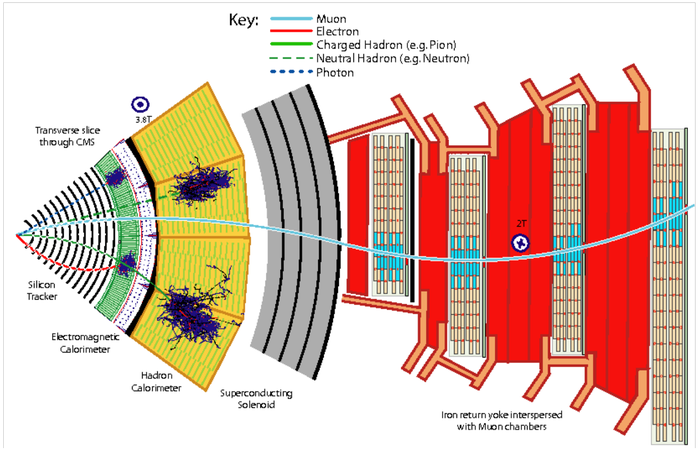
- How to choose x' ? Standard variables often lose information
  [JB, K. Cranmer, F. Kling, T. Plehn 1612.05261; JB, F. Kling, T. Plehn, T. Tait 1712.02350]

- "Curse of dimensionality": Histograms don't scale to high-dimensional x

Solve it by approximating the integral

- Problem: high-dim. integral over shower / detector trajectories

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



[CMS]

Solve it by approximating the integral

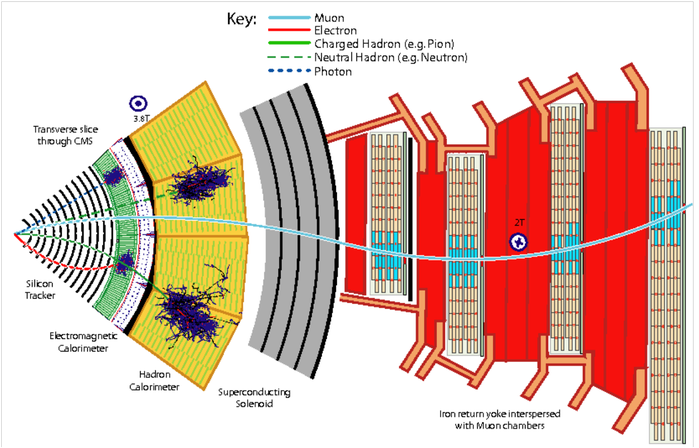
- Problem: high-dim. integral over shower / detector trajectories

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

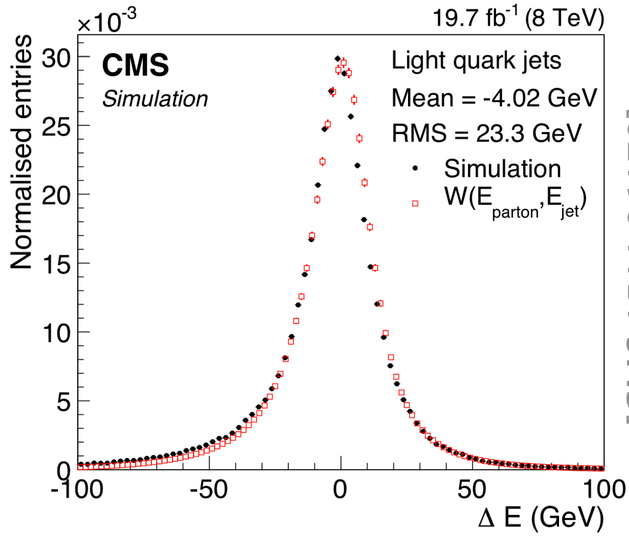
- Matrix Element Method: [K. Kondo 1988]
  - approximate shower + detector effects into transfer function $\hat{p}(x|z_p)$
  - explicitly calculate remaining integral

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

- Shower / Event Deconstruction [D. E. Soper, M. Spannowsky 1102.3480]
  extend explicit calculation to the shower



[CMS]



[CMS 1511.06170]

Solve it by approximating the integral

- Problem: high-dim. integral over shower / detector trajectories

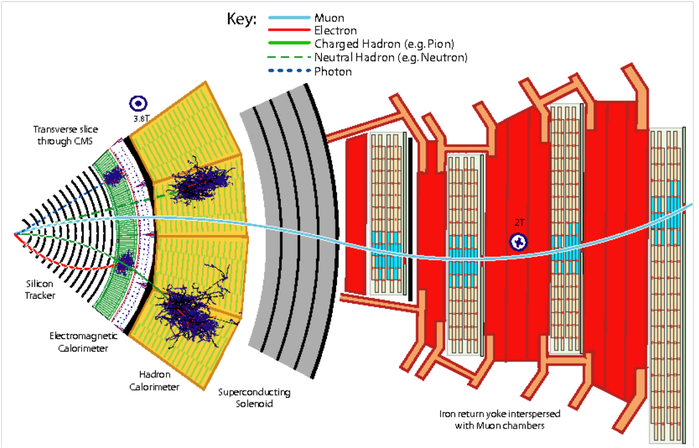
$$p(x|\theta) = \int \mathrm{d}z_d \int \mathrm{d}z_s \int \mathrm{d}z_p \; p(x|z_d)\, p(z_d|z_s)\, p(z_s|z_p)\, p(z_p|\theta)$$

- Matrix Element Method: [K. Kondo 1988]
  - approximate shower + detector effects into transfer function $\hat{p}(x|z_p)$
  - explicitly calculate remaining integral

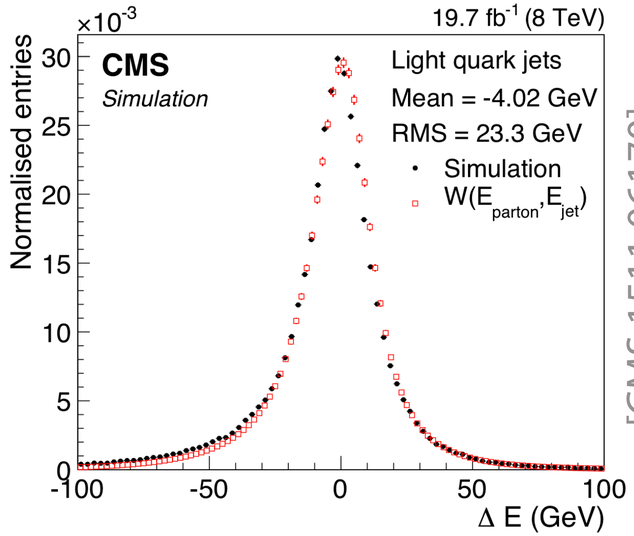
$$\hat{p}(x|\theta) = \int \mathrm{d}z_p \; \hat{p}(x|z_p)\, p(z_p|\theta)$$

  - Shower / Event Deconstruction [D. E. Soper, M. Spannowsky 1102.3480]
    extend explicit calculation to the shower

- Uses matrix-element information, no summary statistics necessary, but:
  - ad-hoc transfer functions (what about extra radiation?)
  - evaluation still requires calculating an expensive integral



[CMS]



[CMS 1511.06170]

What if we could estimate the likelihood…

- for high-dimensional measurements, including correlations?

  like MEM: no need to pick summary statistics

- including state-of-the-art shower and detector models?

  allowing for extra radiation, no need for ad-hoc transfer functions

- in microseconds?

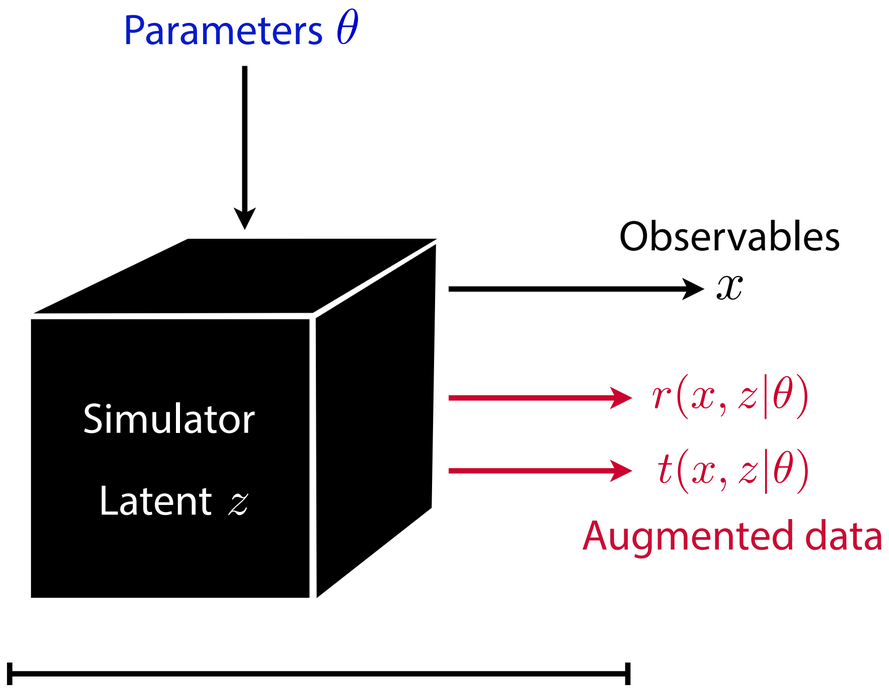
  amortized inference: train once, evaluate fast

- requiring less statistics than established machine learning methods?

  using matrix element information

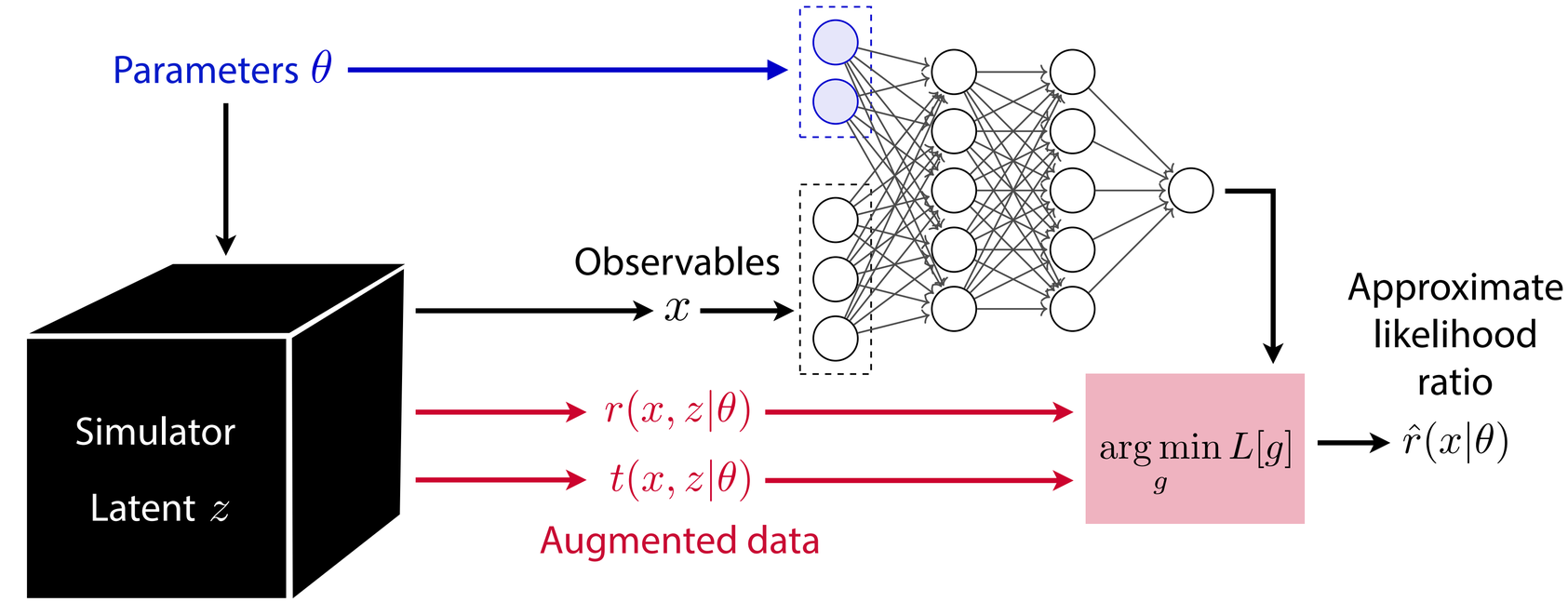
A new approach:
Mining gold from the simulator

Bird's-eye view



"Mining gold": Extract additional information from simulator

Bird's-eye view



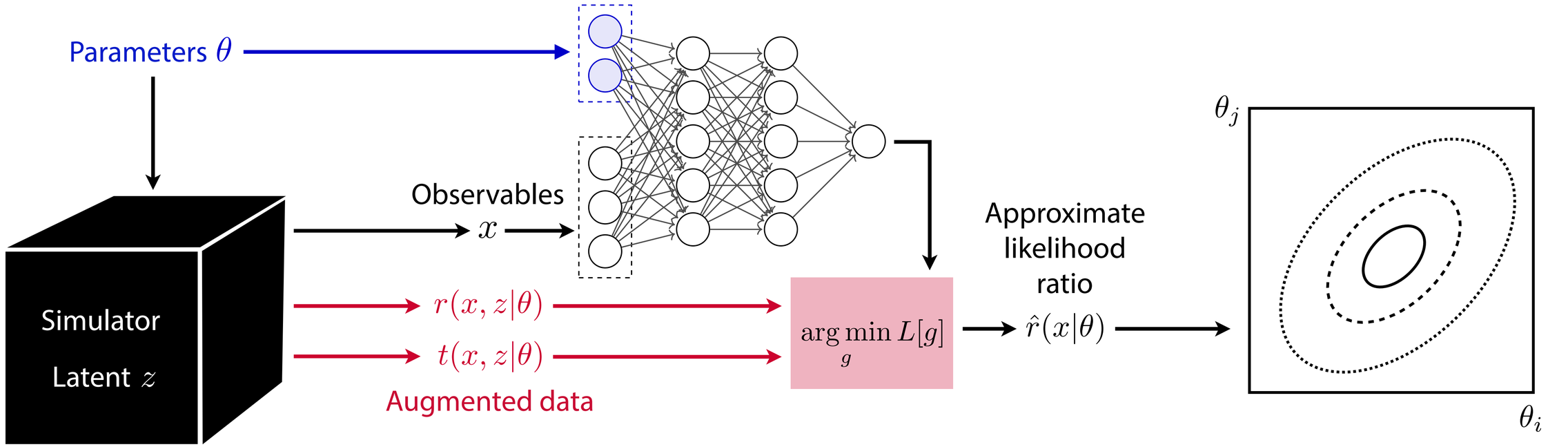
1. Simulation

"Mining gold": Extract additional information from simulator

2. Machine Learning

Use this information to train estimator for likelihood ratio

Bird's-eye view



1. Simulation

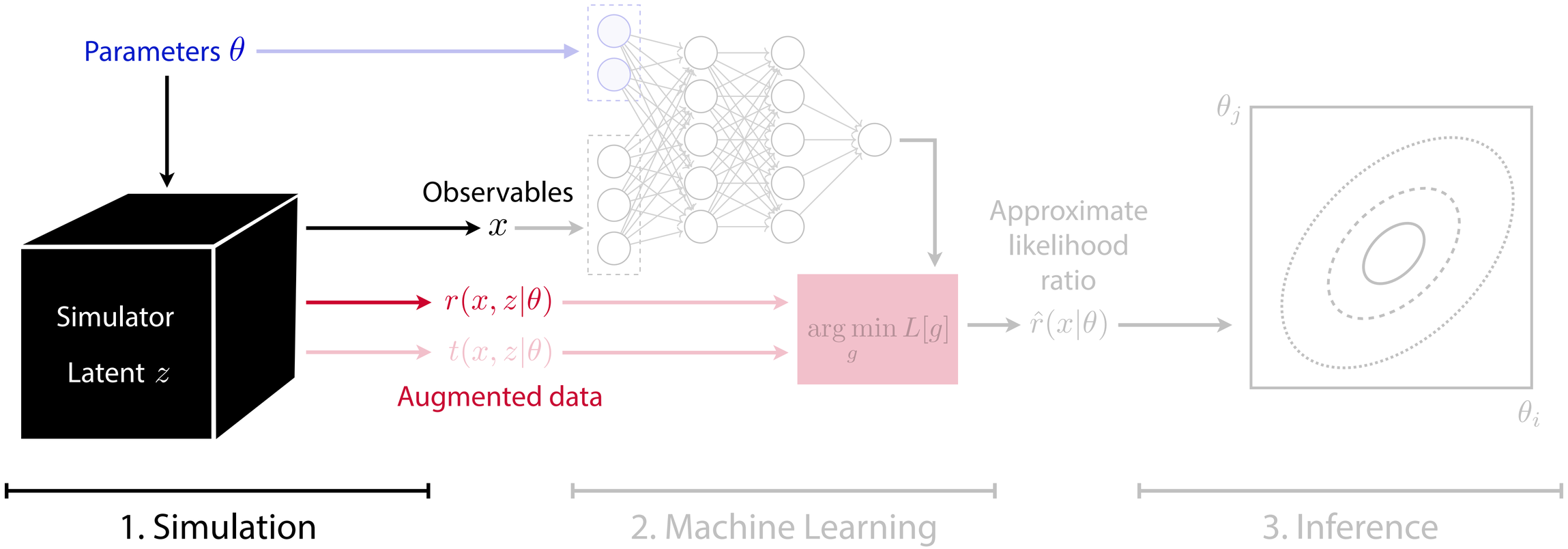
"Mining gold": Extract additional information from simulator

