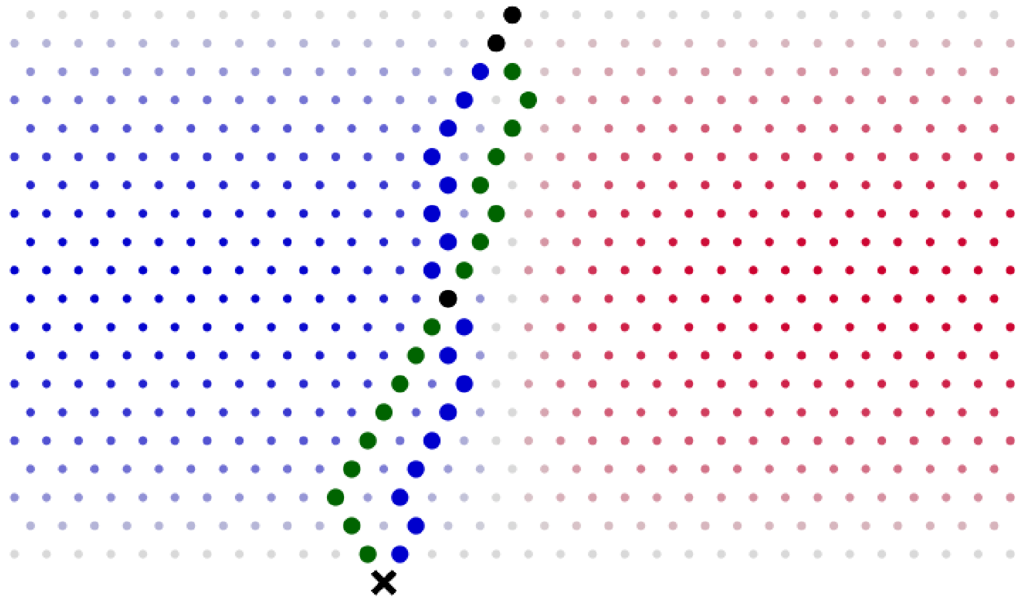
Likelihood-free inference in Physical Sciences

November 19, Data Institute, Univ. Grenoble Alpes

Gilles Louppe
g.louppe@uliege.be


@physicsfun



The probability of ending in bin x corresponds to the total probability of all the paths z from start to x.

$$p(x|\theta) = \int p(x, z|\theta) dz = \binom{n}{x} \theta^x (1-\theta)^{n-x}$$

What if we shift or remove some of the pins?

The probability of ending in bin x still corresponds to the cumulative probability of all the paths from start to x:

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

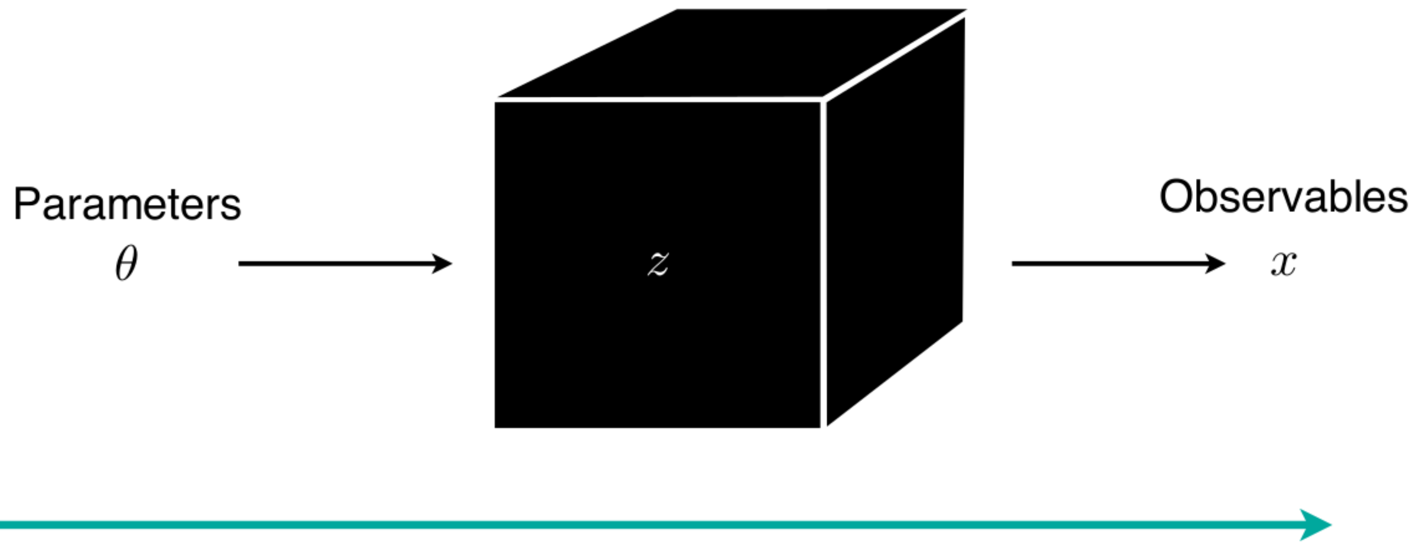
- But this integral can no longer be simplified analytically!
- As n grows larger, evaluating $p(x|\theta)$ becomes intractable since the number of paths grows combinatorially.
- Generating observations remains easy: drop the balls.

Since $p(x|\theta)$ cannot be evaluated, does this mean inference is not possible?

The Galton board is a metaphore of simulation-based science:

Galton board device	$\rightarrow$	Computer simulation
Parameters θ	$\rightarrow$	Model parameters θ
Buckets x	$\rightarrow$	Observables x
Random paths z	$\rightarrow$	Latent variables z (stochastic execution traces through simulator)

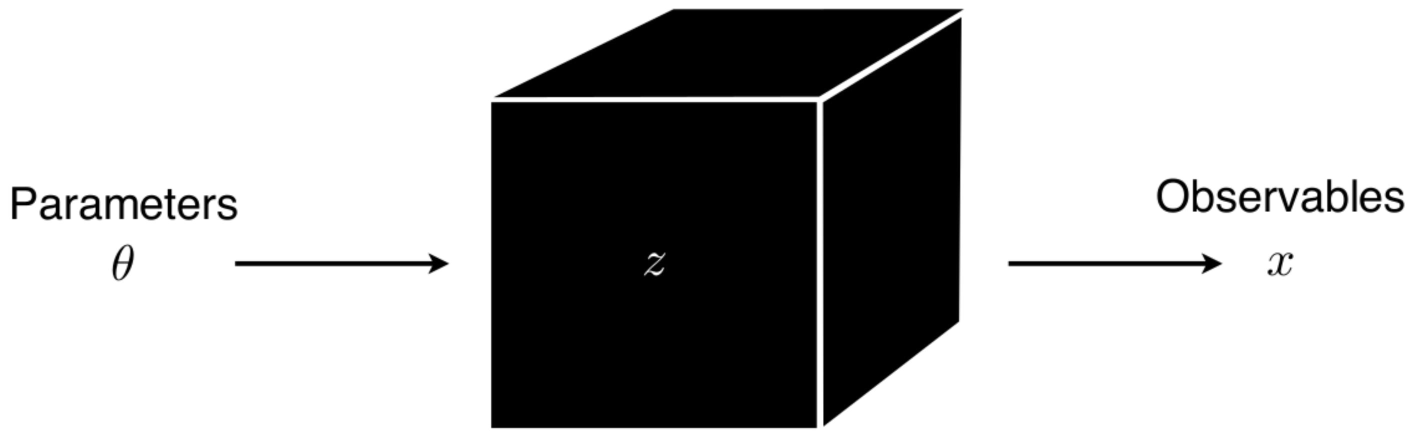
Inference in this context requires likelihood-free algorithms.



Prediction (simulation):

- Well-understood mechanistic model
- Simulator can generate samples

Credits: Johann Brehmer



Prediction (simulation):

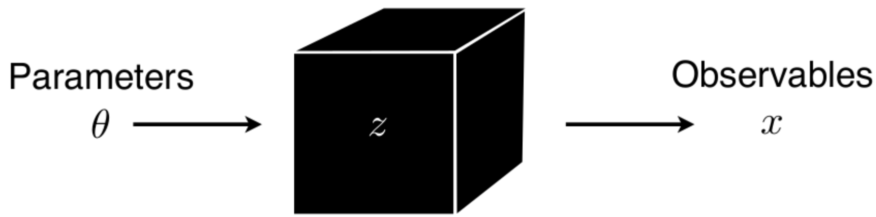
- Well-understood mechanistic model
- Simulator can generate samples

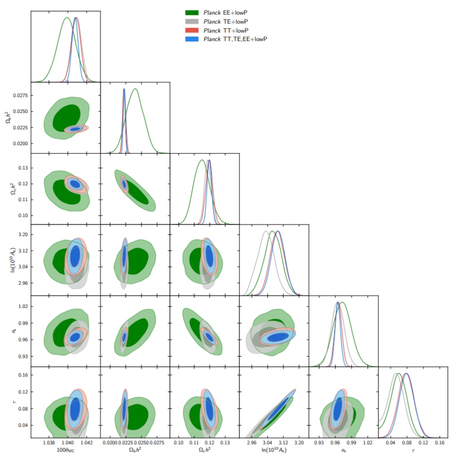
Inference:

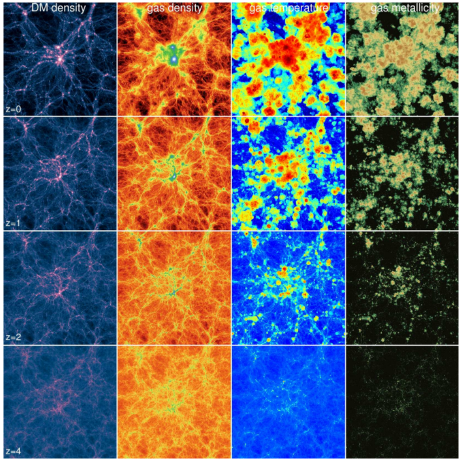
- Likelihood function $p(x|\theta)$ is intractable
- Goal: estimator $\hat{p}(x|\theta)$