2. Machine Learning

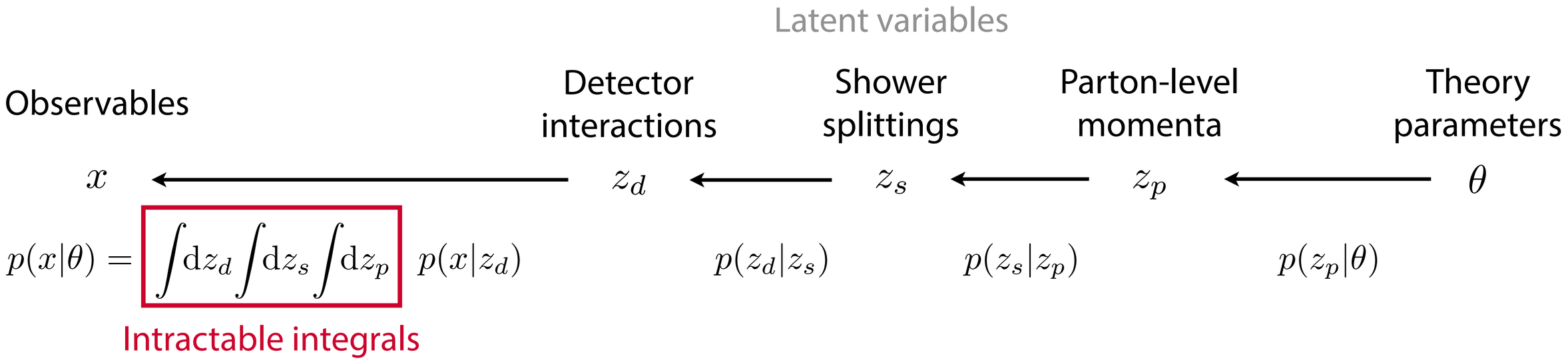
Use this information to train estimator for likelihood ratio

3. Inference

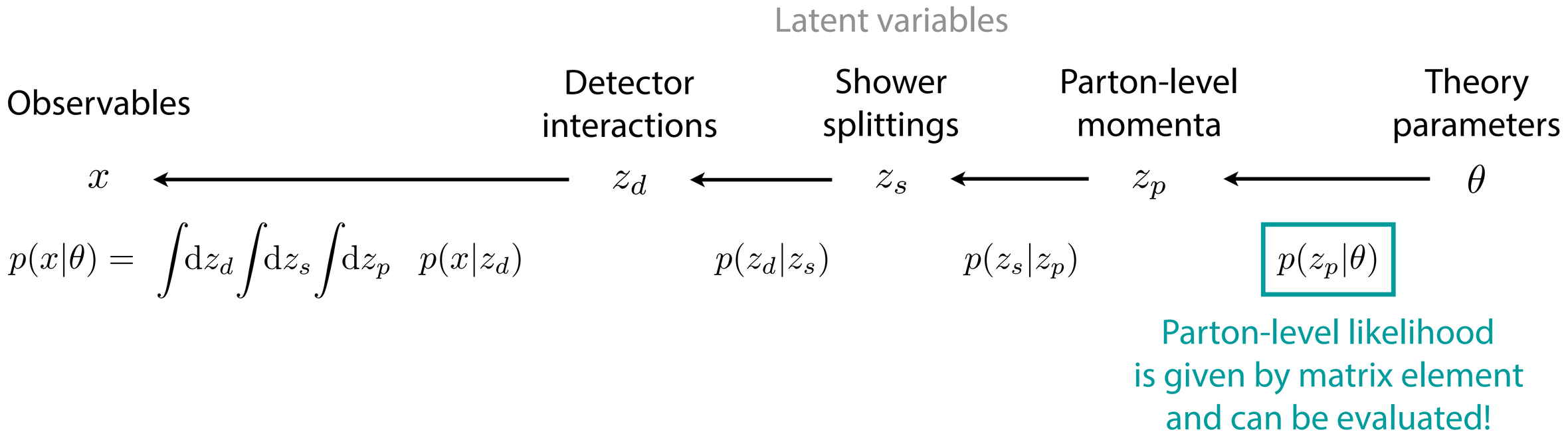
Limit setting with standard hypothesis tests


Parameters θ
Observables
x
Simulator
Latent z
$r(x, z|\theta)$
$t(x, z|\theta)$
Augmented data
$\arg\min_g L[g]$
Approximate
likelihood
ratio
$\hat{r}(x|\theta)$
θ_j
θ_i
1. Simulation
2. Machine Learning
3. Inference

Mining gold from the simulator

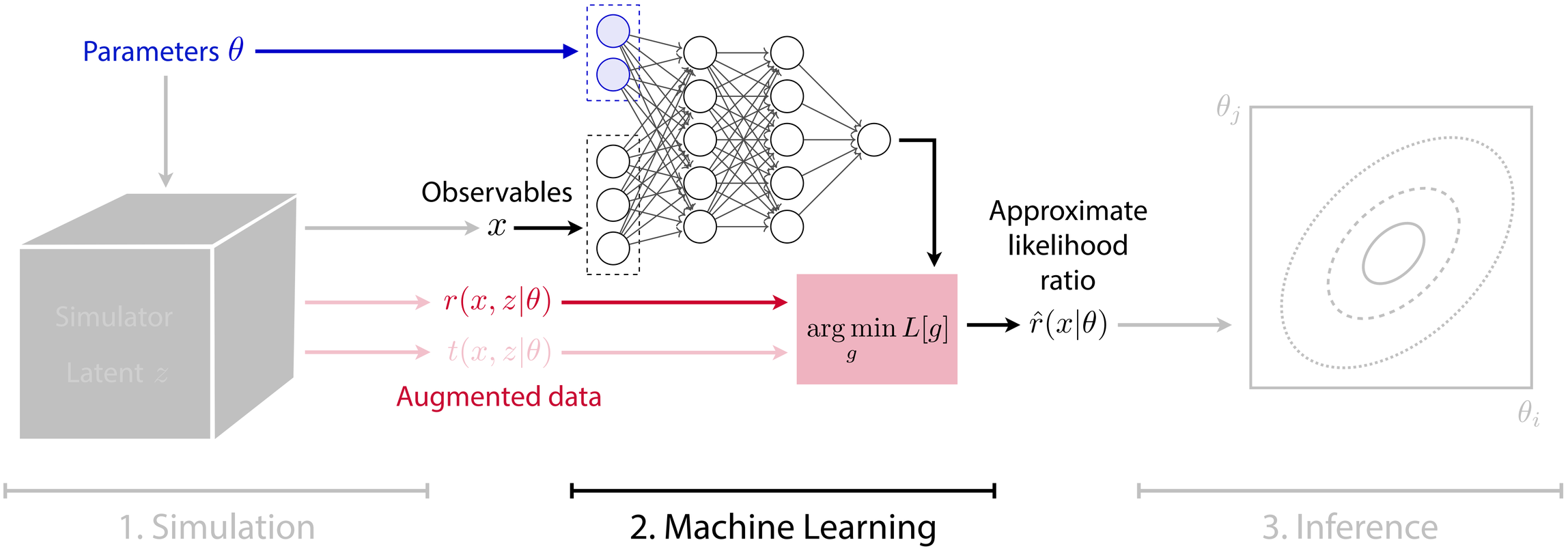


Mining gold from the simulator



⇒ For each generated event, we can calculate the joint likelihood ratio conditional on its specific evolution:

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)} = \frac{p(x|z_d)}{p(x|z_d)} \frac{p(z_d|z_s)}{p(z_d|z_s)} \frac{p(z_s|z_p)}{p(z_s|z_p)} \quad \boxed{\frac{p(z_p|\theta_0)}{p(z_p|\theta_1)} \sim \frac{|\mathcal{M}(z_p|\theta_0)|^2}{|\mathcal{M}(z_p|\theta_1)|^2}}$$


Parameters θ
Observables
x
Simulator
Latent z
$r(x, z|\theta)$
$t(x, z|\theta)$
Augmented data
$\arg\min_g L[g]$
Approximate
likelihood
ratio
$\hat{r}(x|\theta)$
θ_j
θ_i
1. Simulation
2. Machine Learning
3. Inference

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

We want the likelihood ratio function

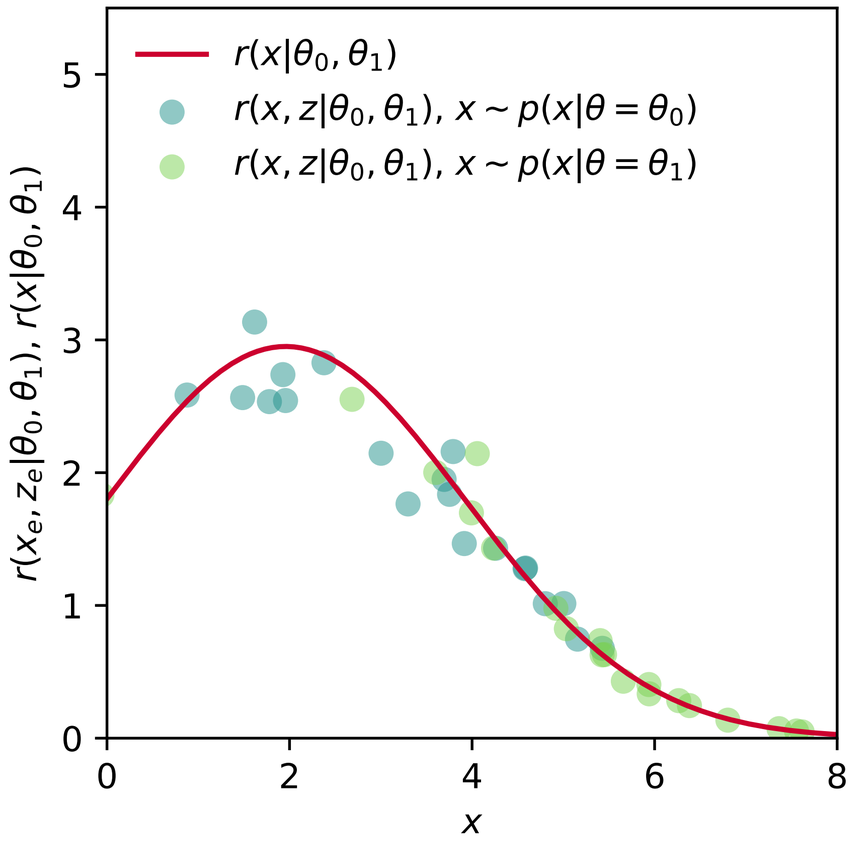
$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

$r(x, z|\theta_0, \theta_1)$ are scattered around $r(x|\theta_0, \theta_1)$



We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

$$\mathbb{E}_{z\sim p(z|x,\theta_1)}\left[r(x, z|\theta_0, \theta_1)\right] = \int \mathrm{d}z\; p(z|x, \theta_1)\, \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)}$$
$$= \int \mathrm{d}z\; \frac{p(x, z|\theta_1)}{p(x|\theta_1)}\, \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)}$$
$$= r(x|\theta_0, \theta_1)$$

We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

With $r(x, z|\theta_0, \theta_1)$, we define a functional like

$$L_r[\hat{r}(x|\theta_0, \theta_1)] = \int \mathrm{d}x \int \mathrm{d}z\, p(x, z|\theta_1) \left[(\hat{r}(x|\theta_0, \theta_1) - r(x, z|\theta_0, \theta_1))^2\right].$$

It is minimized by

$$r(x|\theta_0, \theta_1) = \underset{\hat{r}(x|\theta_0,\theta_1)}{\arg\min}\, L_r[\hat{r}(x|\theta_0, \theta_1)]!$$

(And we can sample from $p(x, z|\theta)$ by running the simulator.)

We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

Machine learning = applied calculus of variatios

We can get a precise estimator of the likelihood ratio by numerically mimimizing a functional:

$$\hat{r}(x|\theta_0,\theta_1) = \underset{\hat{r}(x|\theta_0,\theta_1)}{\arg\min} \underbrace{\int \mathrm{d}x \int \mathrm{d}z \; p(x,z|\theta_1) \left[\hat{r}(x|\theta_0,\theta_1) - r(x,z|\theta_0,\theta_1)\right]^2}_{L_r[\hat{r}(x|\theta_0,\theta_1)]}$$

Machine learning = applied calculus of variatios

We can get a precise estimator of the likelihood ratio by numerically mimimizing a functional:

$$\hat{r}(x|\theta_0,\theta_1) = \underset{\hat{r}(x|\theta_0,\theta_1)}{\arg\min} \underbrace{\int \mathrm{d}x \int \mathrm{d}z\; p(x,z|\theta_1) \left[\hat{r}(x|\theta_0,\theta_1) - r(x,z|\theta_0,\theta_1)\right]^2}_{L_r[\hat{r}(x|\theta_0,\theta_1)]}$$

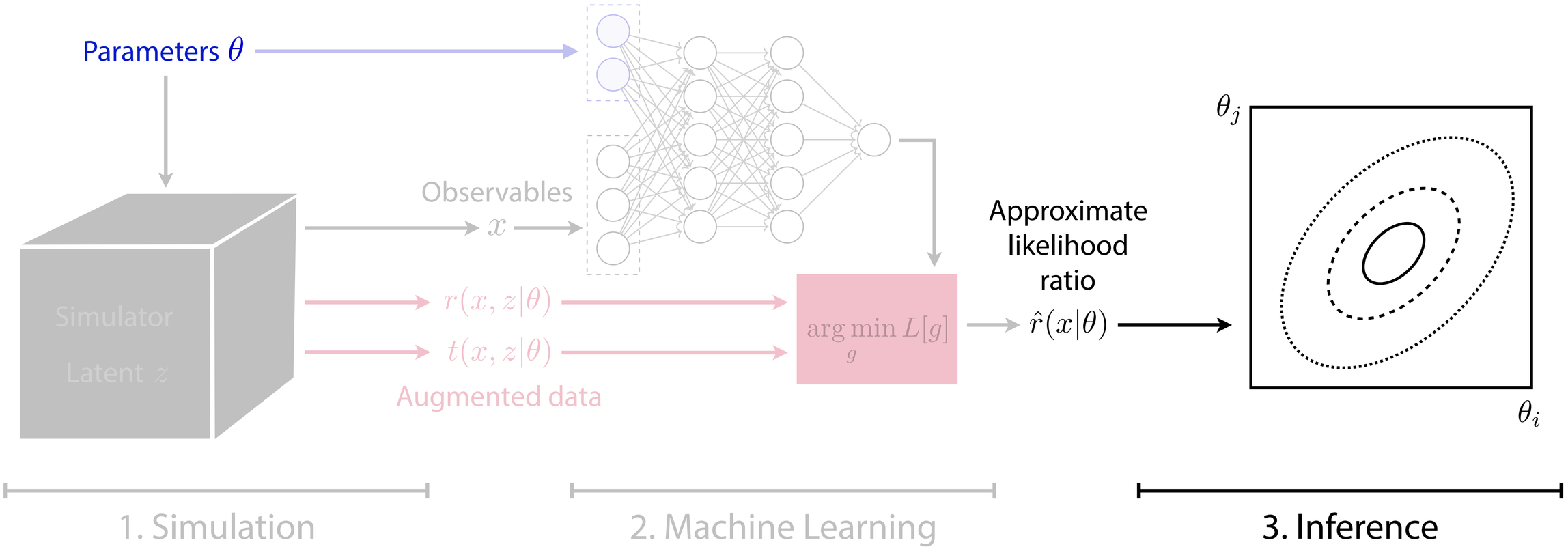
We do this through machine learning:

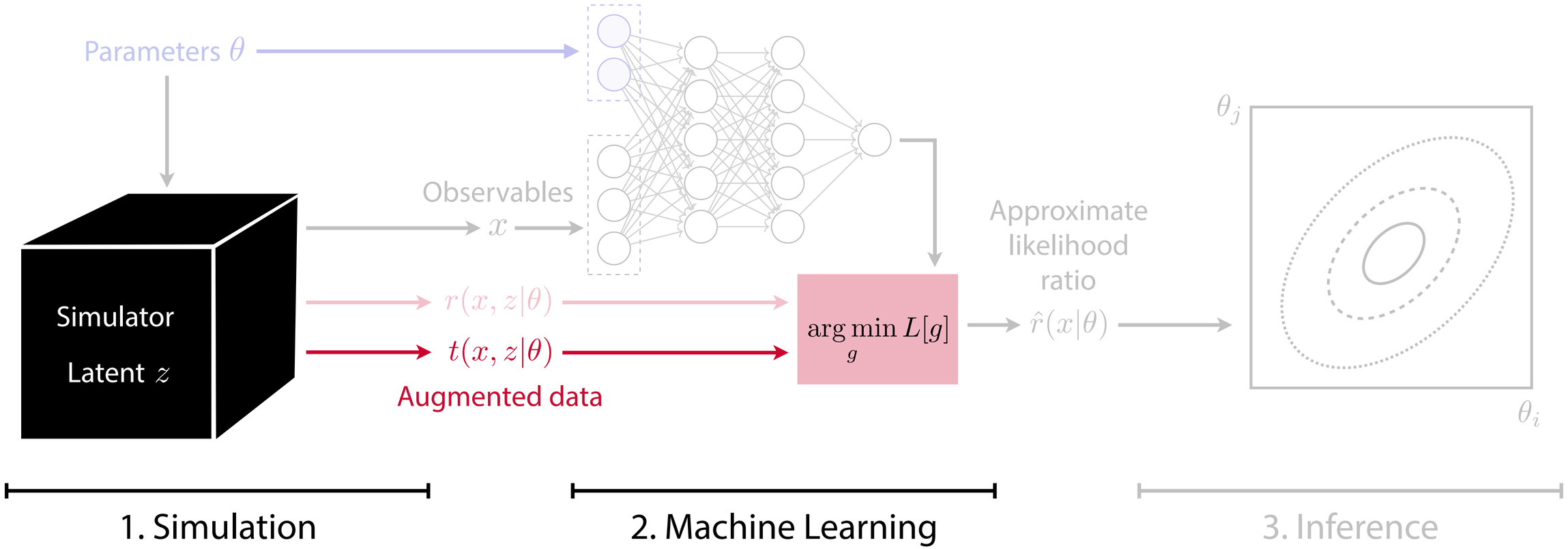
- Functional L_r $\longrightarrow$ Loss function

$$\hat{L}_r[\hat{r}(x|\theta_0,\theta_1)] = \frac{1}{N} \sum_{(x_i,z_i)\sim p(x,z|\theta_1)} \left[\hat{r}(x_i|\theta_0,\theta_1) - r(x_i,z_i|\theta_0,\theta_1)\right]^2$$

- Variational family $\hat{r}(x|\theta_0,\theta_1)$ $\longrightarrow$ Flexible parametric function (e.g. neural network)
- Exact minimization $\longrightarrow$ Numerical optimization algorithm (e.g. stochastic gradient descent)

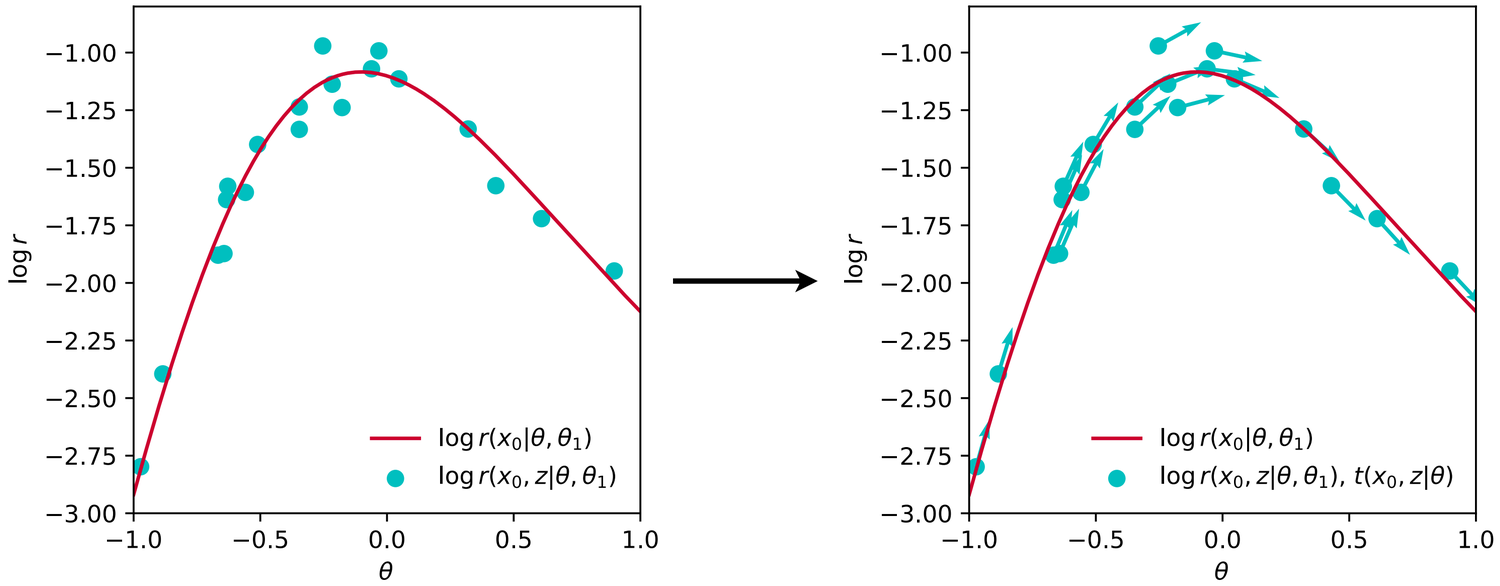
A sufficiently expressive neural network efficiently trained in this way with enough data will learn the likelihood ratio function $r(x|\theta_0,\theta_1)$!


Parameters θ
Observables
x
Simulator
Latent z
$r(x, z|\theta)$
$t(x, z|\theta)$
Augmented data
$\arg\min_g L[g]$
Approximate likelihood ratio
$\hat{r}(x|\theta)$
θ_j
θ_i
1. Simulation
2. Machine Learning
3. Inference


Parameters θ
Observables
x
Simulator
Latent z
$r(x, z|\theta)$
$t(x, z|\theta)$
Augmented data
$\arg\min_g L[g]$
Approximate
likelihood
ratio
$\hat{r}(x|\theta)$
θ_j
θ_i
1. Simulation
2. Machine Learning
3. Inference

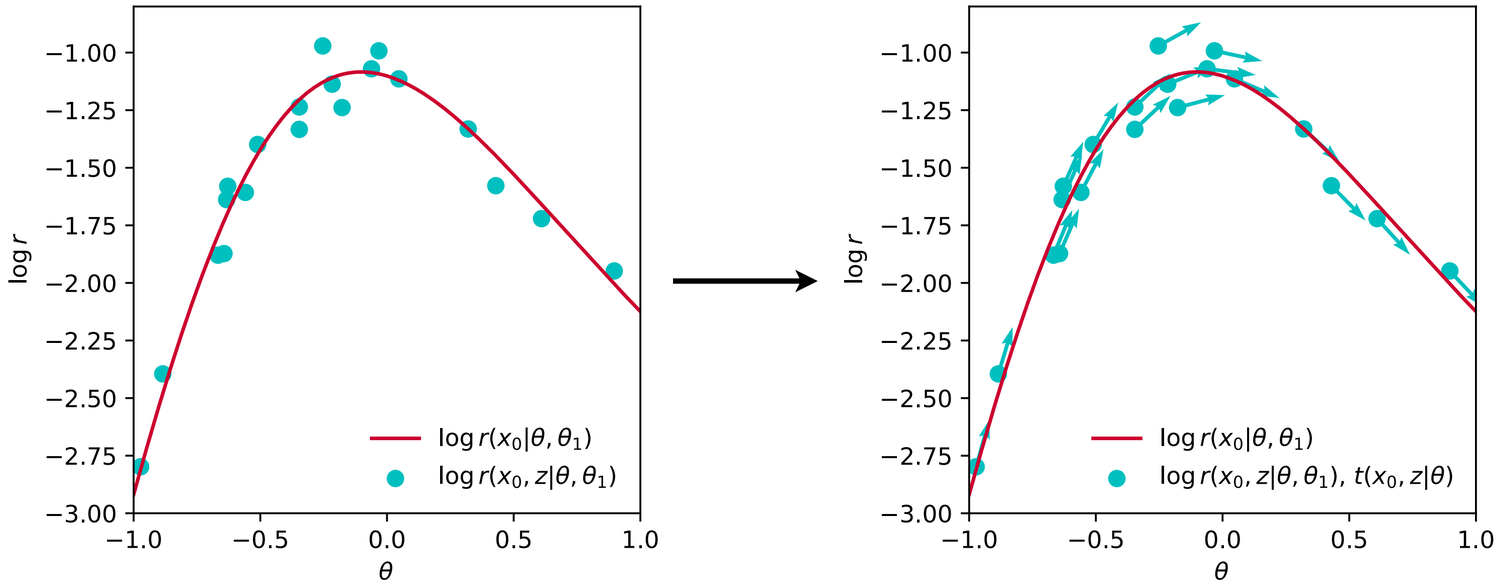
One more piece: the score

- Knowing derivative often helps fitting:



One more piece: the score

- Knowing derivative often helps fitting:



- In our case, the relevant quantity is the score $t(x|\theta_0) \equiv \nabla_\theta \log p(x|\theta)\Big|_{\theta_0}$.

  - The score fully characterizes the likelihood function in the neighborhood of θ_0
  - The score itself is intractable. But…

Learning the score

Similar to the joint likelihood ratio, from the simulator we can extract the joint score

$$t(x, z|\theta_0) \equiv \nabla_\theta \log p(x, z_d, z_s, z_p|\theta)\Big|_{\theta_0}$$

We want the score

$$t(x|\theta_0) \equiv \nabla_\theta \log p(x|\theta)\Big|_{\theta_0}$$

Learning the score

Similar to the joint likelihood ratio, from the simulator we can extract the joint score

$$t(x, z|\theta_0) \equiv \nabla_\theta \log p(x, z_d, z_s, z_p|\theta)\Big|_{\theta_0}$$

We want the score

$$t(x|\theta_0) \equiv \nabla_\theta \log p(x|\theta)\Big|_{\theta_0}$$

Given $t(x, z|\theta_0)$,
we define the functional

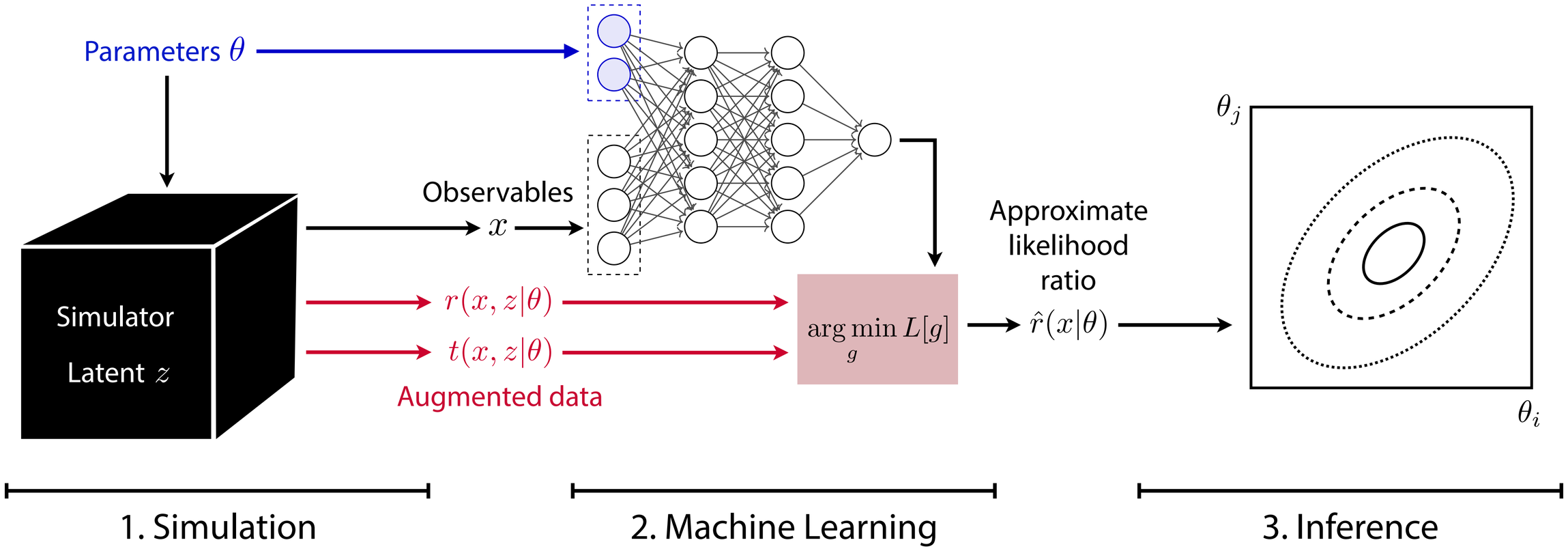
$$L_t[\hat{t}(x|\theta_0)] = \int \mathrm{d}x \int \mathrm{d}z \;\; p(x, z|\theta_0) \;\; \Big[\big(\hat{t}(x|\theta_0) - t(x, z|\theta_0)\big)^2\Big].$$