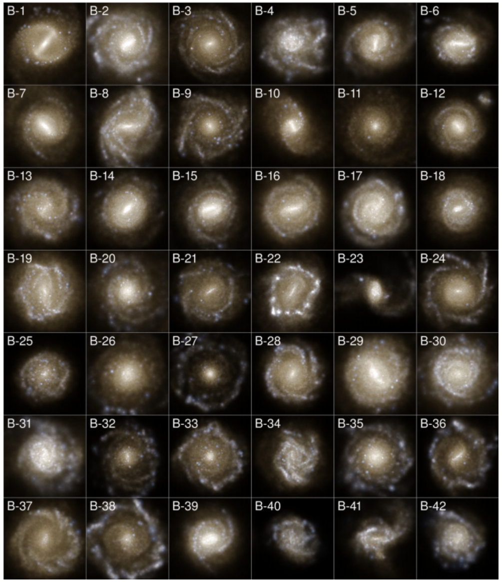
Applications

Cosmological N-body simulations



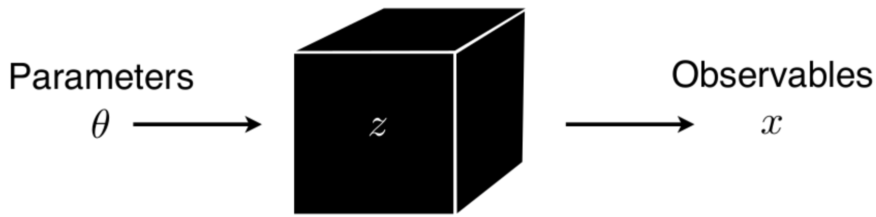


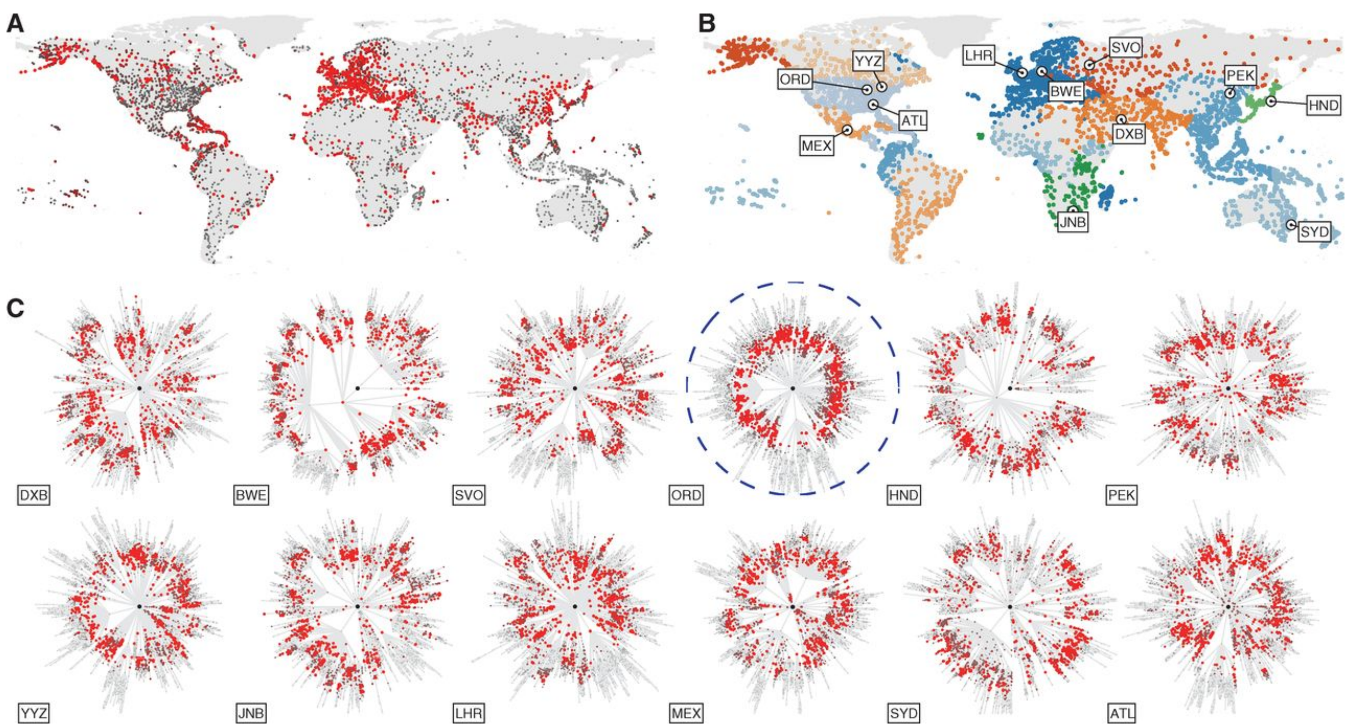




Refs: Planck Collaboration, 2015 (arXiv:1502.01589); Vogelsberger et al, 2014 (arXiv:1405.2921)

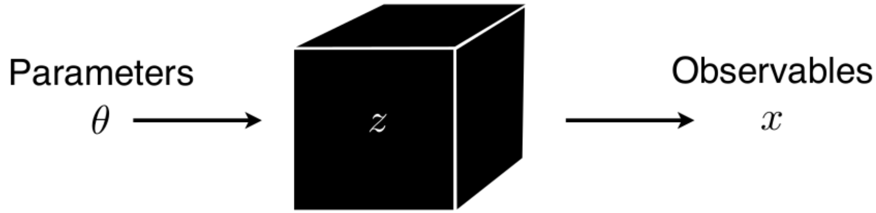
Epidemiology

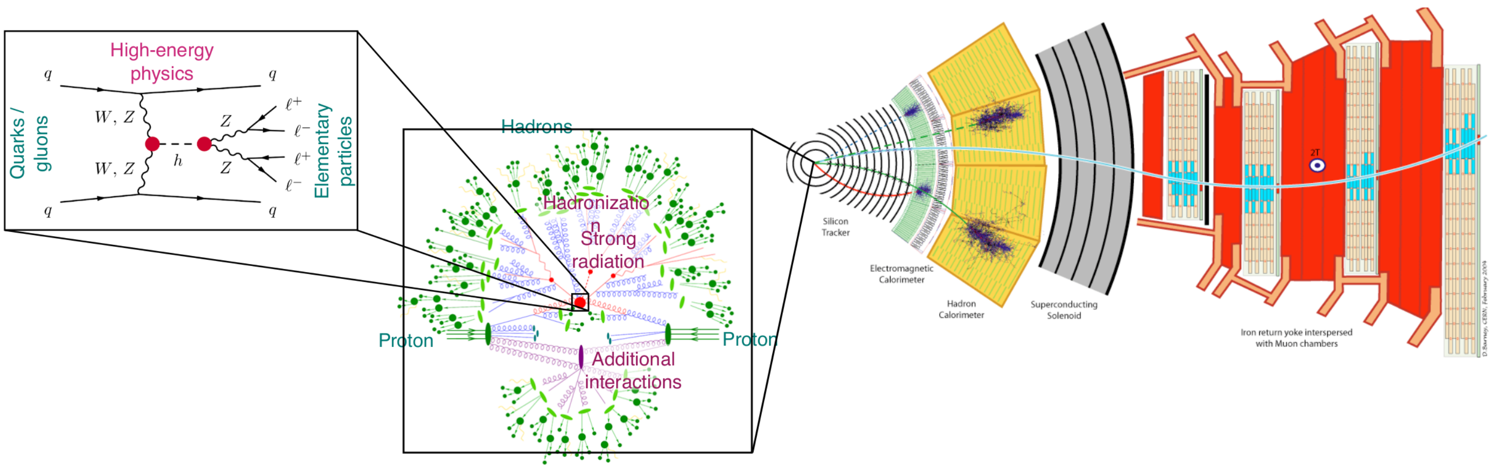




Refs: Brockmann and Helbing, 2013 (doi:10.1126/science.1245200)

Particle physics





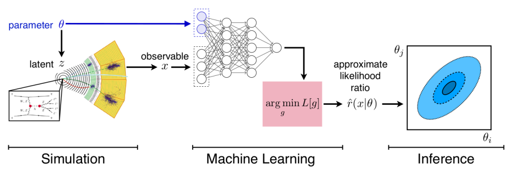
The Galton board of particle physics

Likelihood-free inference algorithms

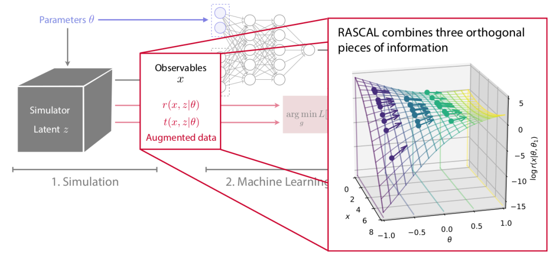
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

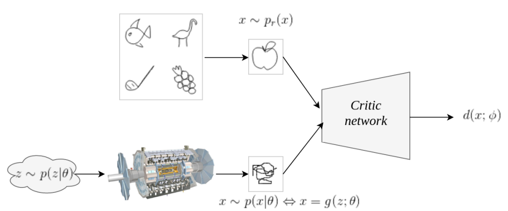


Histograms of observables
Neural density (ratio) estimation

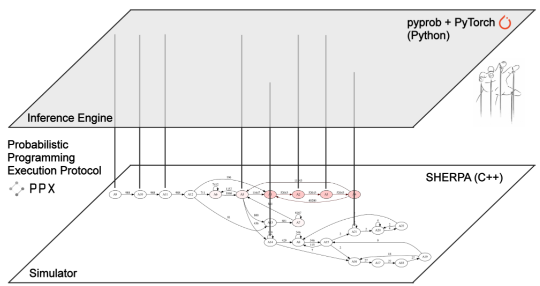


Mining gold from implicit models

Learn to control the simulator



Adversarial variational optimization

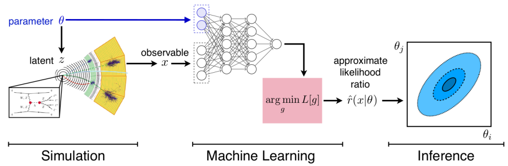


Probabilistic programming

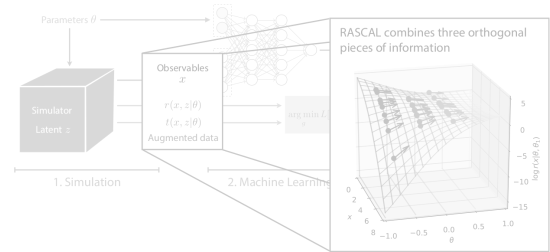
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

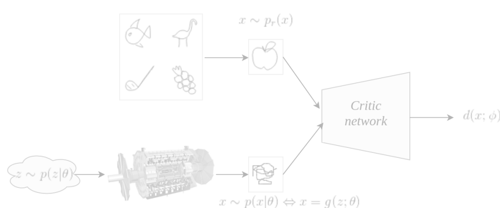


Histograms of observables
Neural density (ratio) estimation

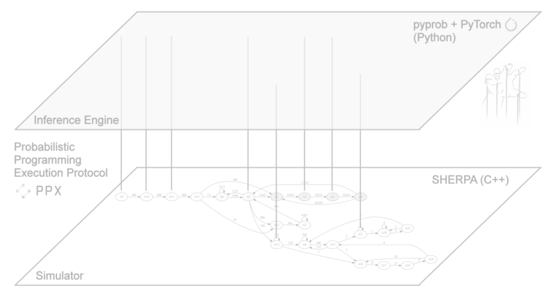


Mining gold from implicit models

Learn to control the simulator



Adversarial variational optimization



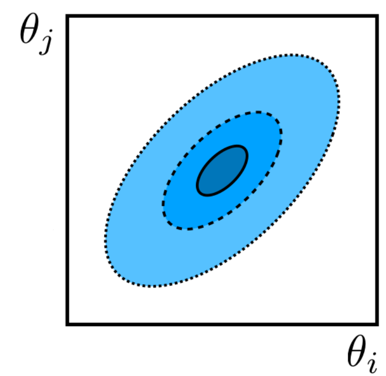
Probabilistic programming

The physicist's way

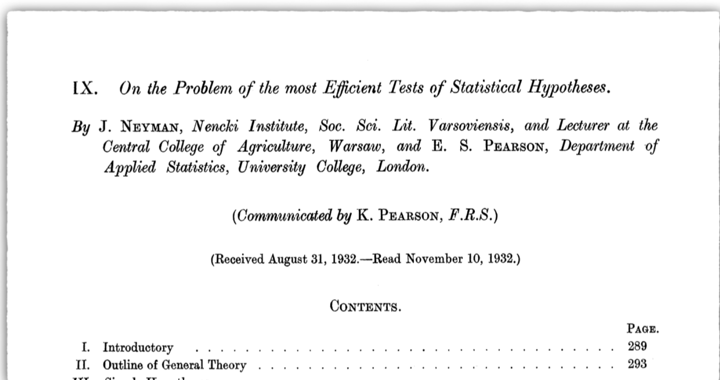
The Neyman-Pearson lemma states that the likelihood ratio

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

is the most powerful test statistic to discriminate between a null hypothesis θ_0 and an alternative θ_1.



How does one compute this ratio in the likelihood-free context?



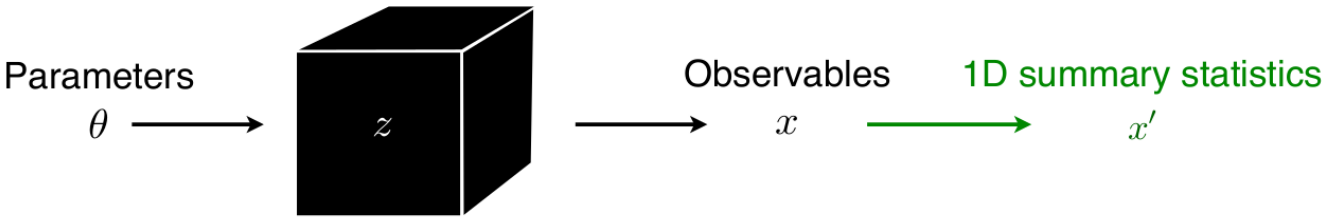
IX. *On the Problem of the most Efficient Tests of Statistical Hypotheses.*

By J. NEYMAN, *Nencki Institute, Soc. Sci. Lit. Varsoviensis, and Lecturer at the Central College of Agriculture, Warsaw, and* E. S. PEARSON, *Department of Applied Statistics, University College, London.*

(*Communicated by* K. PEARSON, *F.R.S.*)

(Received August 31, 1932.—Read November 10, 1932.)