One can show it is minimized by

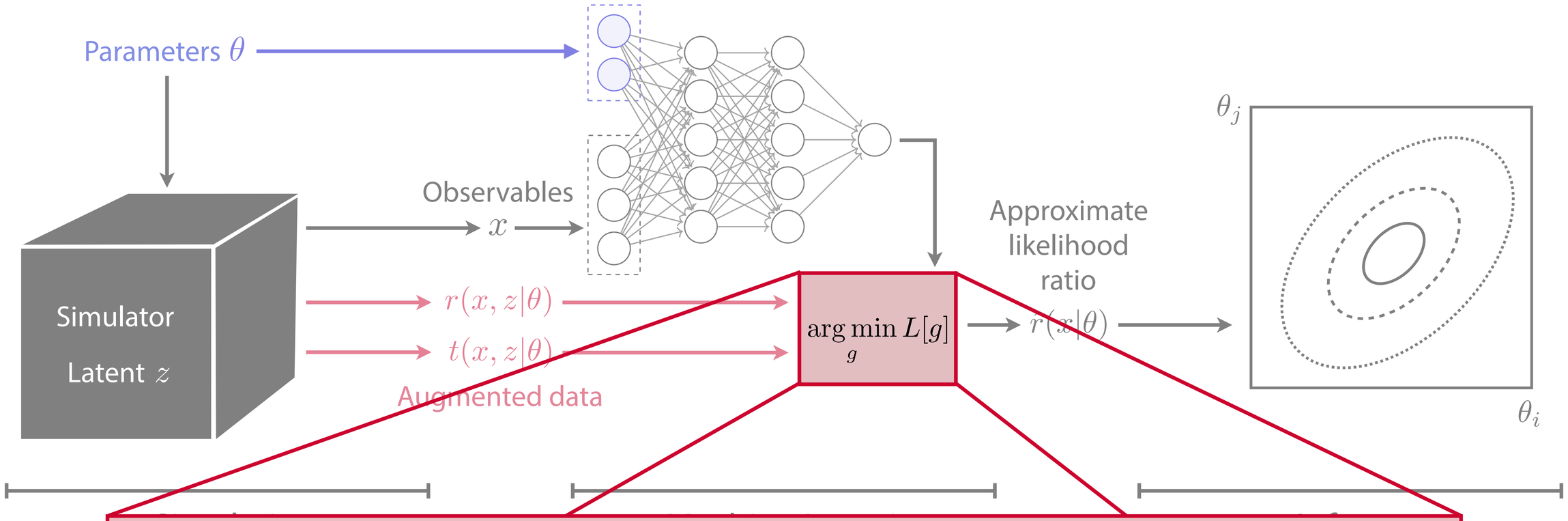
$$t(x|\theta_0) = \arg\min_{\hat{t}(x|\theta_0)} L_t[\hat{t}(x|\theta_0)]\,.$$

Again, we implement this minimization
through machine learning.

Putting the pieces together: RASCAL (Ratio and score approximate likelihood ratio)



Putting the pieces together: RASCAL (Ratio and score approximate likelihood ratio)

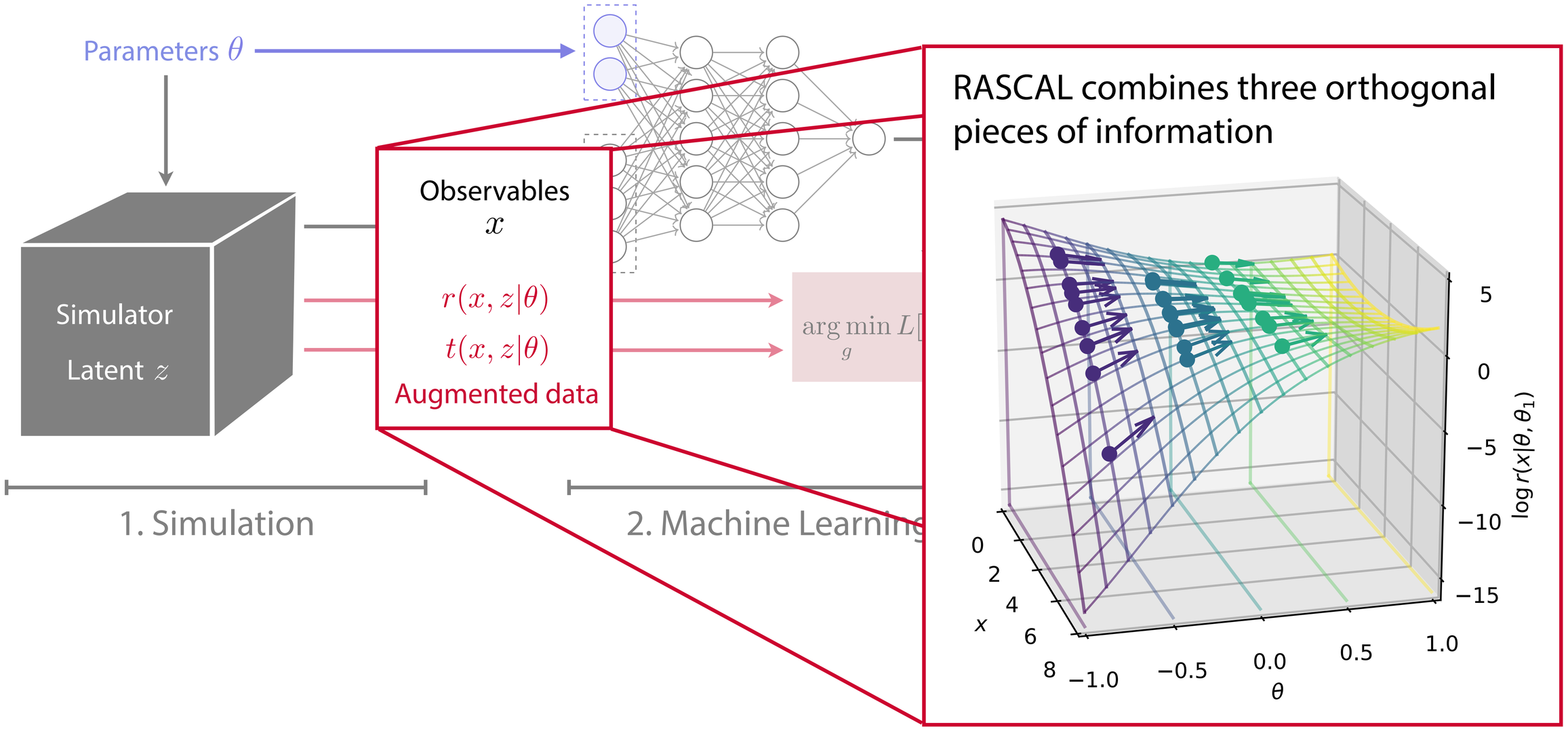


Combining ratio and score information:

- If ratio estimator is differentiable, we can calculate the estimated score $\hat{t}(x|\theta_0) = \nabla_{\theta_0} \log \hat{r}(x|\theta_0, \theta_1)$
- Minimize combined loss function:

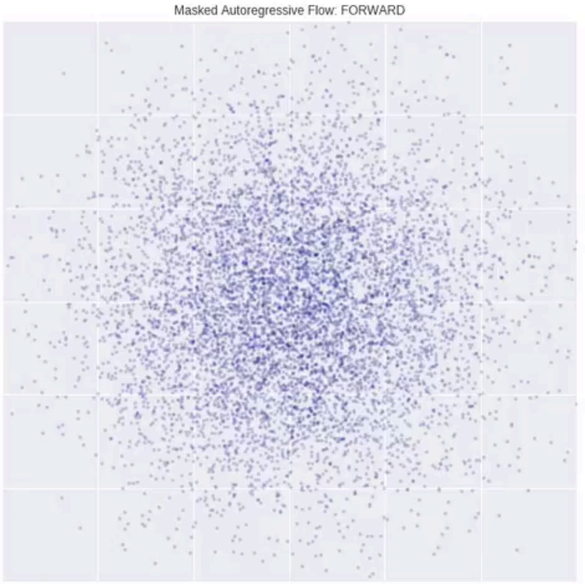
$$L_{\text{RASCAL}}[\hat{r}(x|\theta_0, \theta_1)] = L_r[\hat{r}(x|\theta_0, \theta_1)] + \alpha\, L_t[\nabla_{\theta_0} \log \hat{r}(x|\theta_0, \theta_1)]$$

Putting the pieces together: RASCAL (Ratio and score approximate likelihood ratio)



There’s much more…

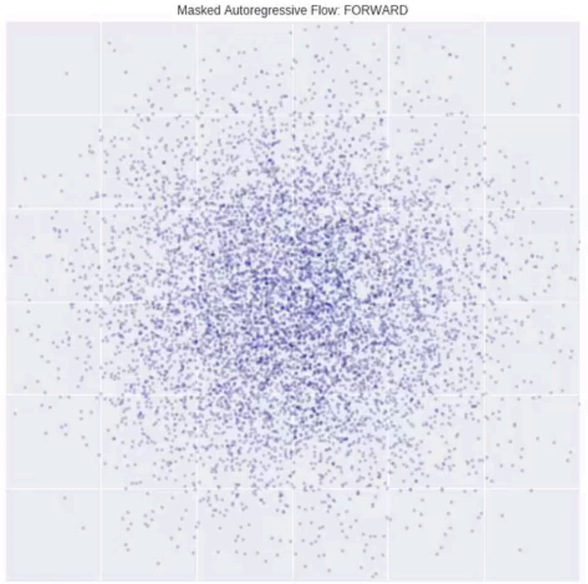
- More than one way to the likelihood (ratio)!
  - ALICE: use cross entropy instead of squared error loss
  - SCANDAL: combine with neural density estimators, e.g. Masked Autoregressive Flows
    [G. Papamakarios, T. Pavlakou, I. Murray 1705.07057]
  - SALLY / SALLINO: use estimated score as “optimal observable”



[Alex Mordvintsev]

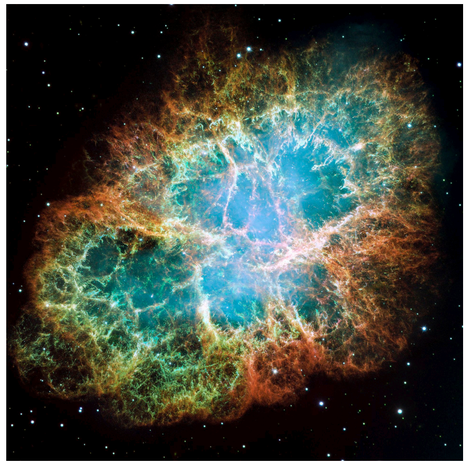
There's much more…

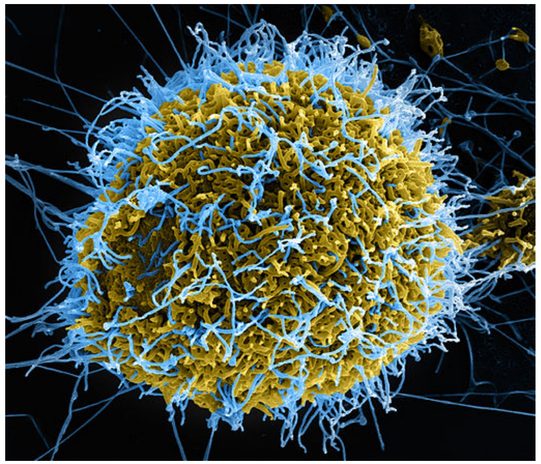
- More than one way to the likelihood (ratio)!
  - ALICE: use cross entropy instead of squared error loss
  - SCANDAL: combine with neural density estimators, e.g. Masked Autoregressive Flows
    [G. Papamakarios, T. Pavlakou, I. Murray 1705.07057]
  - SALLY / SALLINO: use estimated score as "optimal observable"



[Alex Mordvintsev]

- What if we don't fully trust the simulator?
  - Nuisance parameters to model systematic uncertainties
  - Learn robustness with adversarial training
    [G. Louppe, M. Kagan, K. Cranmer 1611.01046;
    C. Englert, P. Galler, P. Harris. M. Spannowsky 1807.08763 + CE's talk]





[NASA, NIAID]

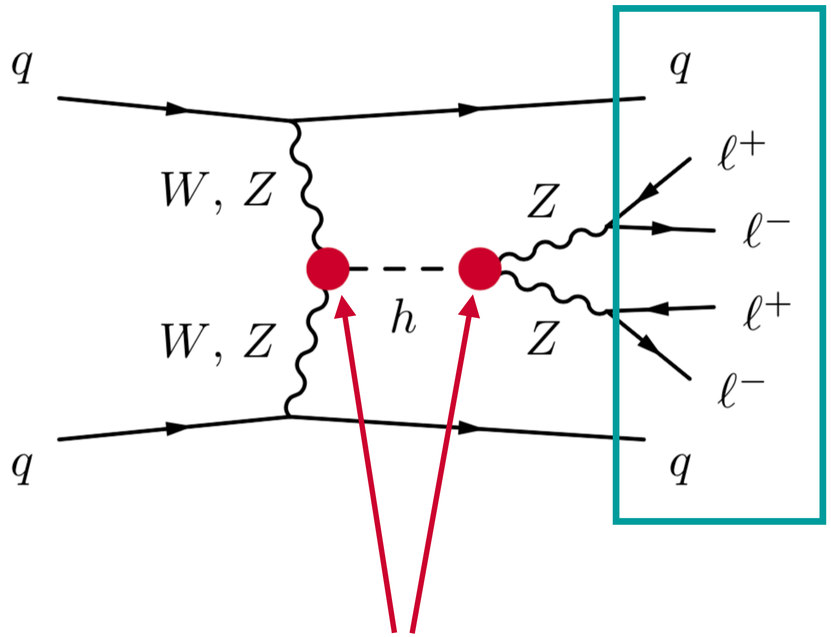
- More general than particle physics
  - Currently being adapted to cosmology, epidemiology

EFT example

Proof of concept

Higgs production in weak boson fusion:

at least 16-dimensional observable space



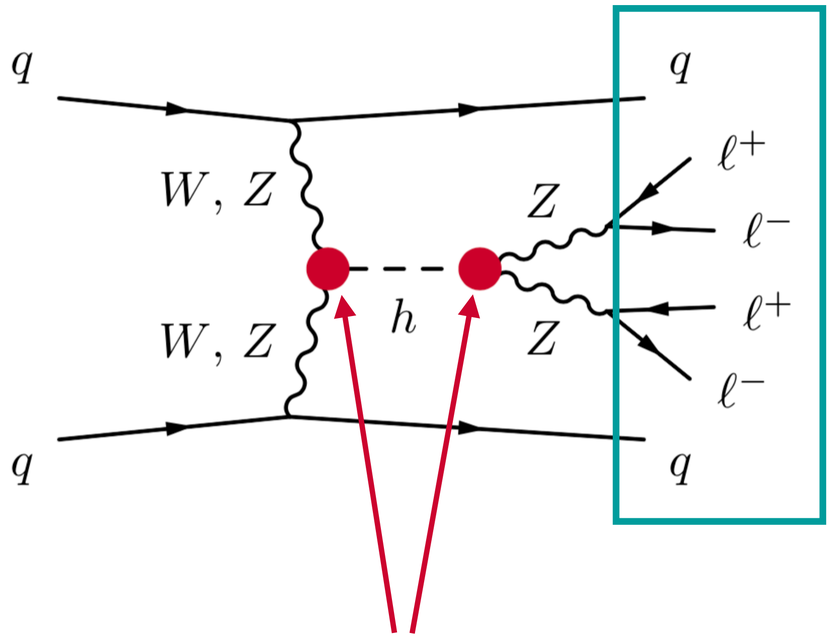
Exciting new physics might hide here!
We parameterize it with two EFT coefficients:

$$\mathcal{L} = \mathcal{L}_{\rm SM} + \boxed{\frac{f_W}{\Lambda^2}} \underbrace{\frac{ig}{2}\,(D^\mu\phi)^\dagger\,\sigma^a\,D^\nu\phi\,W^a_{\mu\nu}}_{\mathcal{O}_W} - \boxed{\frac{f_{WW}}{\Lambda^2}} \underbrace{\frac{g^2}{4}\,(\phi^\dagger\phi)\,W^a_{\mu\nu}\,W^{\mu\nu\,a}}_{\mathcal{O}_{WW}}$$

Proof of concept

Higgs production in weak boson fusion:

at least 16-dimensional observable space



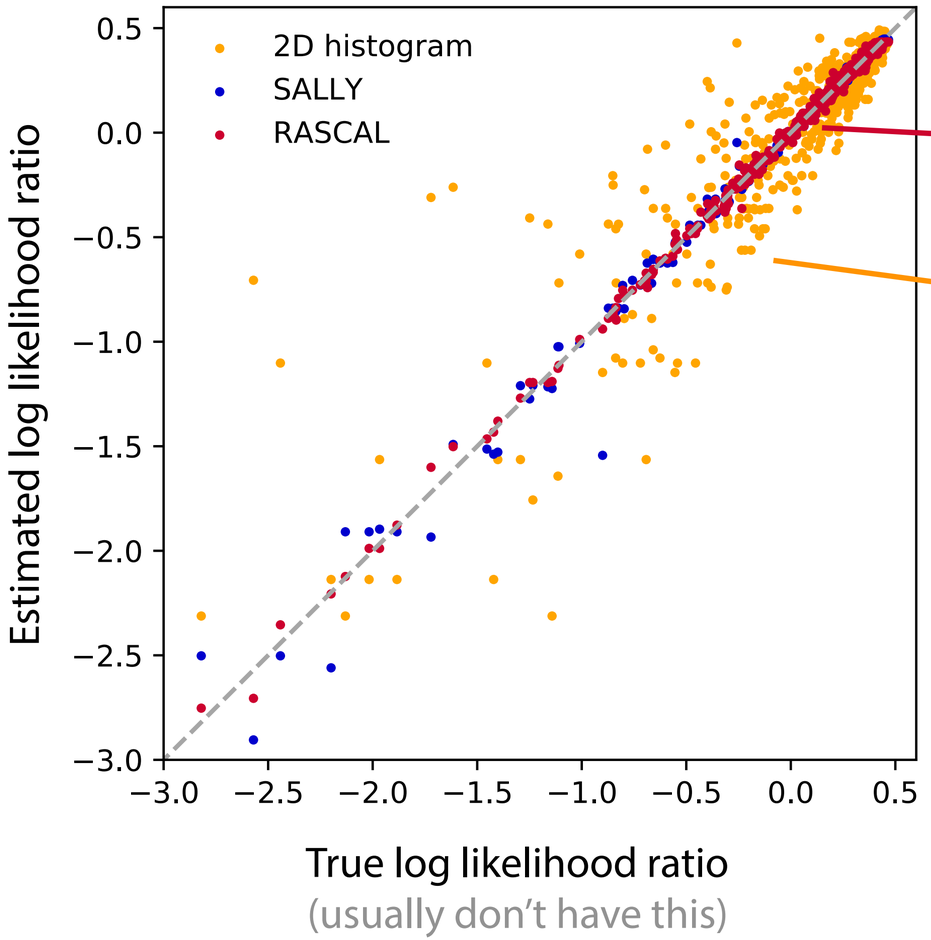
Exciting new physics might hide here!
We parameterize it with two EFT coefficients:

$$\mathcal{L} = \mathcal{L}_{\rm SM} + \boxed{\frac{f_W}{\Lambda^2}} \underbrace{\frac{ig}{2}\,(D^\mu\phi)^\dagger\,\sigma^a\,D^\nu\phi\,W^a_{\mu\nu}}_{\mathcal{O}_W} - \boxed{\frac{f_{WW}}{\Lambda^2}} \underbrace{\frac{g^2}{4}\,(\phi^\dagger\phi)\,W^a_{\mu\nu}\,W^{\mu\nu\,a}}_{\mathcal{O}_{WW}}$$

- Goal: constrain the two EFT parameters
  - new inference methods
  - baseline: 2d histogram analysis of jet momenta & angular correlations
- Two scenarios:
  - Simplified setup in which we can compare to true likelihood
  - "Realistic" simulation with approximate detector effects
- Simulation: MadGraph + MadMax
  [J. Alwall et al. 1405.0301; K. Cranmer, T. Plehn hep-ph/0605268; T. Plehn, P. Schichtel, D. Wiegand 1311.2591]

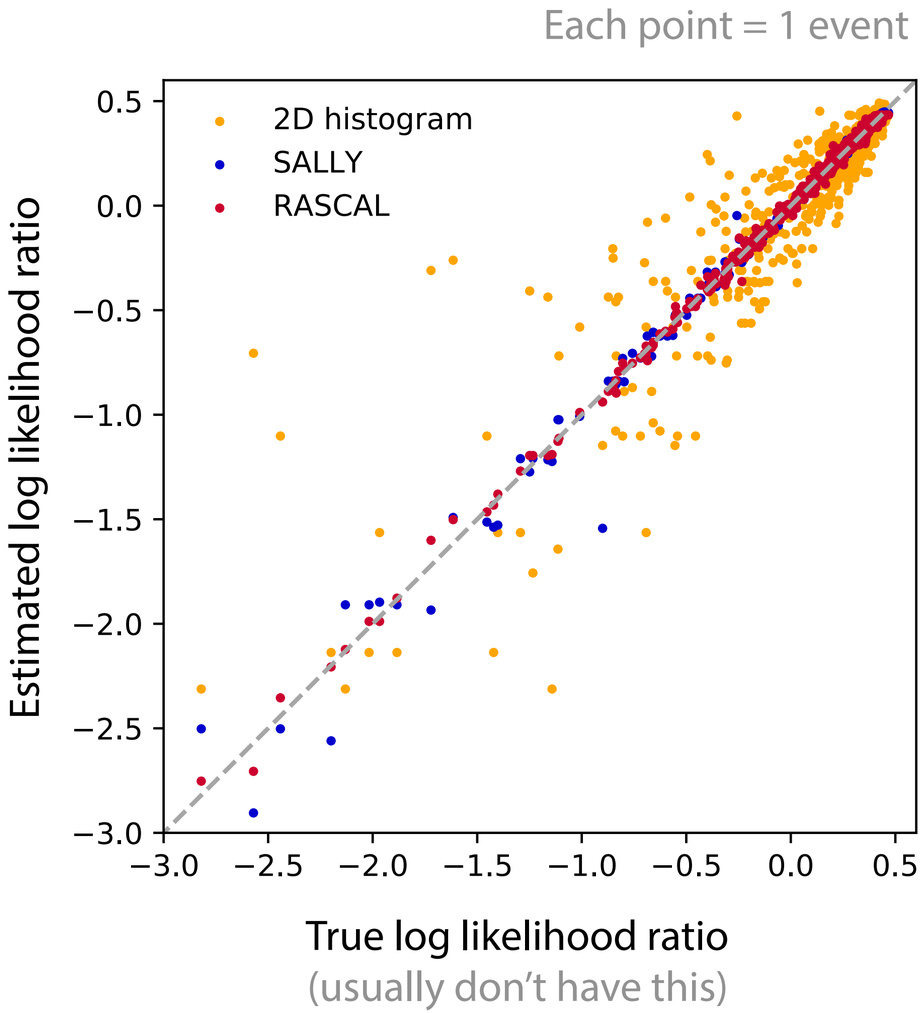
More precise likelihood ratio estimates with less training data

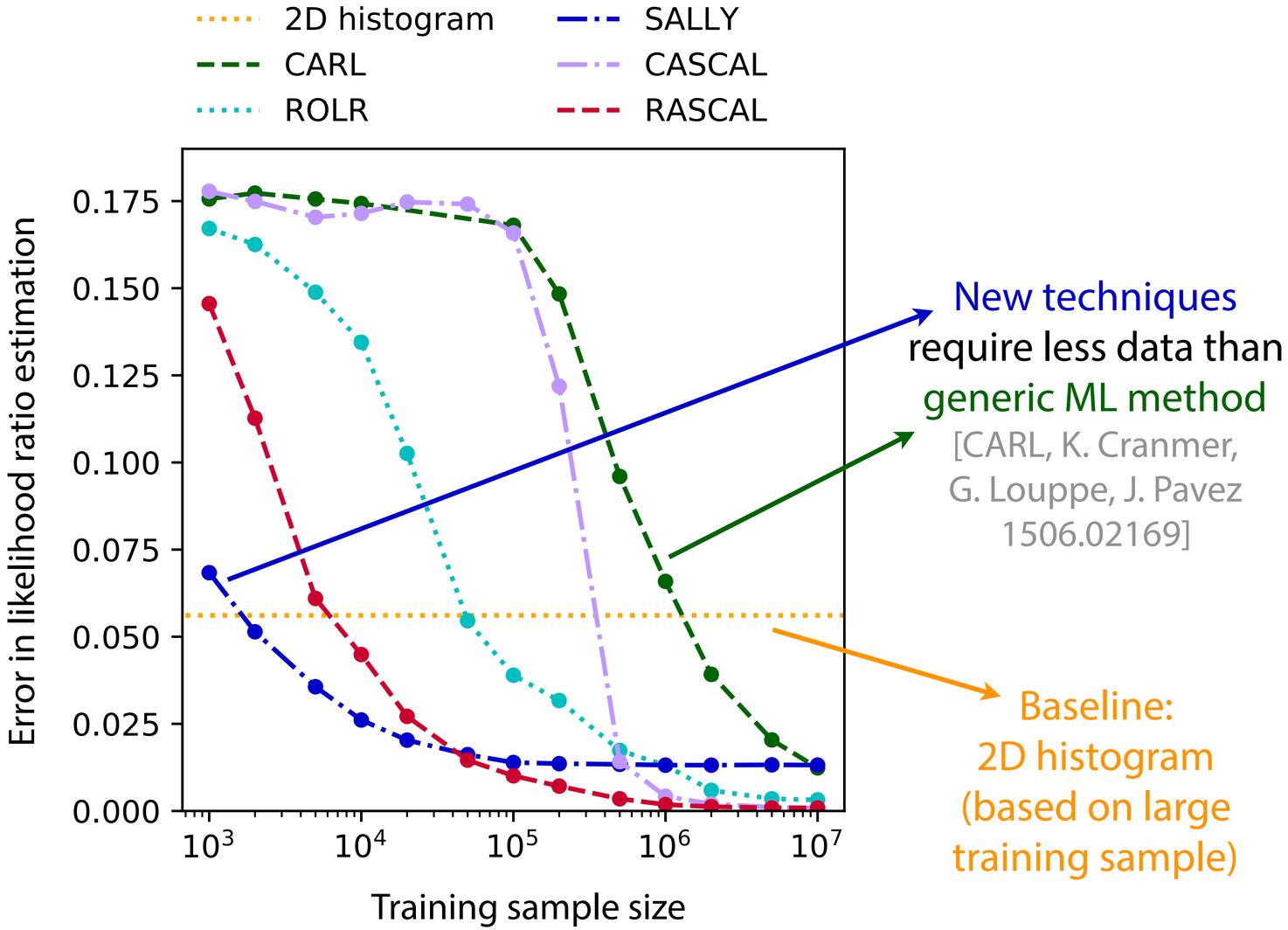
Each point = 1 event



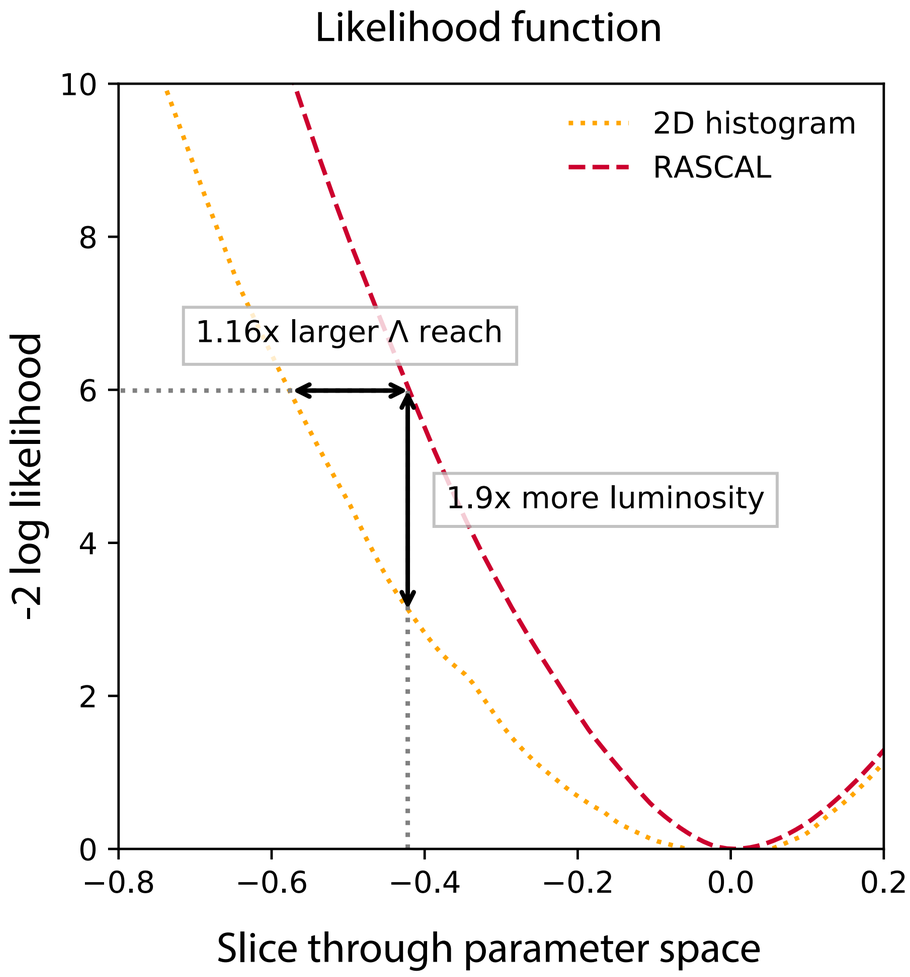
RASCAL
gives more precise likelihood ratio estimates than the
2D histogram