CONTENTS.



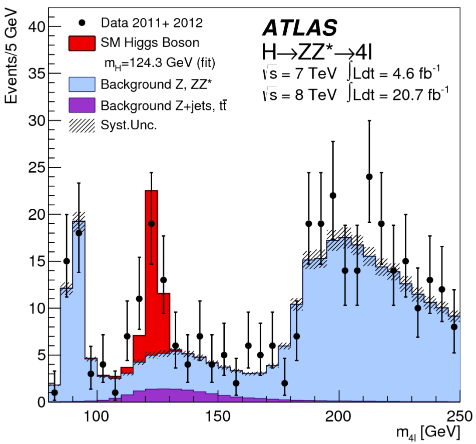
Define a projection function $s : \mathcal{X} \rightarrow \mathbb{R}$ mapping observables x to a summary statistics $x' = s(x)$.

Then, approximate the likelihood $p(x|\theta)$ as

$$p(x|\theta) \approx \hat{p}(x|\theta) = p(x'|\theta).$$

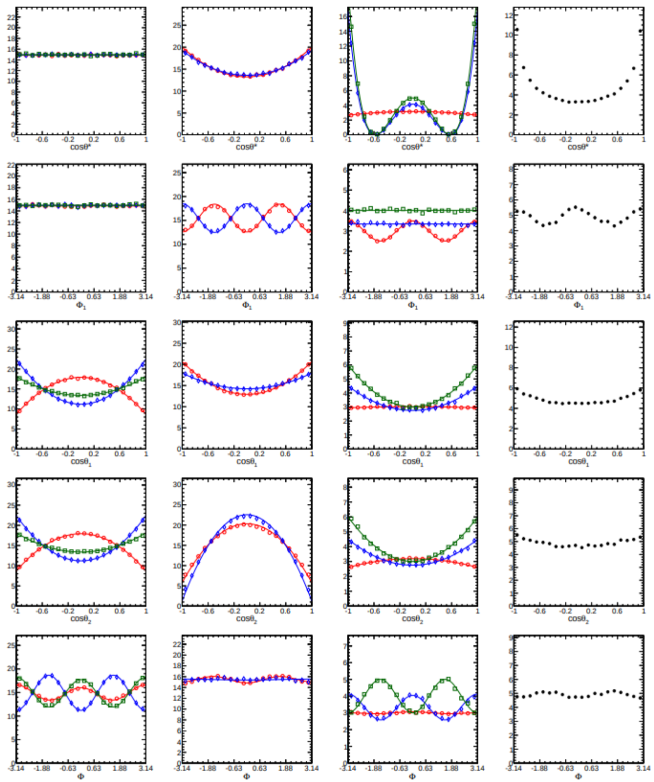
From this it comes

$$\frac{p(x|\theta_0)}{p(x|\theta_1)} \approx \frac{\hat{p}(x|\theta_0)}{\hat{p}(x|\theta_1)} = \hat{r}(x|\theta_0, \theta_1).$$



This methodology has worked great for physicists for the last 20-30 years, but ...

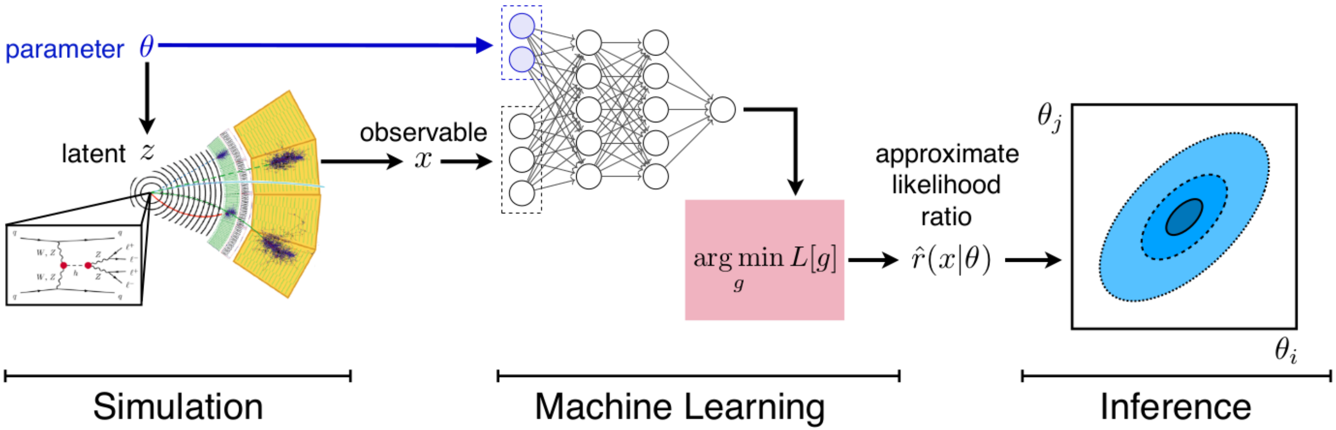
- Choosing the projection s is difficult and problem-dependent.
- Often there is no single good variable: compressing to any x' loses information.
- Ideally: analyse high-dimensional x', including all correlations.



Unfortunately, filling high-dimensional histograms is not tractable.

Who you gonna call? Machine learning!

CARL



Key insights

- The likelihood ratio is often sufficient for inference.
- Evaluating the likelihood ratio does not require evaluating the individual likelihoods.
- Supervised learning indirectly estimates likelihood ratios.

Refs: Cranmer et al, 2016 (arXiv:1506.02169)

Supervised learning provides a way to automatically construct s:

- Let us consider a binary classifier $\hat{s}$ (e.g., a neural network) trained to distinguish $x \sim p(x|\theta_0)$ from $x \sim p(x|\theta_1)$.
- $\hat{s}$ is trained by minimizing the cross-entropy loss

$$\begin{aligned} L_{XE}[\hat{s}] = -\mathbb{E}_{p(x|\theta)\pi(\theta)}[&1(\theta = \theta_0) \log \hat{s}(x) + \\ &1(\theta = \theta_1) \log(1 - \hat{s}(x))] \end{aligned}$$

The solution $\hat{s}$ found after training approximates the optimal classifier

$$\hat{s}(x) \approx s^*(x) = \frac{p(x|\theta_1)}{p(x|\theta_0) + p(x|\theta_1)}.$$

Therefore,

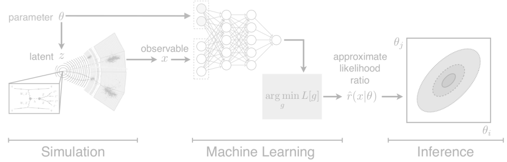
$$r(x|\theta_0, \theta_1) \approx \hat{r}(x|\theta_0, \theta_1) = \frac{1 - \hat{s}(x)}{\hat{s}(x)}$$

That is, supervised classification is equivalent to likelihood ratio estimation and can therefore be used for MLE inference.

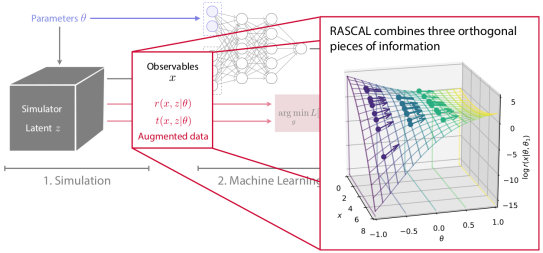
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

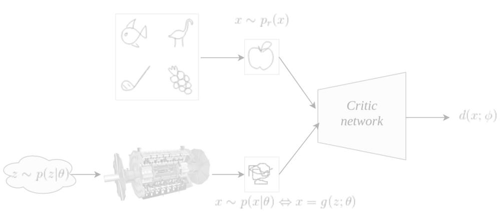


Histograms of observables
Neural density (ratio) estimation

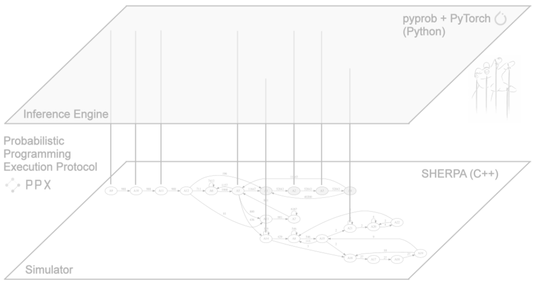


Mining gold from implicit models

Learn to control the simulator

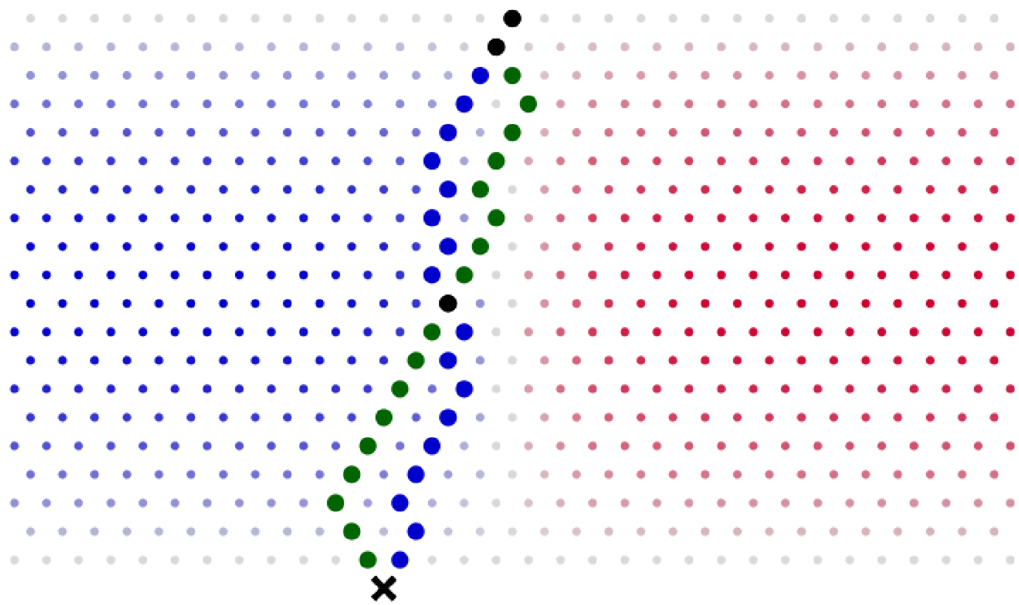


Adversarial variational optimization



Probabilistic programming

Mining gold from simulators



$p(x|\theta)$ is usually intractable.

What about $p(x, z|\theta)$?

As the trajectory $z_1, ..., z_T$ and the observable x are emitted, it is often possible:

- to calculate the joint likelihood $p(x, z|\theta)$;
- to calculate the joint likelihood ratio $r(x, z|\theta_0, \theta_1)$;
- to calculate the joint score $t(x, z|\theta_0) = \nabla_\theta \log p(x, z|\theta)\big|_{\theta_0}$.