More precise likelihood ratio estimates with less training data



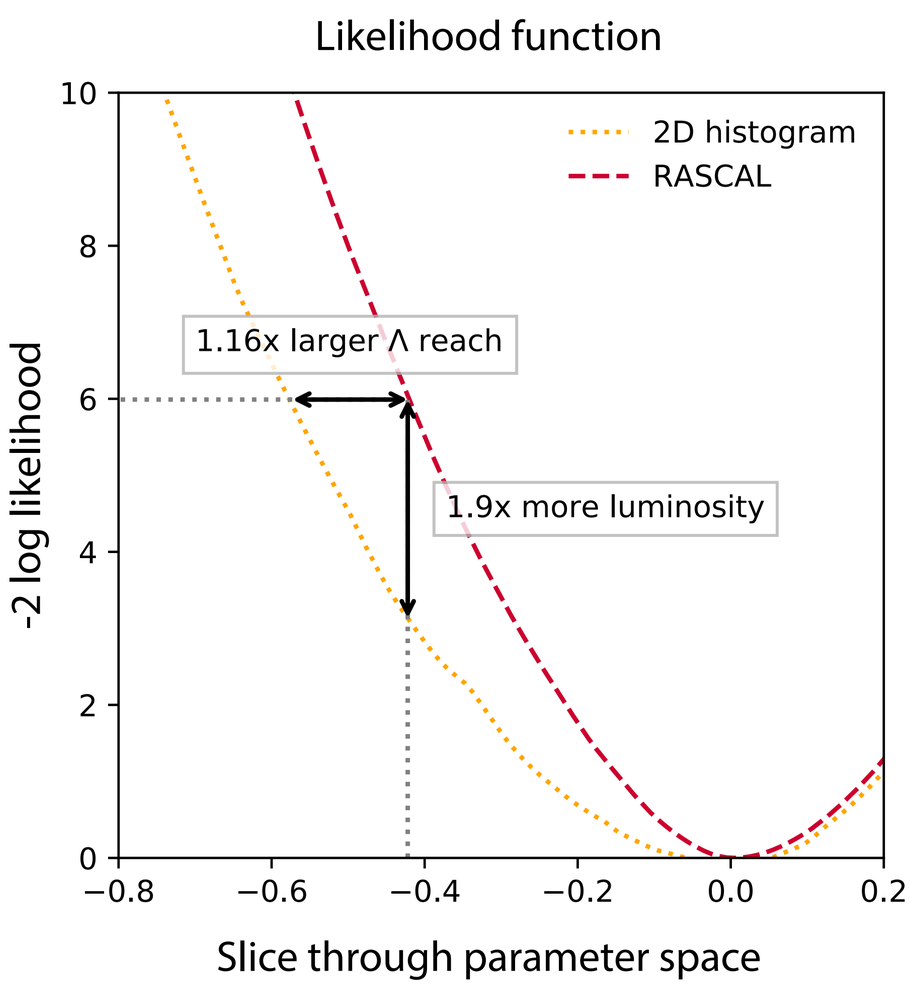


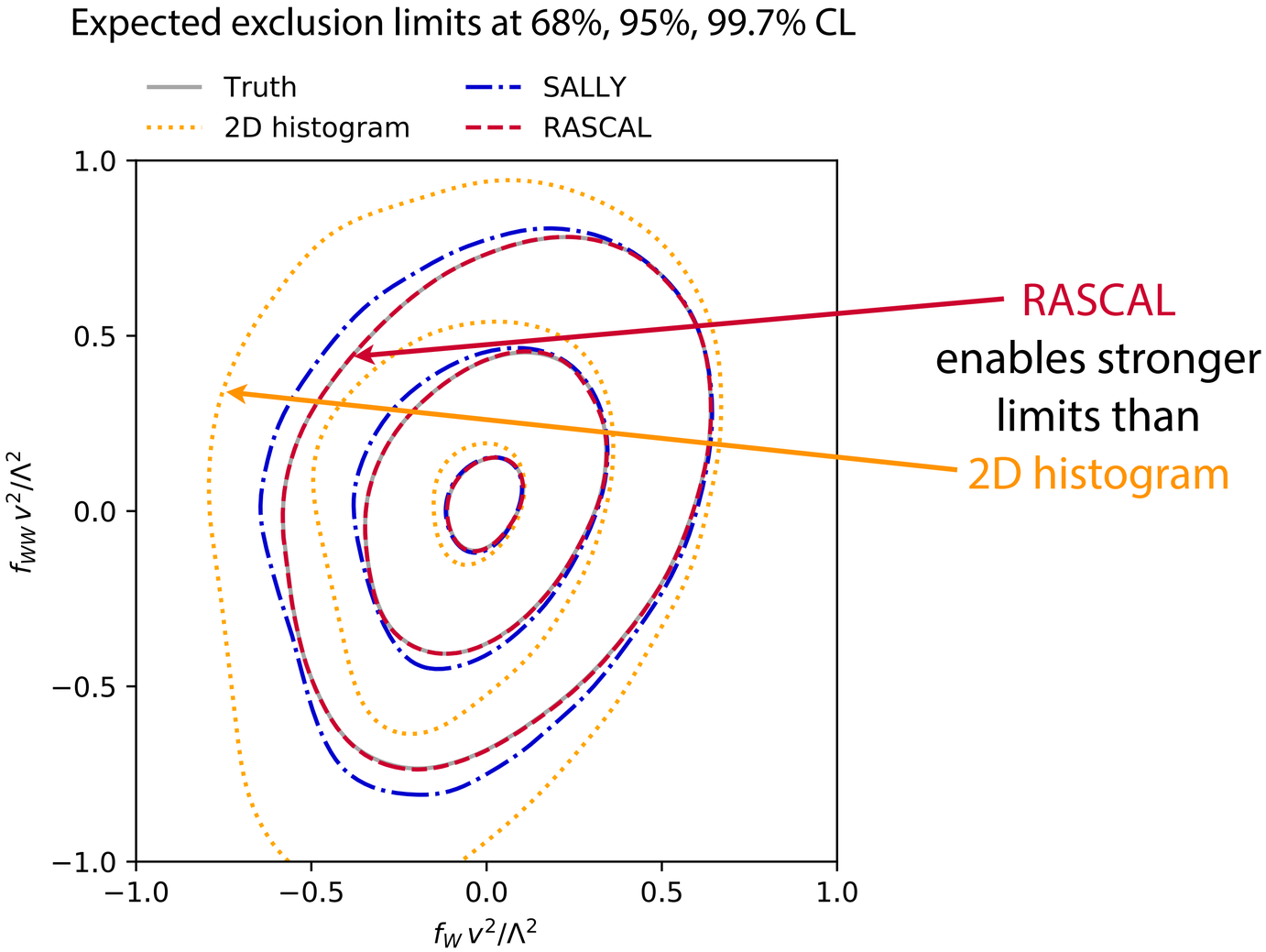
Better sensitivity to new physics



Results are based on 36 observed events, assuming SM

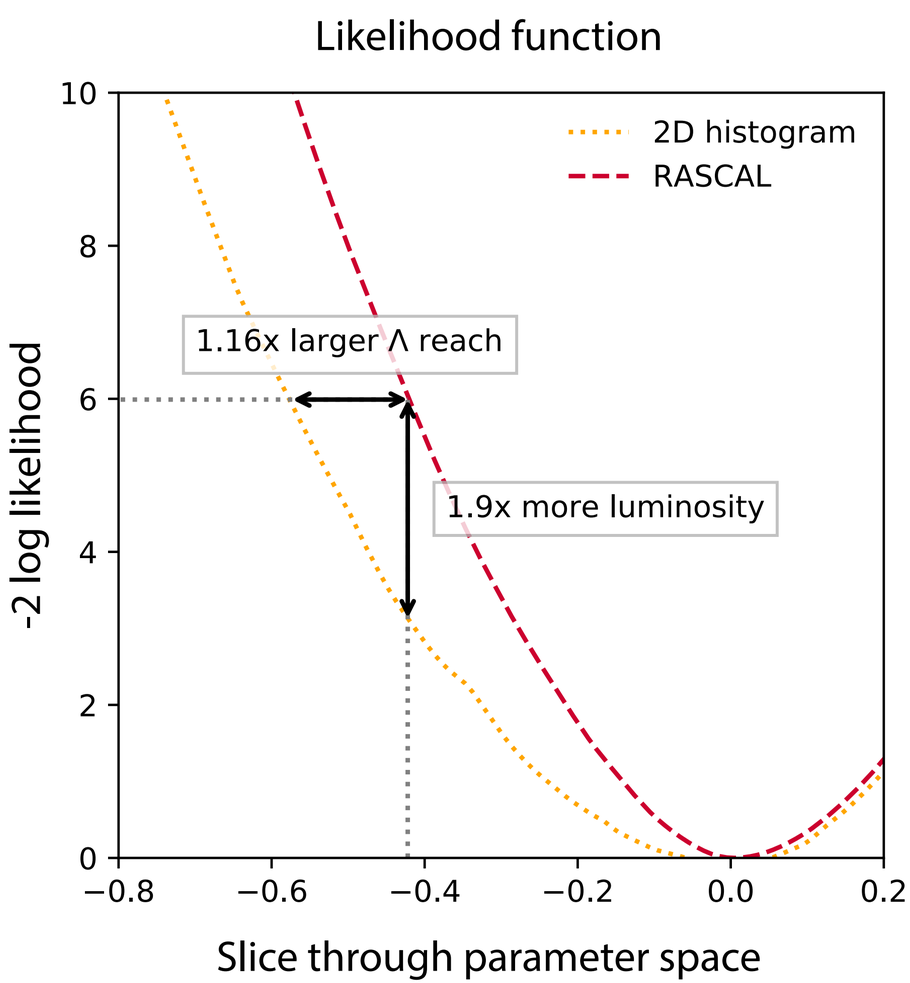
Better sensitivity to new physics

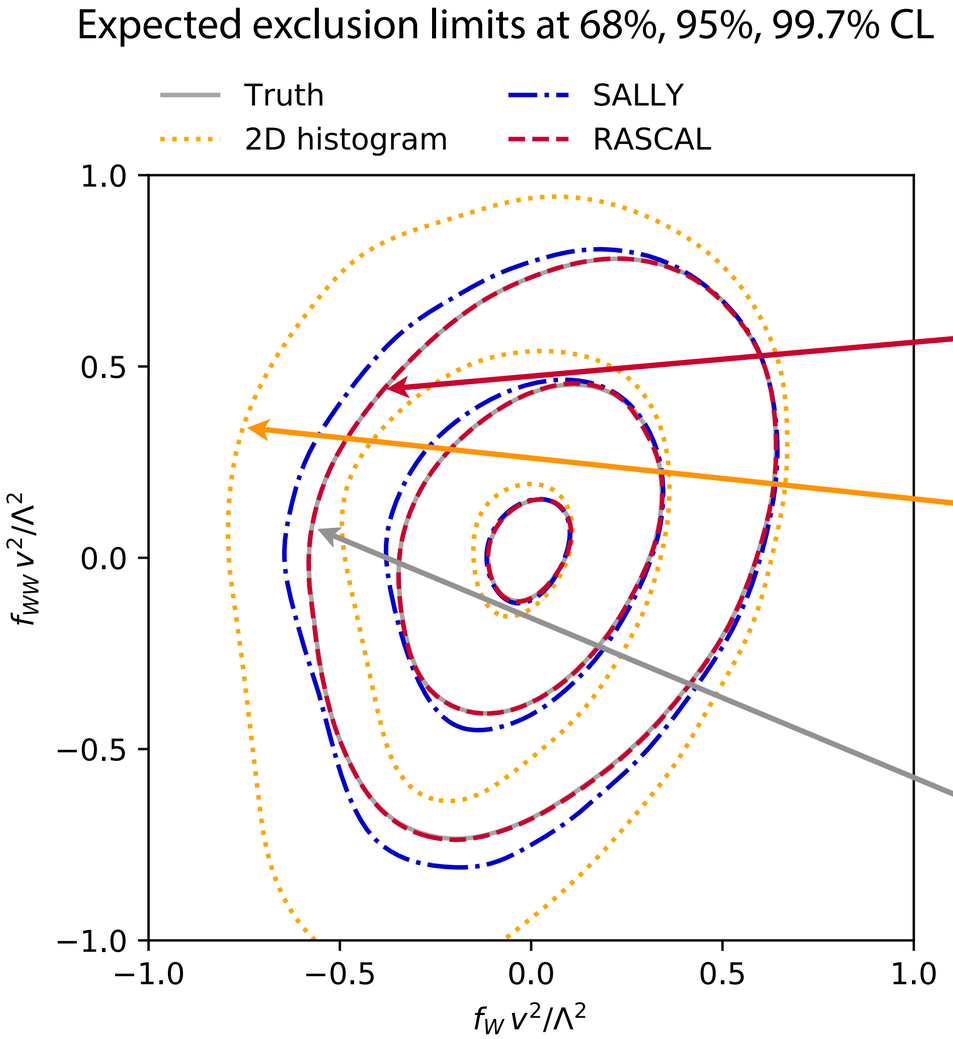




Results are based on 36 observed events, assuming SM

Better sensitivity to new physics





RASCAL enables stronger limits than 2D histogram

Limits from RASCAL indistinguishable from true likelihood (usually we don't have that)

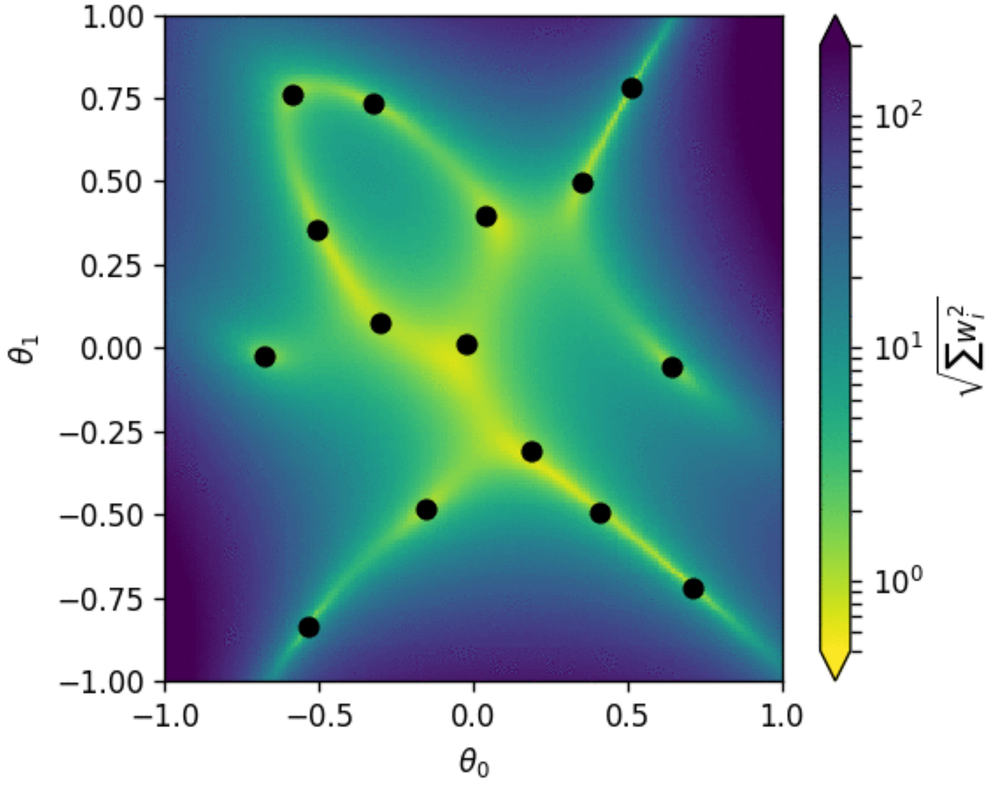
Results are based on 36 observed events, assuming SM

MadMiner

Can I use any of this?