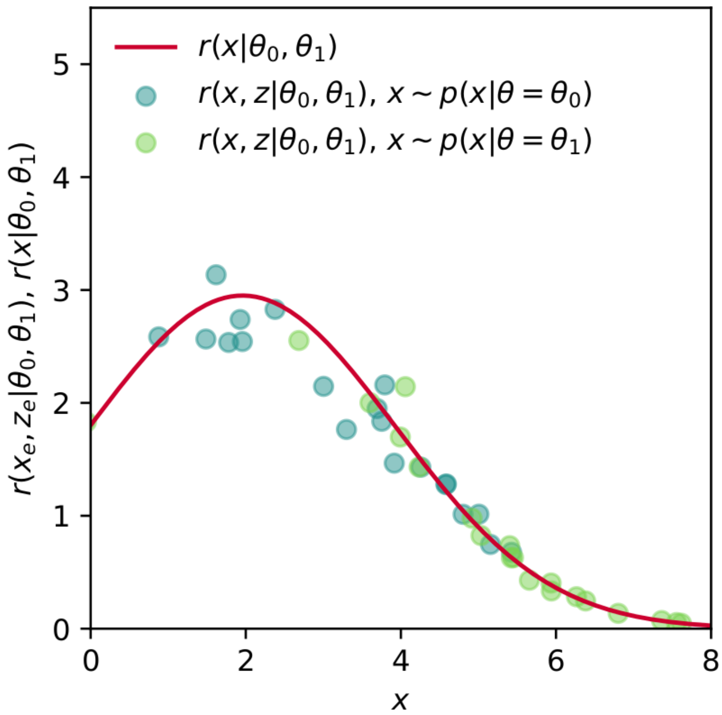
We call this process mining gold from your simulator!

Observe that the joint likelihood ratios

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

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

Can we use them to approximate $r(x|\theta_0, \theta_1)$?



Key insights

Consider the squared error of a function $\hat{g}(x)$ that only depends on x, but is trying to approximate a function $g(x, z)$ that also depends on the latent z:

$$L_{MSE} = \mathbb{E}_{p(x,z|\theta)}\left[\left(g(x,z) - \hat{g}(x)\right)^2\right].$$

Via calculus of variations, we find that the function $g^*(x)$ that extremizes $L_{MSE}[g]$ is given by

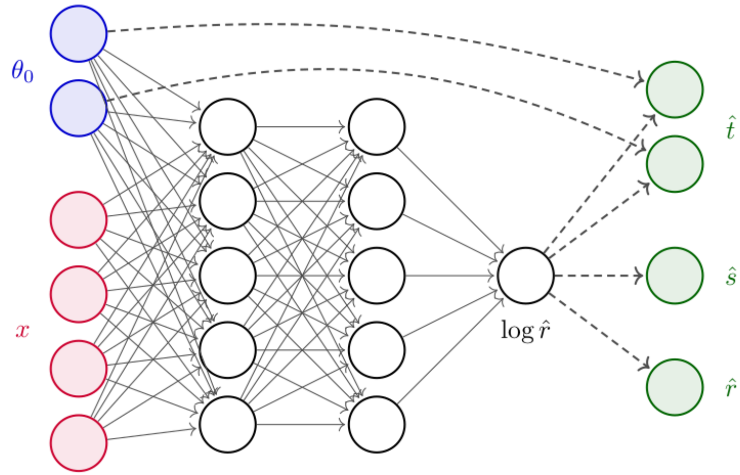
$$\begin{aligned} g^*(x) &= \frac{1}{p(x|\theta)} \int p(x,z|\theta) g(x,z) dz \\ &= \mathbb{E}_{p(z|x,\theta)}\left[g(x,z)\right] \end{aligned}$$

Therefore, by identifying the $g(x, z)$ with the joint likelihood ratio $r(x, z|\theta_0, \theta_1)$ and θ with θ_1, we define

$$L_r = \mathbb{E}_{p(x,z|\theta_1)} \left[\left(r(x, z|\theta_0, \theta_1) - \hat{r}(x) \right)^2 \right],$$

which is minimized by

$$\begin{aligned} r^*(x) &= \frac{1}{p(x|\theta_1)} \int p(x, z|\theta_1) \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} dz \\ &= \frac{p(x|\theta_0)}{p(x|\theta_1)} \\ &= r(x|\theta_0, \theta_1). \end{aligned}$$



How does one find r^*?

$$r^*(x|\theta_0, \theta_1) = \arg\min_{\hat{r}} L_r[\hat{r}]$$

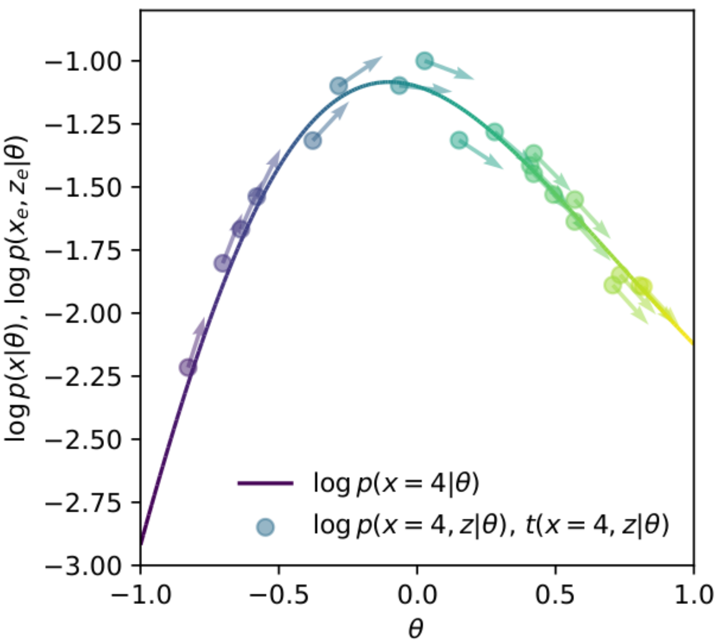
Minimizing functionals is exactly what machine learning does. In our case,

- $\hat{r}$ are neural networks (or the parameters thereof);
- L_r is the loss function;
- minimization is carried out using stochastic gradient descent from the data extracted from the simulator.

Similarly, we can mine the simulator to extract the joint score

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

which indicates how much more or less likely x, z would be if one changed θ_0.



Using the same trick, by identifying $g(x, z)$ with the joint score $t(x, z|\theta_0)$ and θ with θ_0, we define

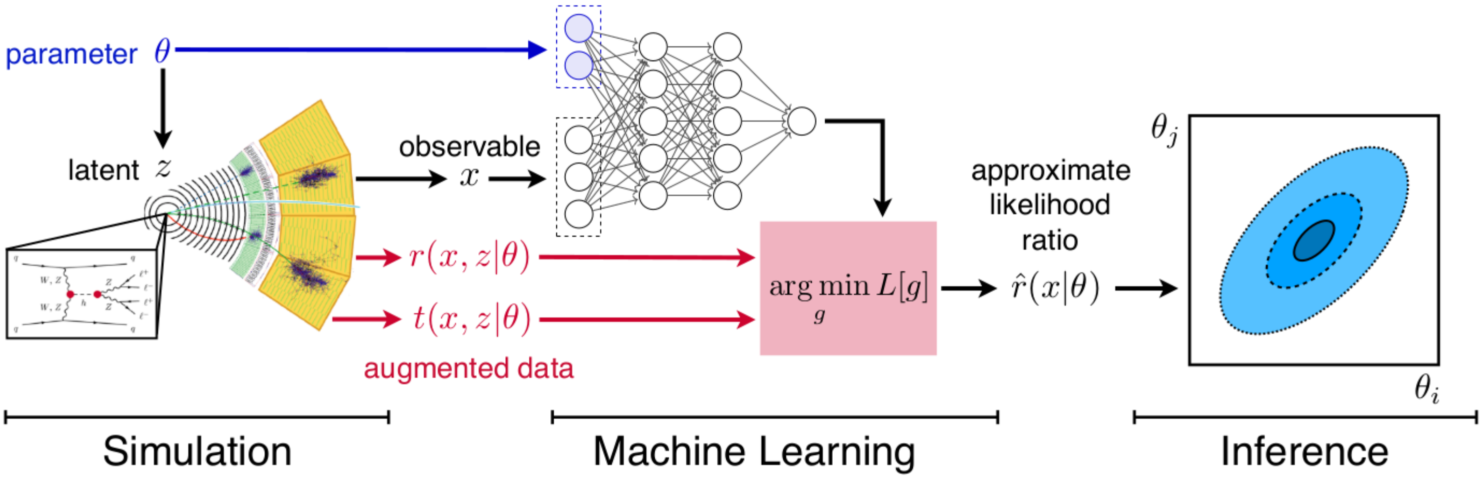
$$L_t = \mathbb{E}_{p(x,z|\theta_0)} \left[(t(x, z|\theta_0) - \hat{t}\,(x))^2 \right],$$

which is minimized by

$$\begin{aligned} t^*(x) &= \frac{1}{p(x|\theta_0)} \int p(x, z|\theta_0)(\nabla_\theta \log p(x, z|\theta)\big|_{\theta_0})dz \\ &= \frac{1}{p(x|\theta_0)} \int p(x, z|\theta_0) \frac{\nabla_\theta p(x, z|\theta)\big|_{\theta_0}}{p(x, z|\theta_0)} dz \\ &= \frac{\nabla_\theta p(x|\theta)\big|_{\theta_0}}{p(x|\theta_0)} \\ &= t(x|\theta_0). \end{aligned}$$

Rascal

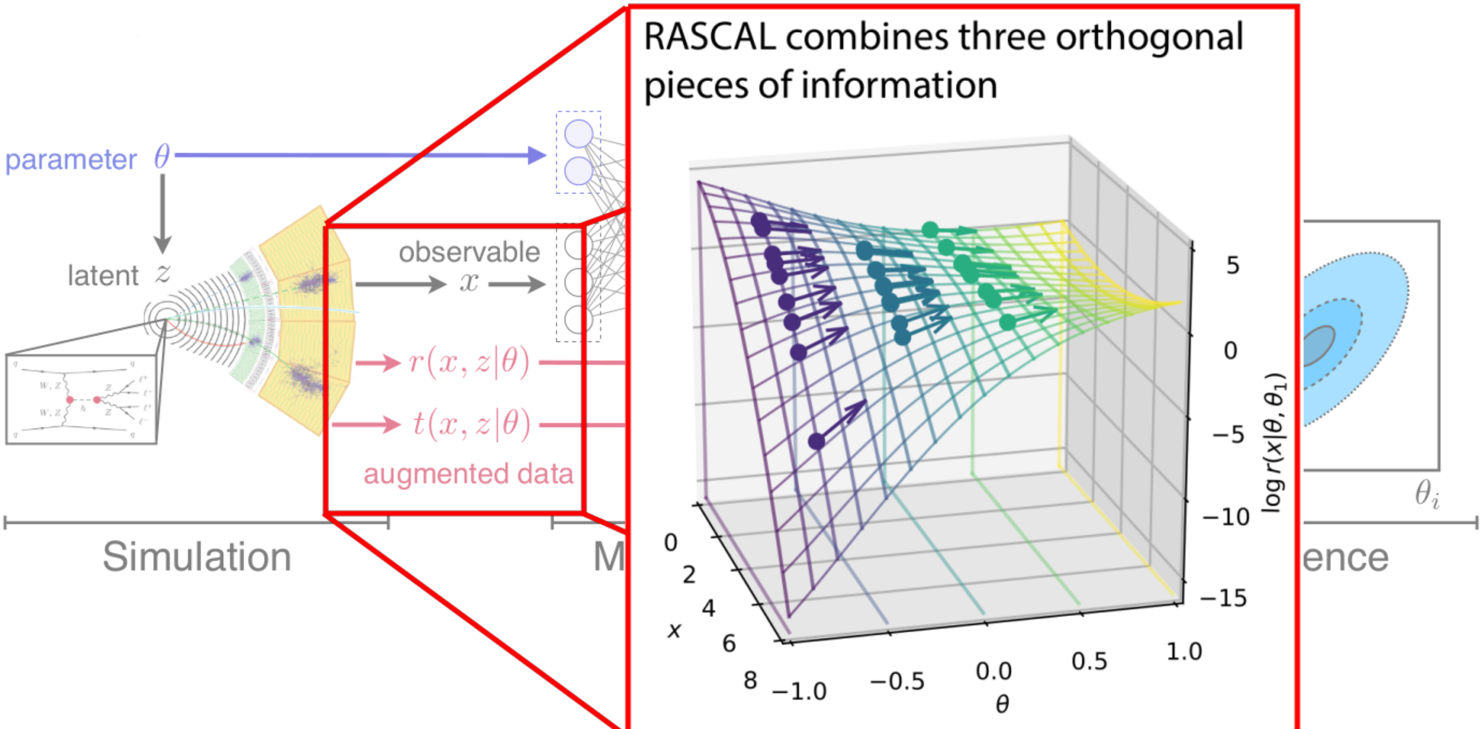
$$L_{RASCAL} = L_r + L_t$$



Refs: Brehmer et al, 2018 (arXiv:1805.12244)