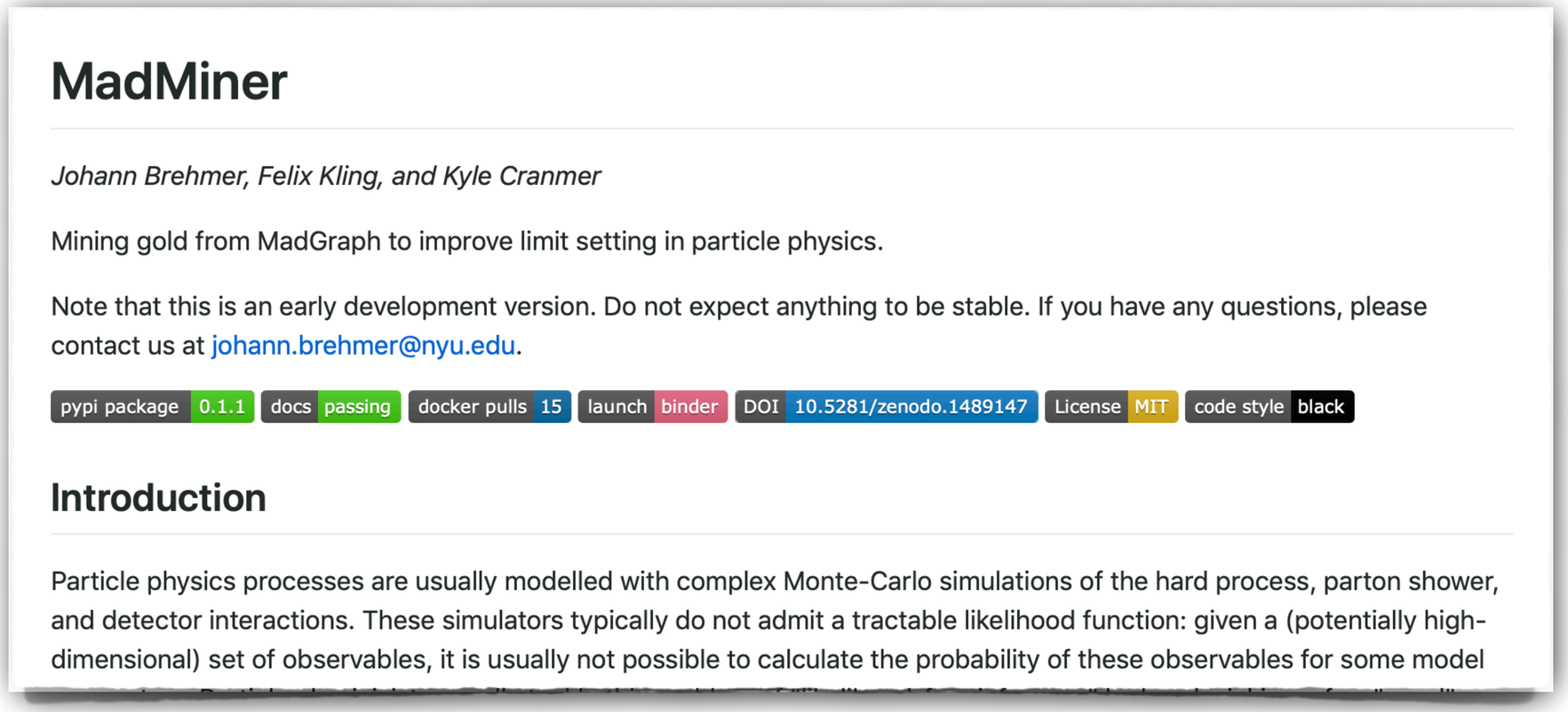
Yes! To make that as painless as possible, we're working on the python package MadMiner:

- "Mining gold" from MadGraph + Pythia + detector simulation
- Parameter morphing (for EFT-like situations)
- Likelihood estimation with RASCAL and friends
- Fisher information calculation
- Systematic uncertainties (currently from PDF variation or scale variation)



We need users / testers / collaborators!

MadMiner resources



MadMiner

Johann Brehmer, Felix Kling, and Kyle Cranmer

Mining gold from MadGraph to improve limit setting in particle physics.

Note that this is an early development version. Do not expect anything to be stable. If you have any questions, please contact us at johann.brehmer@nyu.edu.

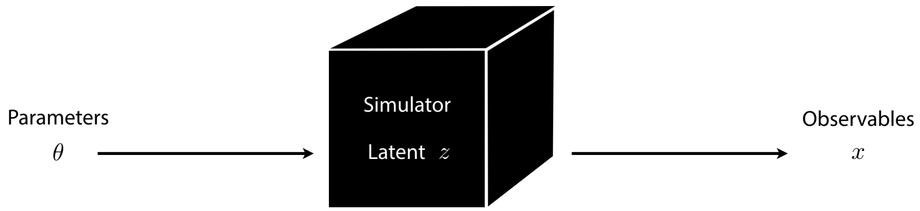
pypi package 0.1.1 | docs passing | docker pulls 15 | launch binder | DOI 10.5281/zenodo.1489147 | License MIT | code style black

Introduction

Particle physics processes are usually modelled with complex Monte-Carlo simulations of the hard process, parton shower, and detector interactions. These simulators typically do not admit a tractable likelihood function: given a (potentially high-dimensional) set of observables, it is usually not possible to calculate the probability of these observables for some model

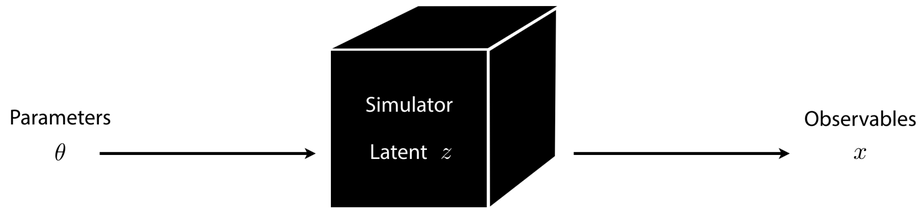
- Current version and tutorials: github.com/johannbrehmer/madminer
- Detailed documentation: madminer.readthedocs.io
- `pip install madminer`

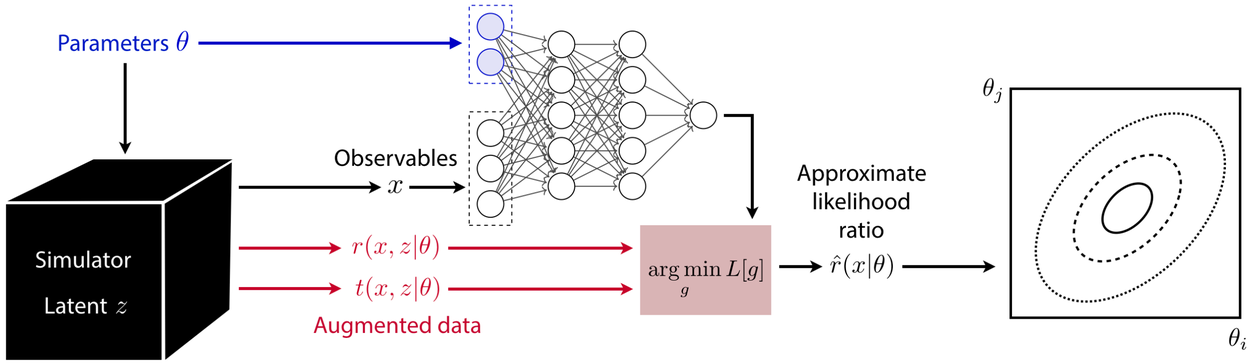
A new approach to simulator-based inference



- Many LHC analyses (and much of modern science) are based on simulations, “likelihood-free”

A new approach to simulator-based inference

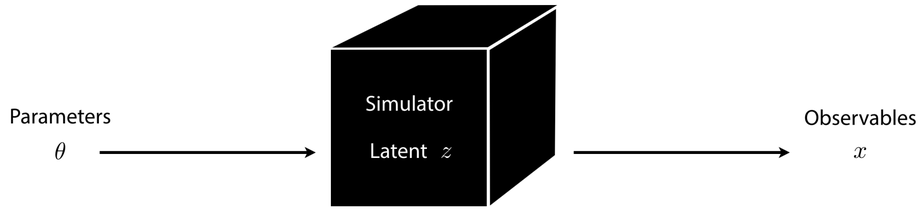


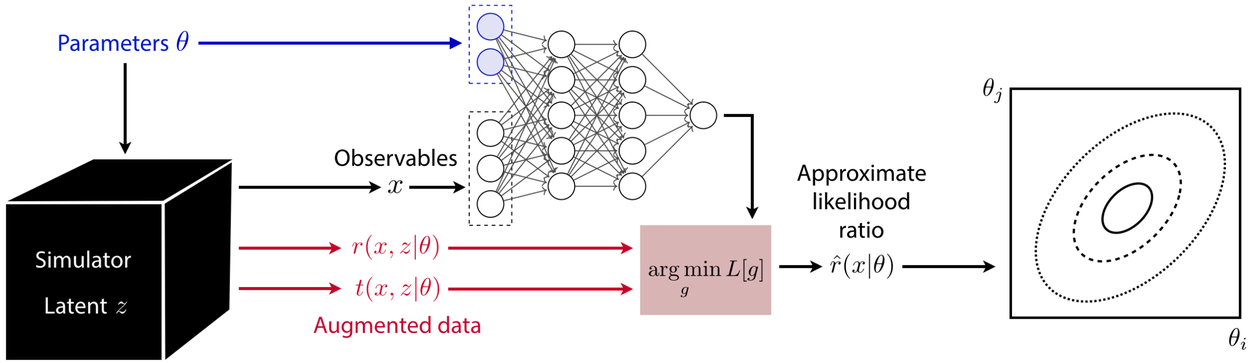


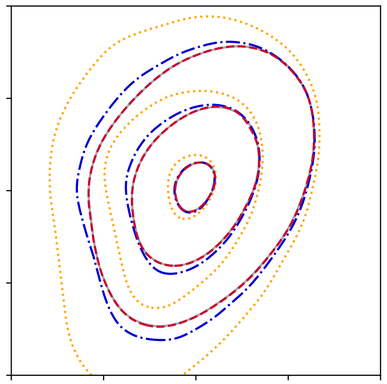
- Many LHC analyses (and much of modern science) are based on simulations, "likelihood-free"

- New multivariate inference techniques: Leverage information in matrix elements + power of machine learning

A new approach to simulator-based inference







- Many LHC analyses (and much of modern science) are based on simulations, "likelihood-free"
- New multivariate inference techniques: Leverage information in matrix elements + power of machine learning
- First application to LHC physics: Stronger EFT constraints with less simulations
- Automatization with new tool MadMiner

References


Kyle Cranmer


Gilles Louppe


Juan Pavez


Markus Stoye


Felix Kling

JB, KC, GL, JP: Constraining Effective Field Theories with Machine Learning [1805.00013]

JB, KC, GL, JP: A Guide to Constraining Effective Field Theories with Machine Learning [1805.00020]

JB, GL, JP, KC: Mining gold from implicit models to improve likelihood-free inference [1805.12244]

MS, **JB**, GL, JP, KC: Likelihood-free inference with an improved cross-entropy estimator [1808.00973]

JB, FK, KC: MadMiner [doi: 10.5281/zenodo.1489147]

Thanks to Kyle and Gilles for inspiring many slides,
to the Moore-Sloan Data Science Environment at NYU for their support,
and to the HKUST IAS for a great workshop!

Bonus material

Variational calculus

$$L[\hat{g}(x)] = \int \mathrm{d}x\, \mathrm{d}z\; p(x,z|\theta)\; |g(x,z) - \hat{g}(x)|^2$$

$$= \int \mathrm{d}x\; \underbrace{\left[\hat{g}^2(x) \int \mathrm{d}z\; p(x,z|\theta) - 2\hat{g}(x) \int \mathrm{d}z\; p(x,z|\theta)\, g(x,z) + \int \mathrm{d}z\; p(x,z|\theta)\, g^2(x,z)\right]}_{F(x)}$$

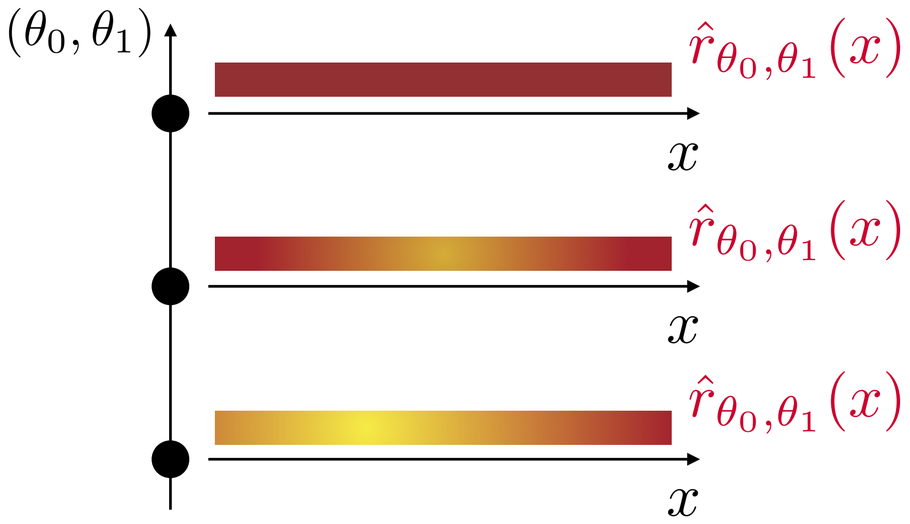
$$0 = \left.\frac{\delta F}{\delta \hat{g}}\right|_{g^*} = 2\hat{g} \underbrace{\int \mathrm{d}z\; p(x,z|\theta)}_{=p(x|\theta)} - 2 \int \mathrm{d}z\; p(x,z|\theta)\, g(x,z)$$

$$g^*(x) = \frac{1}{p(x|\theta)} \int \mathrm{d}z\; p(x,z|\theta)\, g(x,z)$$

Two types of likelihood ratio estimators

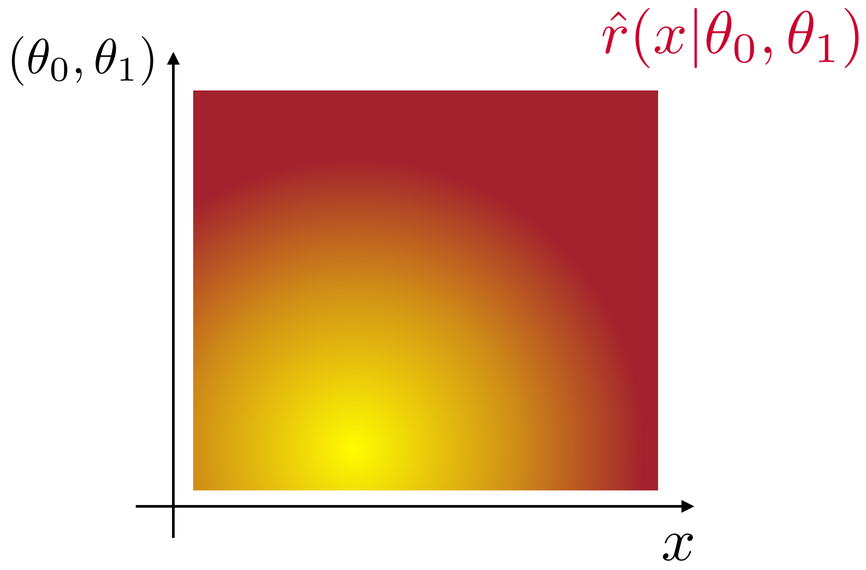
A) Point by point:

- first, define grid of parameter points $\{(\theta_0, \theta_1)\}$
- for each combination (θ_0, θ_1), create separate estimator $\hat{r}_{\theta_0,\theta_1}(x)$
- final results can be interpolated between grid points