RASCAL

$$L_{RASCAL} = L_r + L_t$$



LHC processes

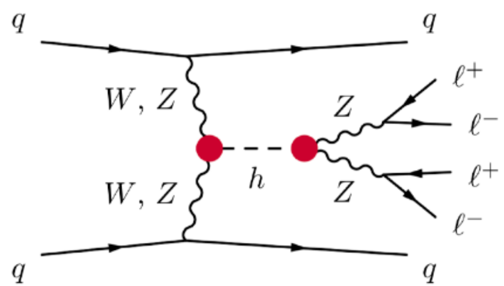
Latent variables

Parameters of interest

Parton-level momenta

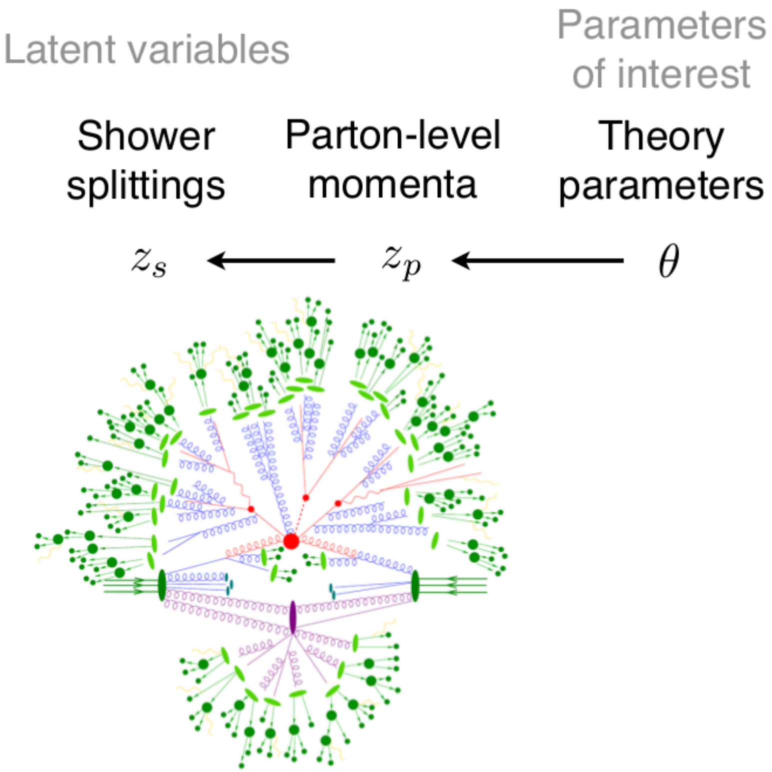
Theory parameters

$z_p \longleftarrow \theta$



Credits: Johann Brehmer

LHC processes



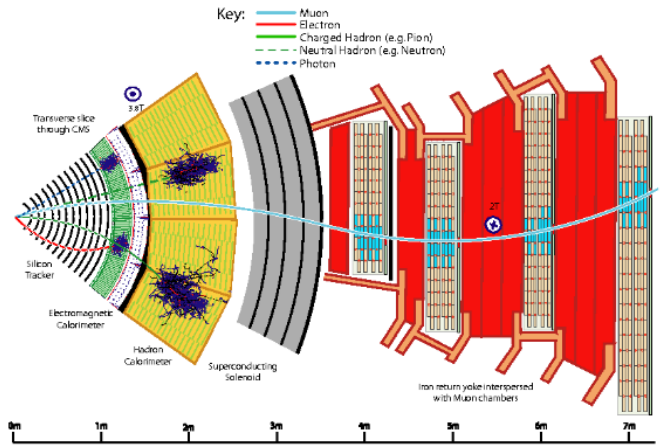
LHC processes

Latent variables

Parameters of interest

Detector interactions	Shower splittings	Parton-level momenta	Theory parameters

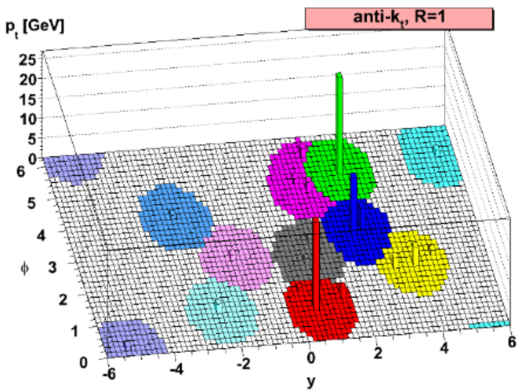
$$z_d \longleftarrow z_s \longleftarrow z_p \longleftarrow \theta$$



LHC processes

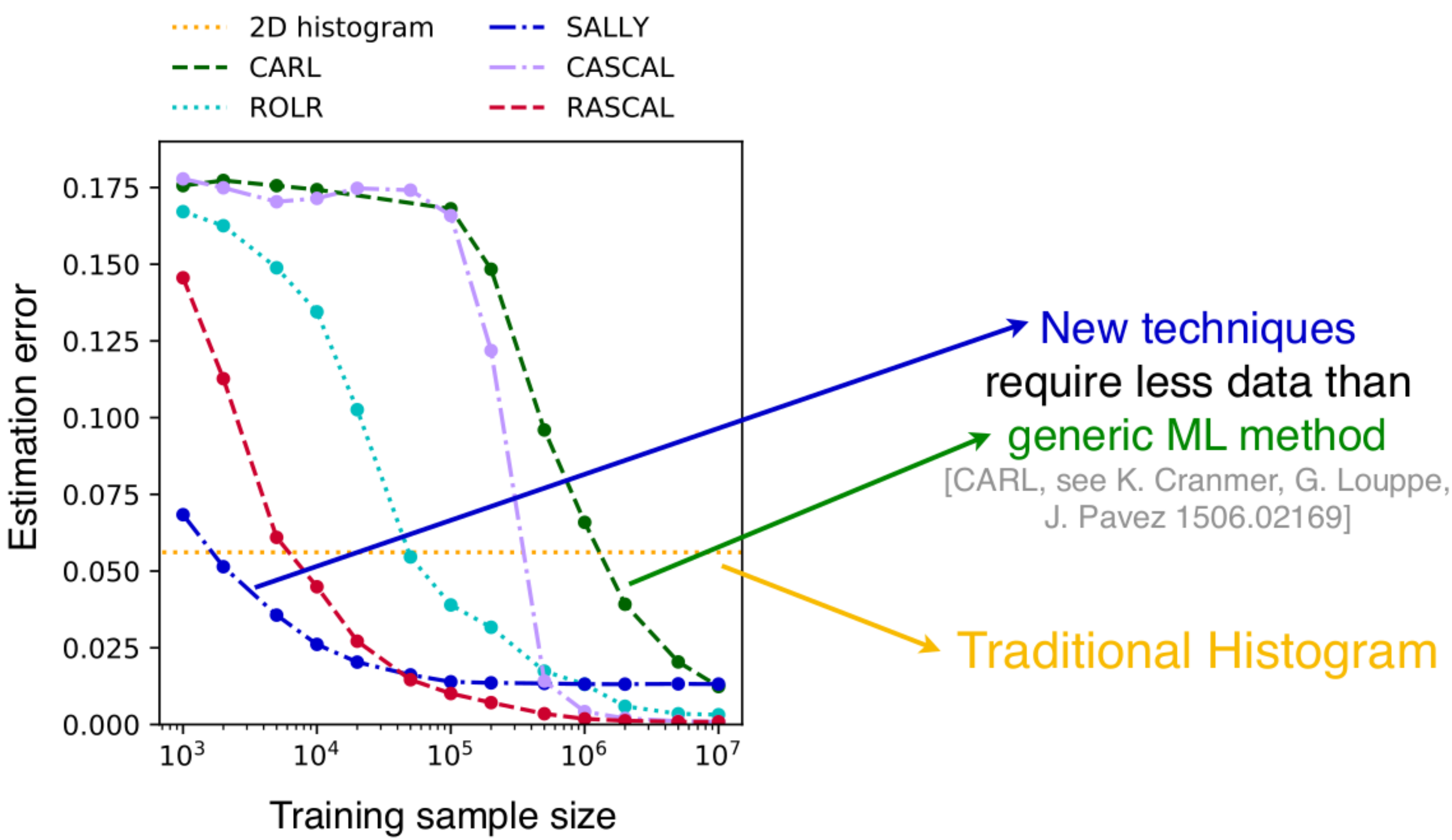
Features		Latent variables		Parameters of interest
Observables	Detector interactions	Shower splittings	Parton-level momenta	Theory parameters
x	z_d	z_s	z_p	θ

$$x \longleftarrow z_d \longleftarrow z_s \longleftarrow z_p \longleftarrow \theta$$



[Image source: M. Cacciari, G. Salam, G. Soyez 0802.1189]

Increased data efficiency

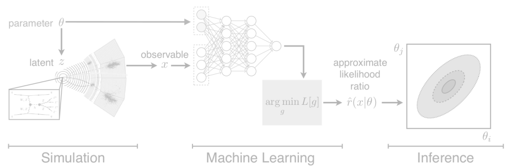


Credits: Johann Brehmer

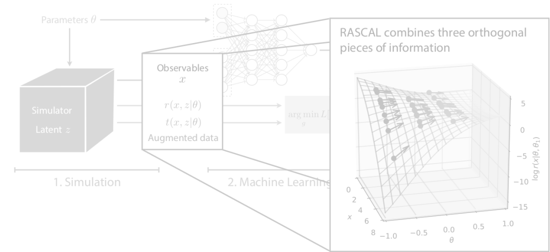
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

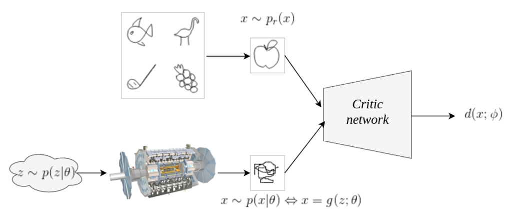


Histograms of observables
Neural density (ratio) estimation

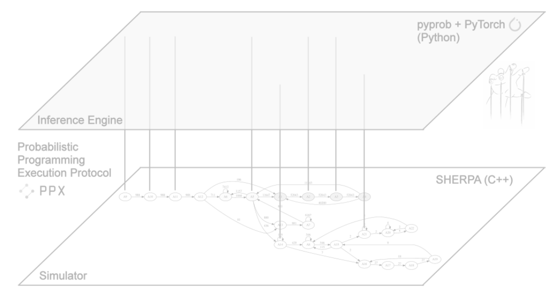


Mining gold from implicit models

Learn to control the simulator

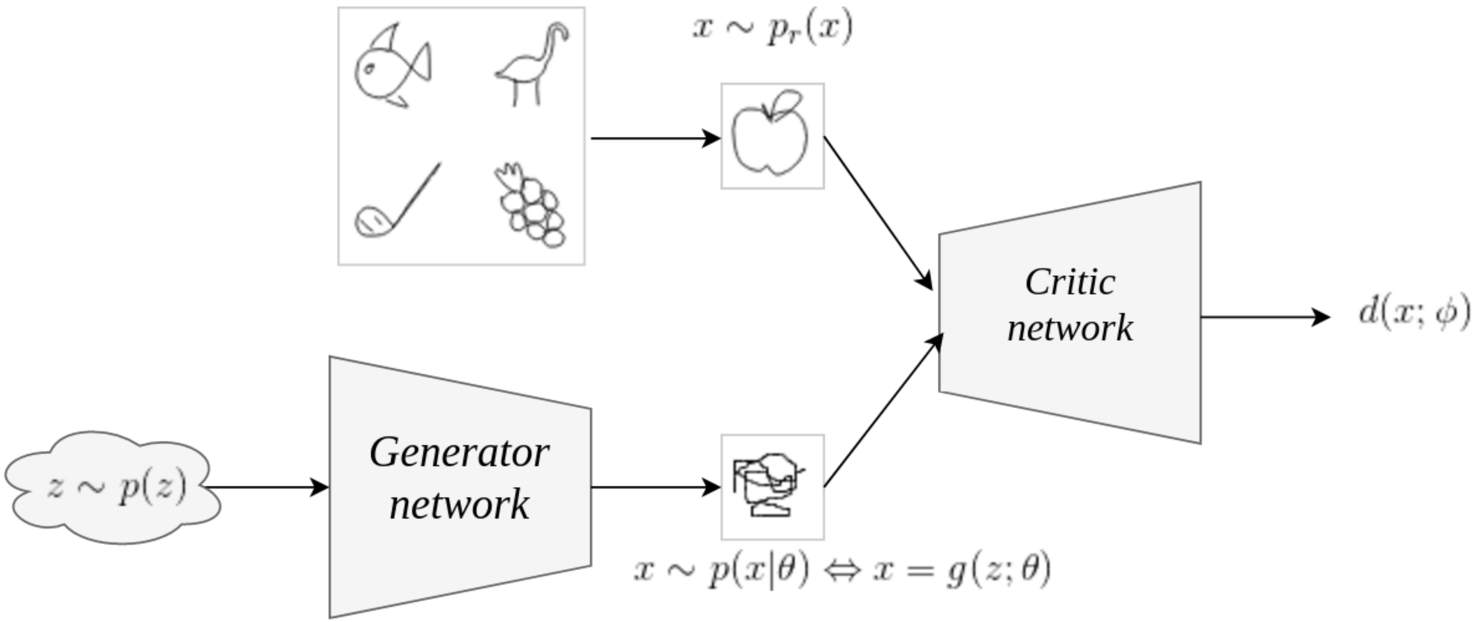


Adversarial variational optimization



Probabilistic programming

Generative adversarial networks



$$\mathcal{L}_d(\phi) = \mathbb{E}_{\mathbf{x}\sim p_r(\mathbf{x})}\left[-\log(d(\mathbf{x};\phi))\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[-\log(1 - d(g(\mathbf{z};\theta);\phi))\right]$$
$$\mathcal{L}_g(\theta) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log(1 - d(g(\mathbf{z};\theta);\phi))\right]$$

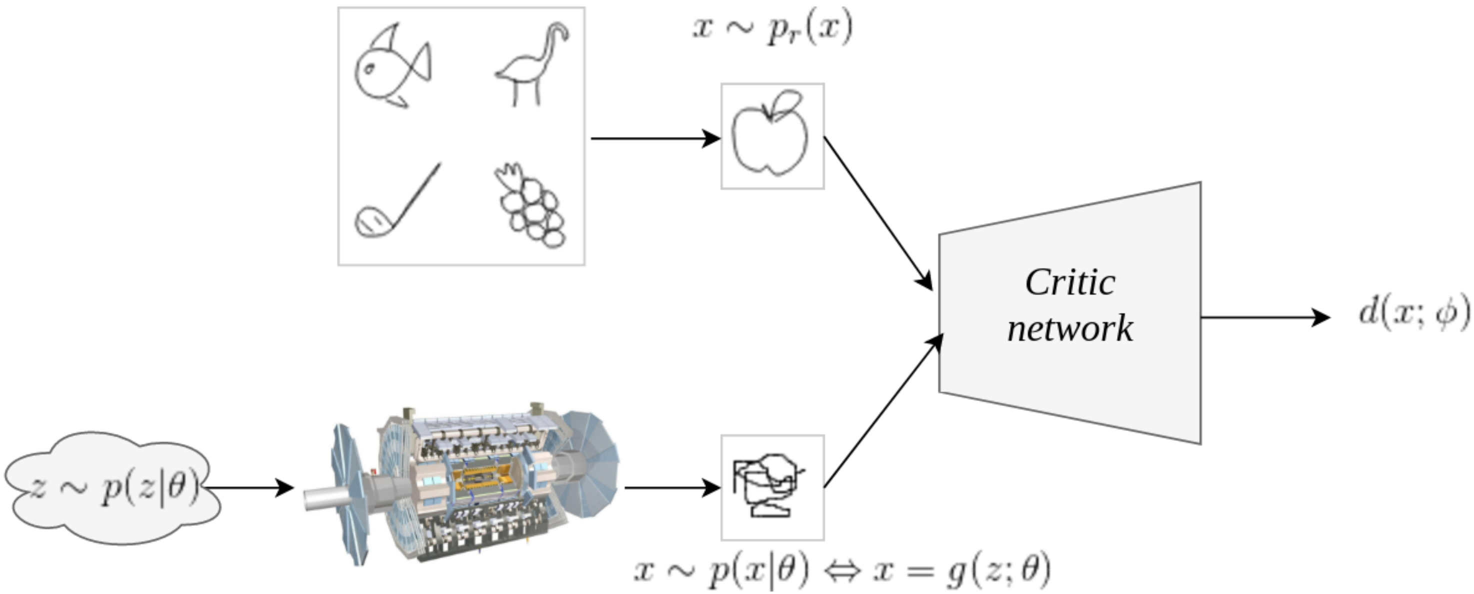
Odena et al 2016

Miyato et al 2017

Zhang et al 2018

Brock et al 2018

AVO



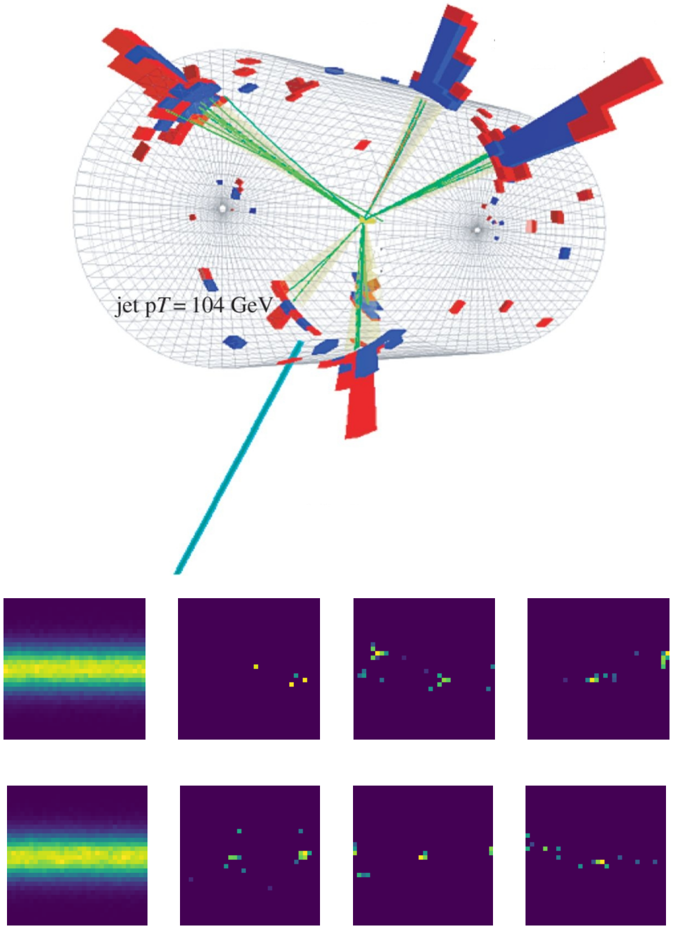
Replace g with an actual scientific simulator!

Key insights

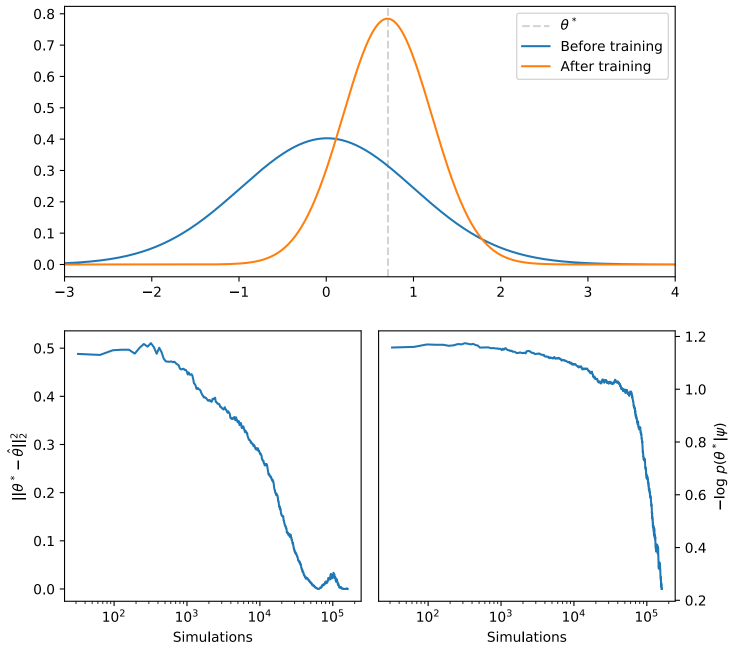
- Replace the generative network with a non-differentiable forward simulator $g(\mathbf{z}; \theta)$.
- Let the neural network critic figure out how to adjust the simulator parameters.
- Combine with variational optimization to bypass the non-differentiability by optimizing upper bounds of the adversarial objectives

$$
\begin{aligned}
U_d(\phi) &= \mathbb{E}_{\theta \sim q(\theta;\psi)} \left[\mathcal{L}_d(\phi)\right] \\
U_g(\psi) &= \mathbb{E}_{\theta \sim q(\theta;\psi)} \left[\mathcal{L}_g(\theta)\right]
\end{aligned}
$$

respectively over ϕ and ψ.



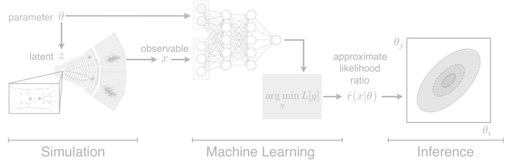
Samples for $\theta = 0$ (top) vs.
samples for $\theta = 0.81$ (bottom).



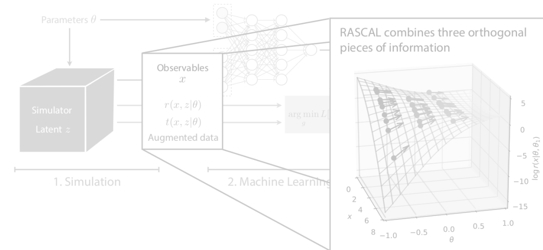
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