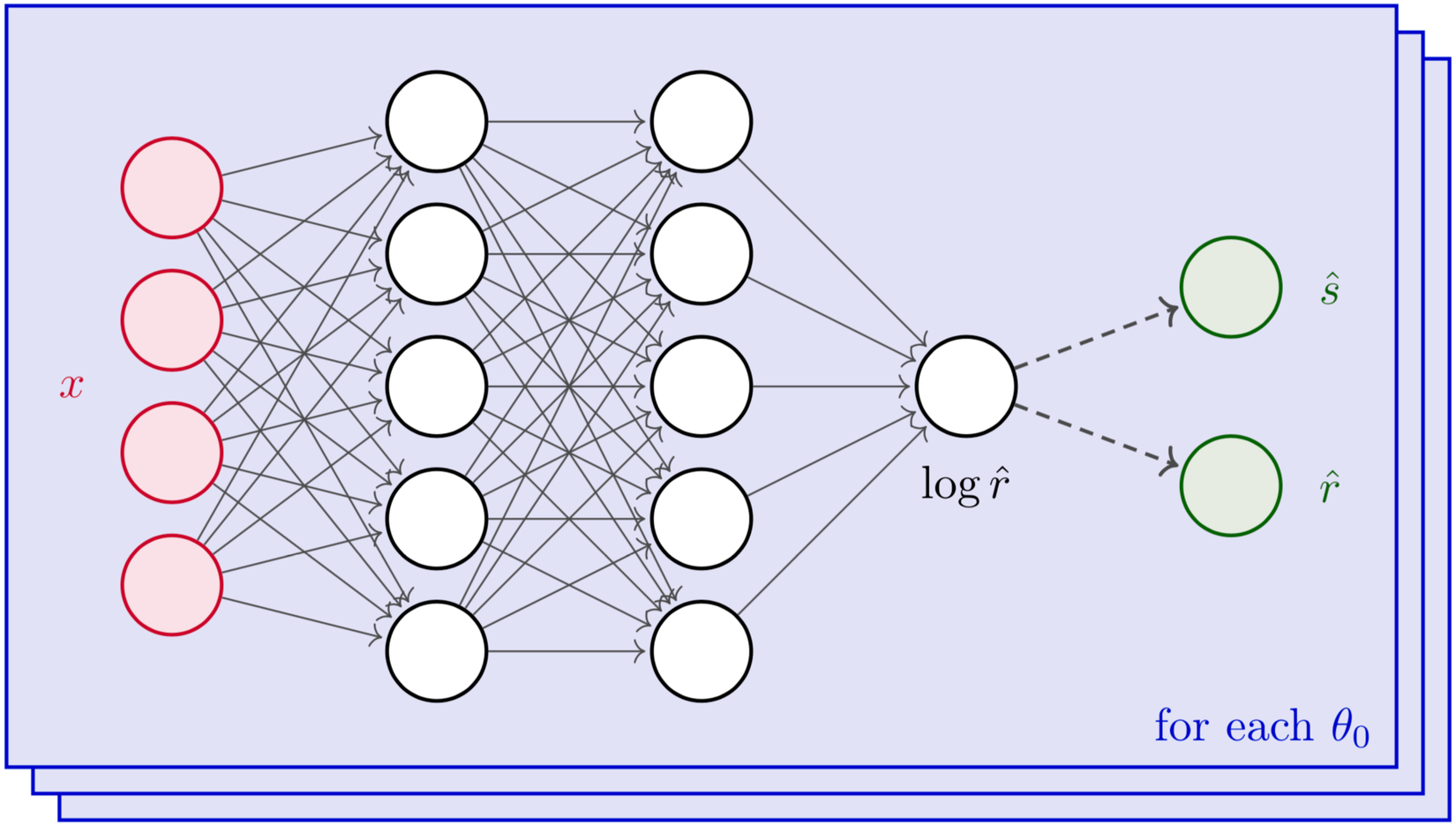


B) Parameterized: [P. Baldi et al. 1506.02169]

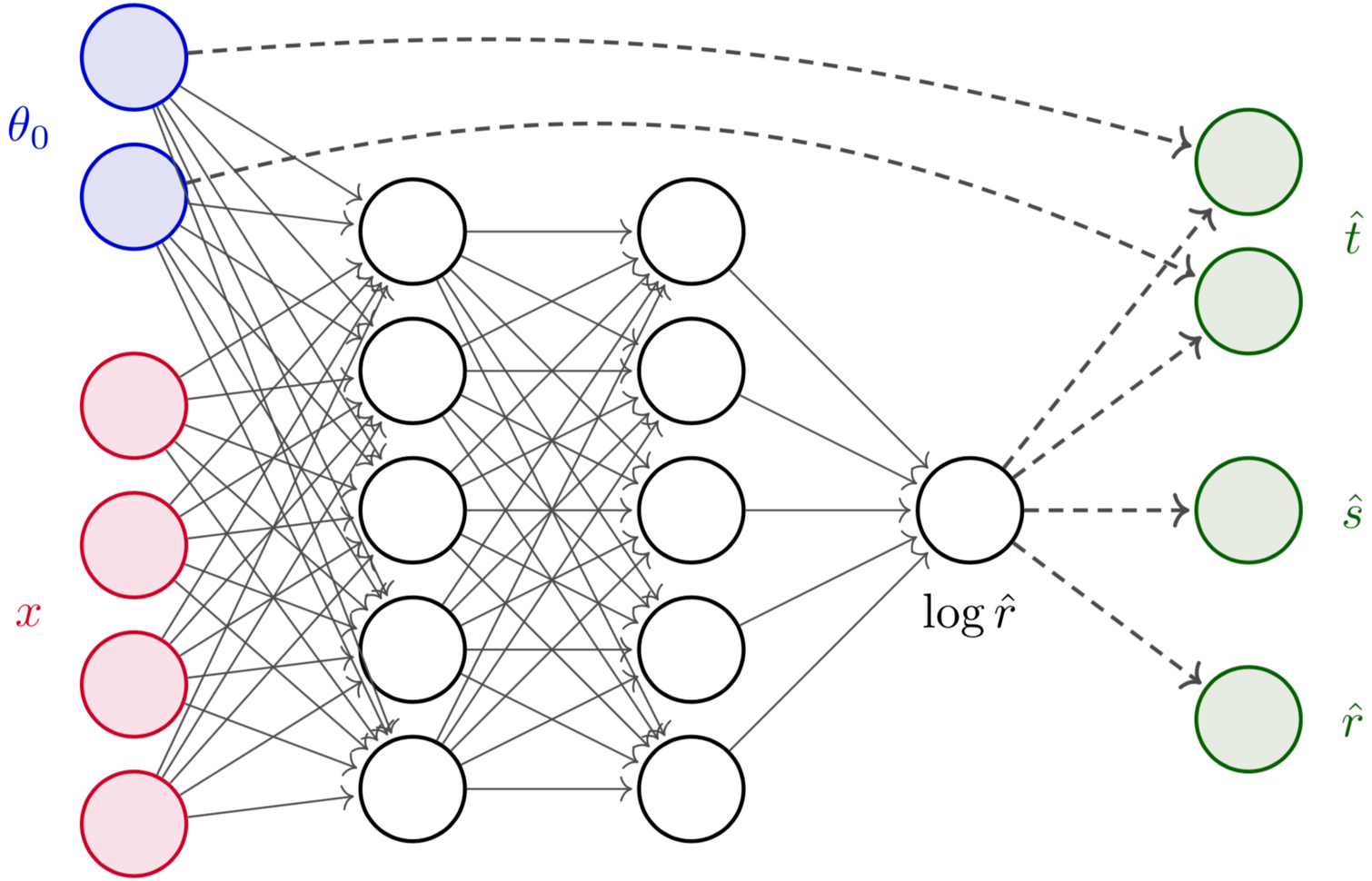
- create one estimator $\hat{r}(x|\theta_0, \theta_1)$ that is a function of θ_0 and θ_1
- no further interpolation necessary
- "borrows information" from close points



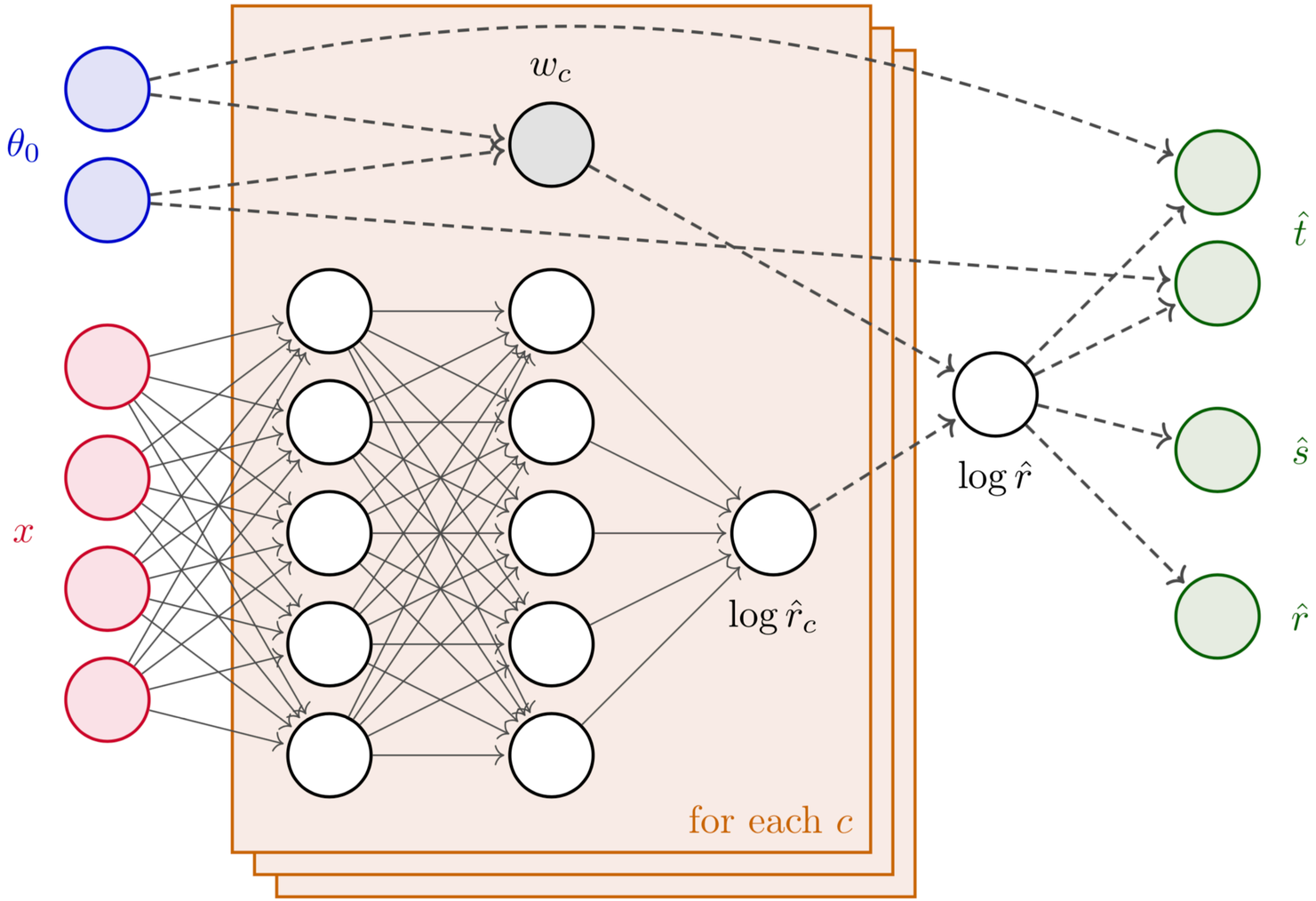
Point by point



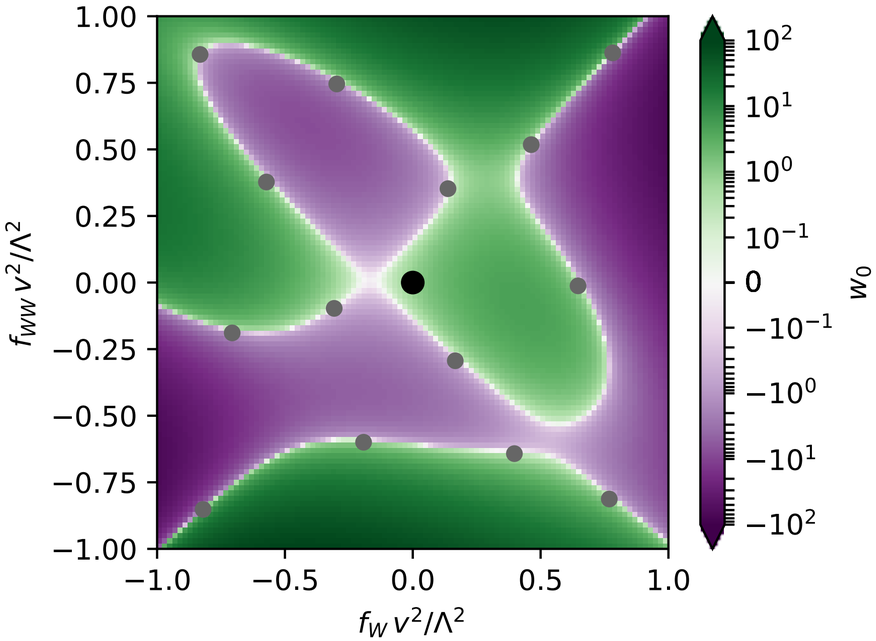
(Agnostic) parameterized estimators



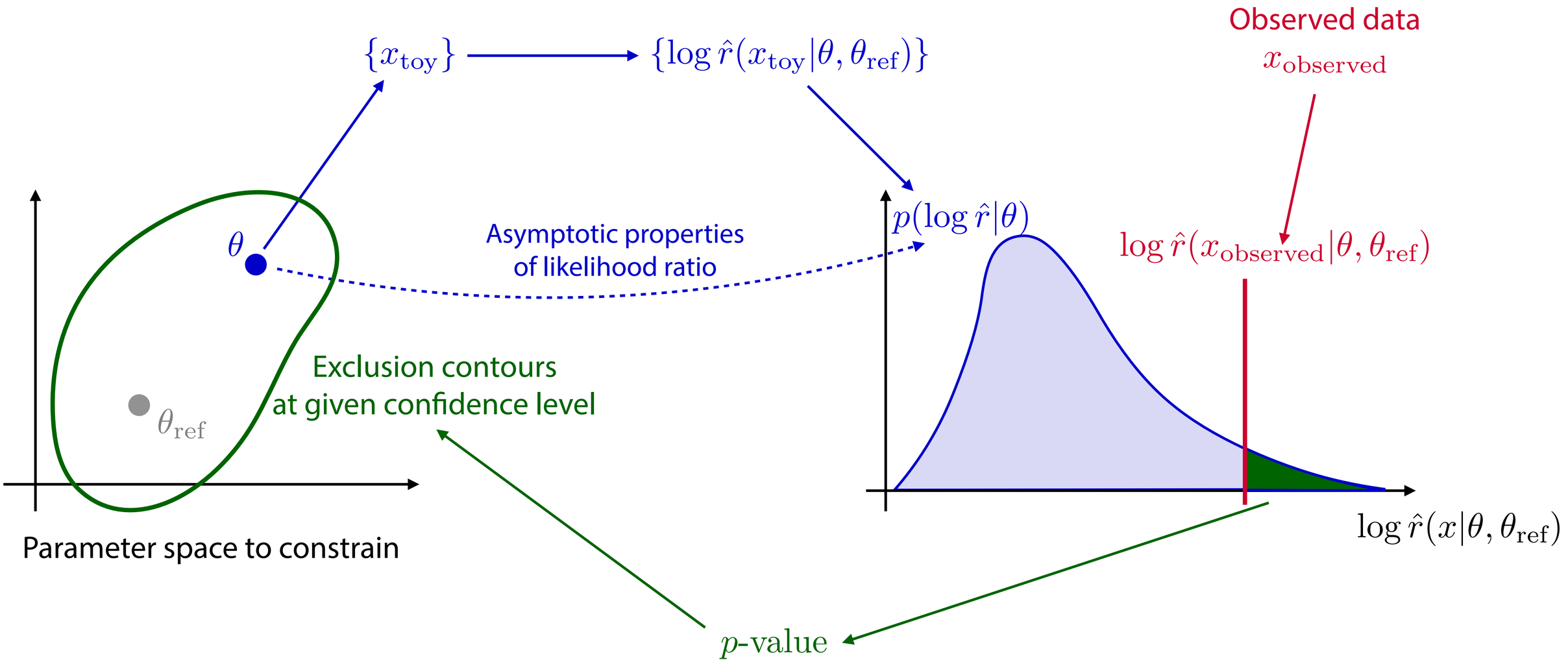
Morphing-aware parameterized estimators



$$\hat{r}(x|\theta_0,\theta_1) = \sum_c w_c(\theta_0)\,\hat{r}_c(x)$$



Limit setting (frequentist)



Detector effects