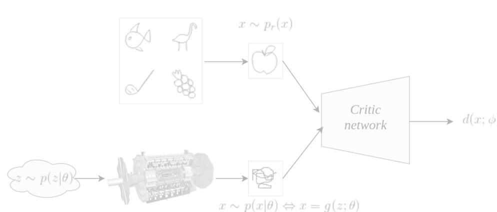


Histograms of observables
Neural density (ratio) estimation

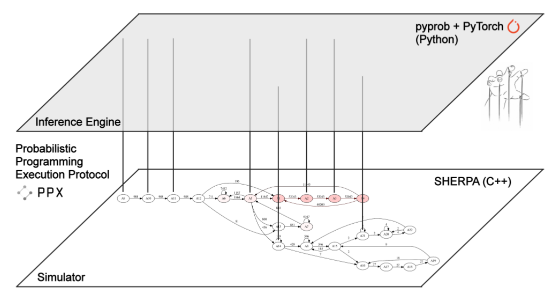


Mining gold from implicit models

Learn to control the simulator

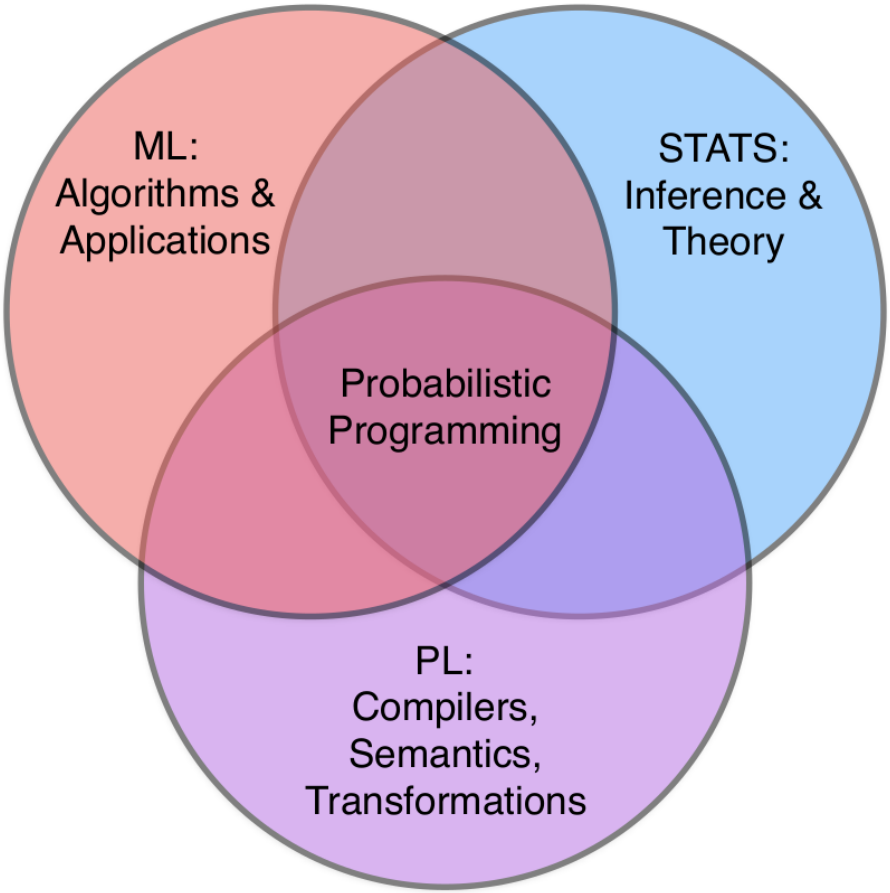


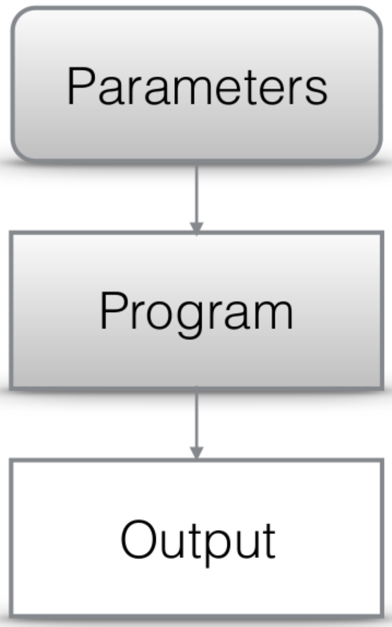
Adversarial variational optimization



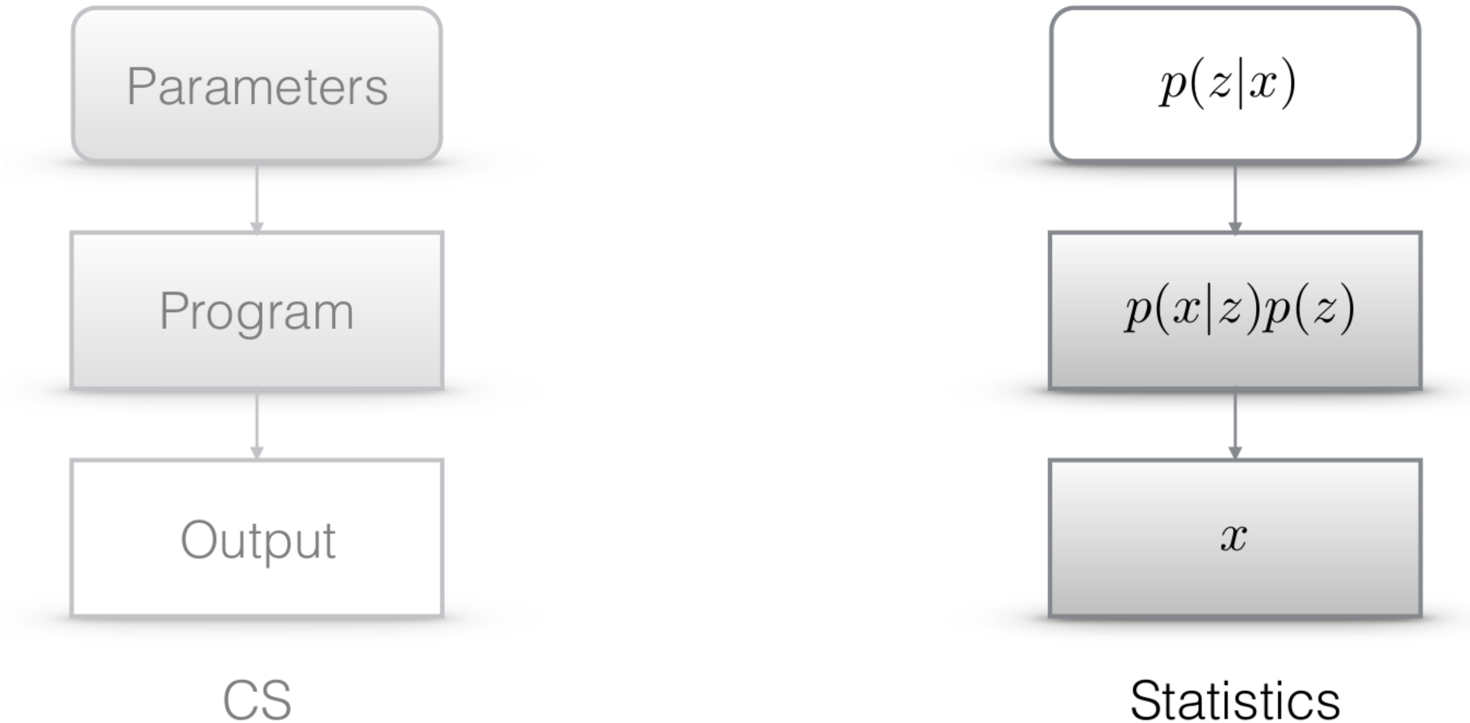
Probabilistic programming

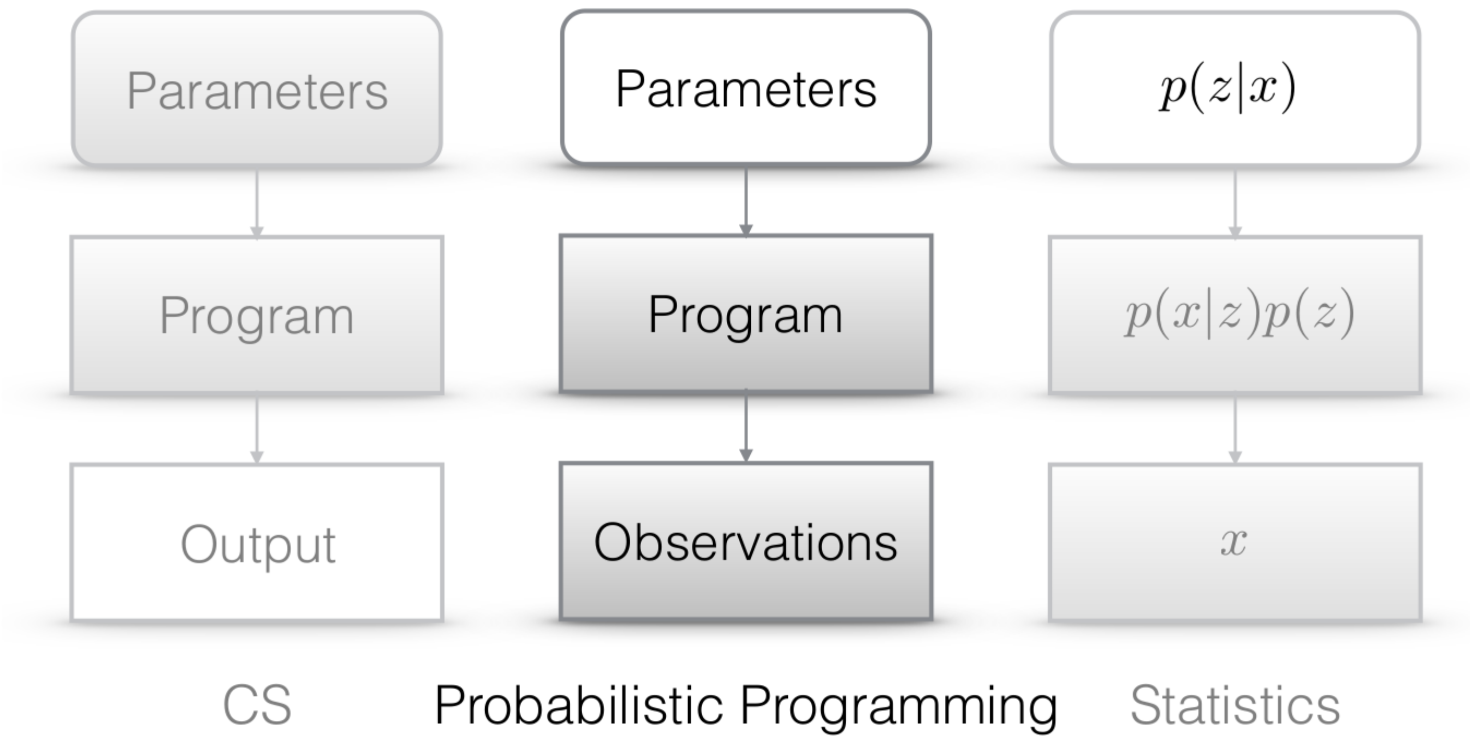
Probabilistic programming

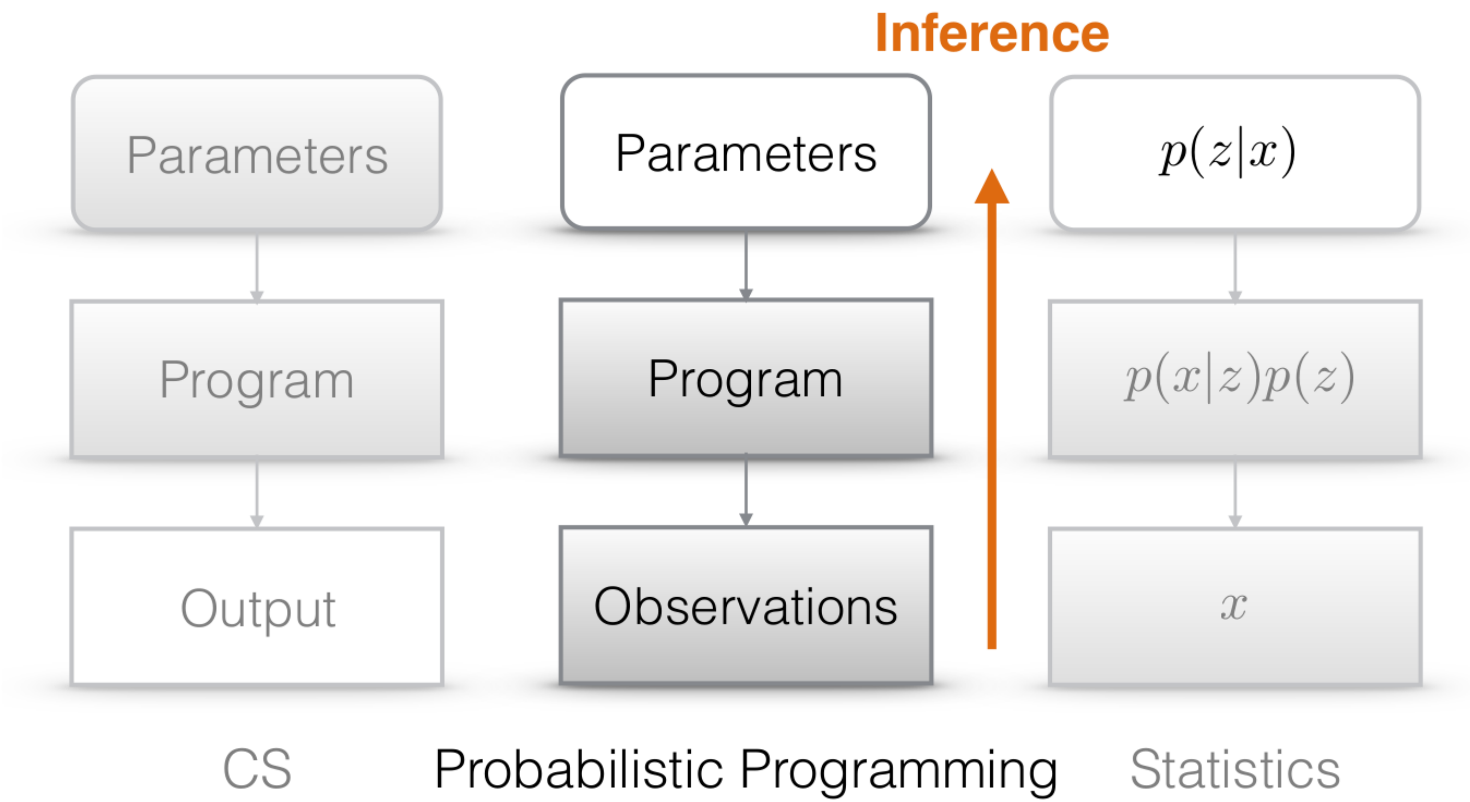


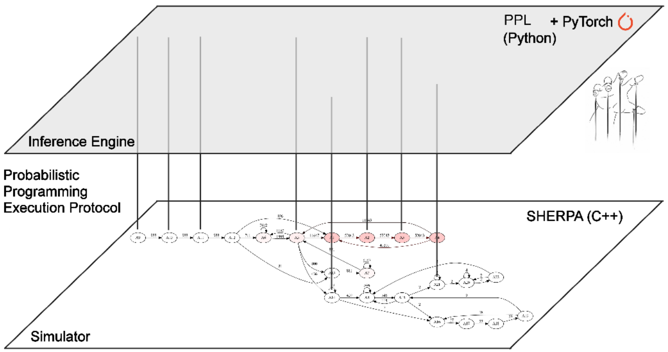


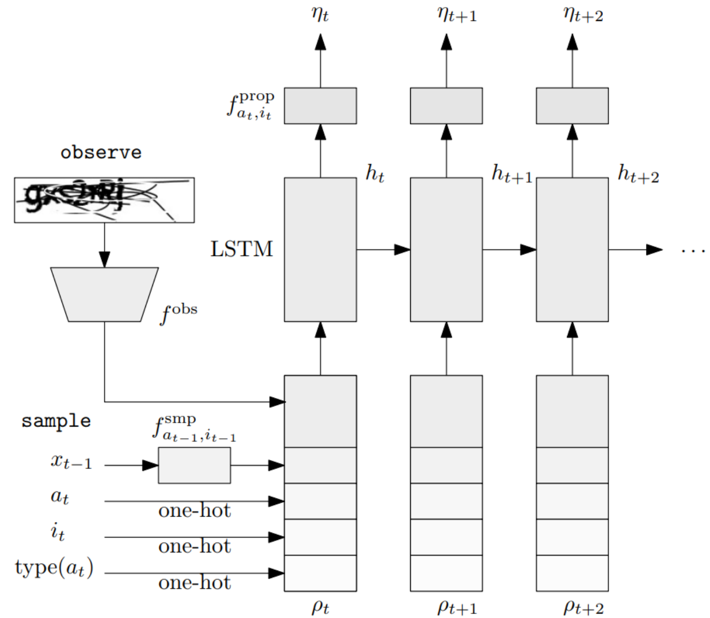
CS


Parameters
Program
Output
CS
$p(z|x)$
$p(x|z)p(z)$
x
Statistics


Parameters
Program
Output
CS
Parameters
Program
Observations
Probabilistic Programming
$p(z|x)$
$p(x|z)p(z)$
x
Statistics


Inference
Parameters
Program
Output
Parameters
Program
Observations
$p(z|x)$
$p(x|z)p(z)$
x
CS
Probabilistic Programming
Statistics





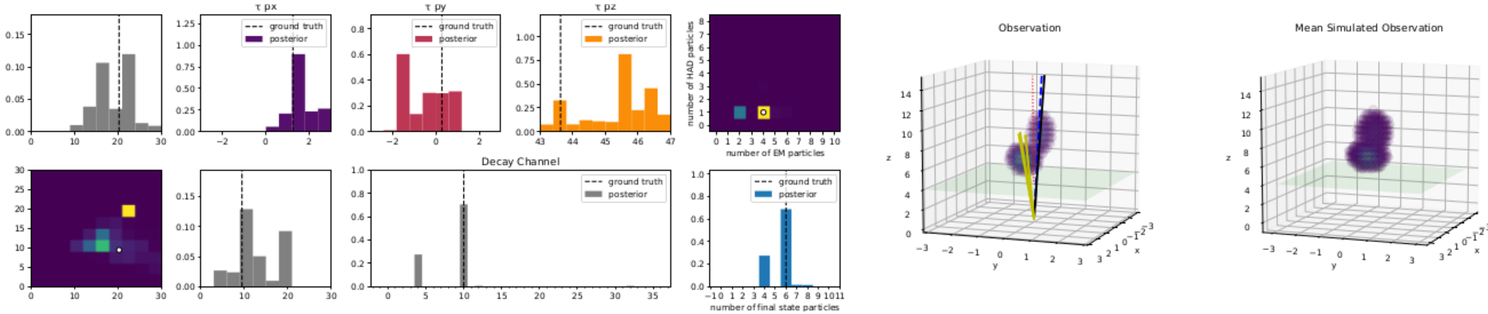
Key insights

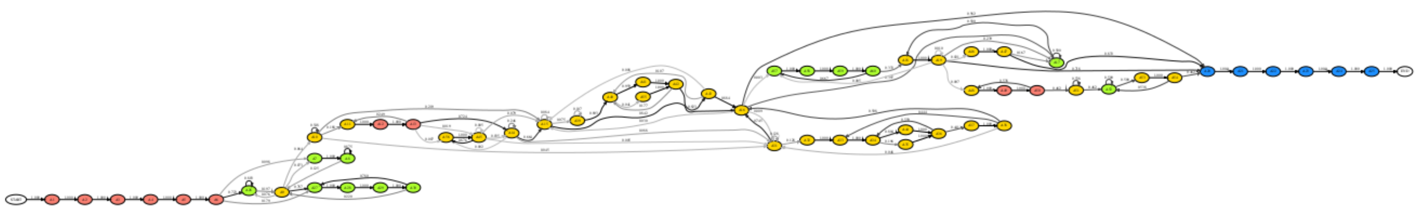
Let a neural network take full control of the internals of the simulation program by hijacking all calls to the random number generator.

Image credits: Le, Baydin and Wood, Inference Compilation and Universal Probabilistic Programming, arXiv:160.09900.

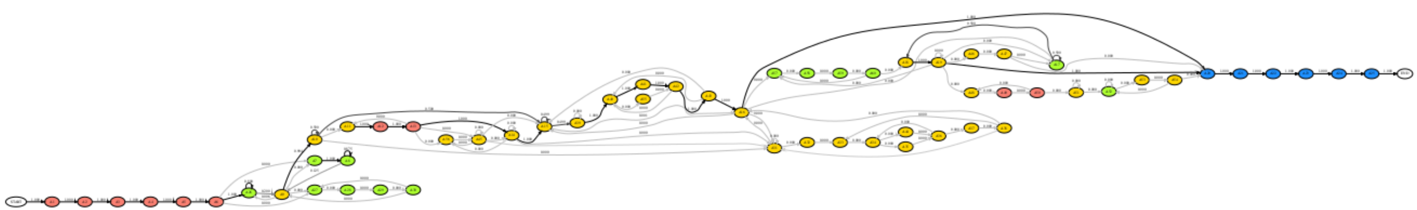
```
(defquery captcha
 [image num-chars tol]
 (let [[w h] (size image)
       ;; sample random characters
       num-chars (sample
                   (poisson num-chars))
       chars (repeatedly
               num-chars sample-char)]
  ;; compare rendering to true image
  (map (fn [y z]
         (observe (normal z tol) y))
       (reduce-dim image)
       (reduce-dim (render chars w h)))
  ;; predict captcha text
  {:text
   (map :symbol (sort-by :x chars))}))
```

How to break captchas with probabilistic programming





(a) Prior execution $p(\mathbf{x})$.



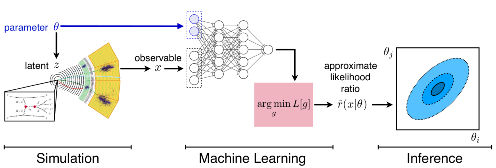
(b) Posterior execution $p(\mathbf{x}|\mathbf{y})$ conditioned on a given calorimeter observation $\mathbf{y}$.

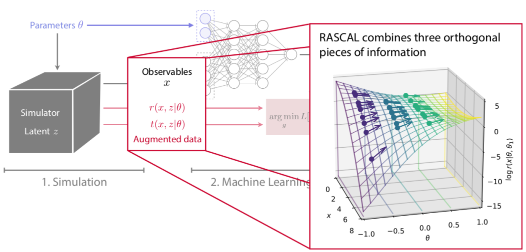
Probabilistic programming for particle physics
(work in progress)

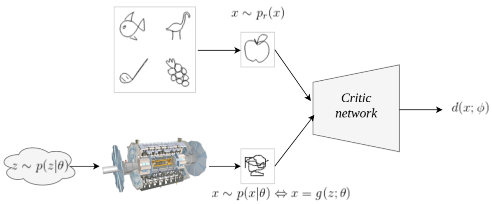
Summary

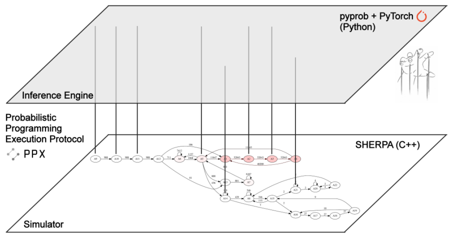
Summary

- Much of modern science is based on "likelihood-free" simulations.
- Recent (and older) developments from machine learning offer solutions for likelihood-free inference, including:
    - Supervised learning
    - Neural networks trained with augmented data
    - Adversarial training
    - Probabilistic programming









Collaborators








References

- Stoye, M., Brehmer, J., Louppe, G., Pavez, J., & Cranmer, K. (2018). Likelihood-free inference with an improved cross-entropy estimator. arXiv preprint arXiv:1808.00973.
- Baydin, A. G., Heinrich, L., Bhimji, W., Gram-Hansen, B., Louppe, G., Shao, L., ... & Wood, F. (2018). Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model. arXiv preprint arXiv:1807.07706.
- Brehmer, J., Louppe, G., Pavez, J., & Cranmer, K. (2018). Mining gold from implicit models to improve likelihood-free inference. arXiv preprint arXiv:1805.12244.
- Brehmer, J., Cranmer, K., Louppe, G., & Pavez, J. (2018). Constraining Effective Field Theories with Machine Learning. arXiv preprint arXiv:1805.00013.
- Brehmer, J., Cranmer, K., Louppe, G., & Pavez, J. (2018). A Guide to Constraining Effective Field Theories with Machine Learning. arXiv preprint arXiv:1805.00020.
- Casado, M. L., Baydin, A. G., Rubio, D. M., Le, T. A., Wood, F., Heinrich, L., ... & Bhimji, W. (2017). Improvements to Inference Compilation for Probabilistic Programming in Large-Scale Scientific Simulators. arXiv preprint arXiv:1712.07901.
- Louppe, G., & Cranmer, K. (2017). Adversarial Variational Optimization of Non-Differentiable Simulators. arXiv preprint arXiv:1707.07113.
- Cranmer, K., Pavez, J., & Louppe, G. (2015). Approximating likelihood ratios with calibrated discriminative classifiers. arXiv preprint arXiv:1506.02169.

The end.