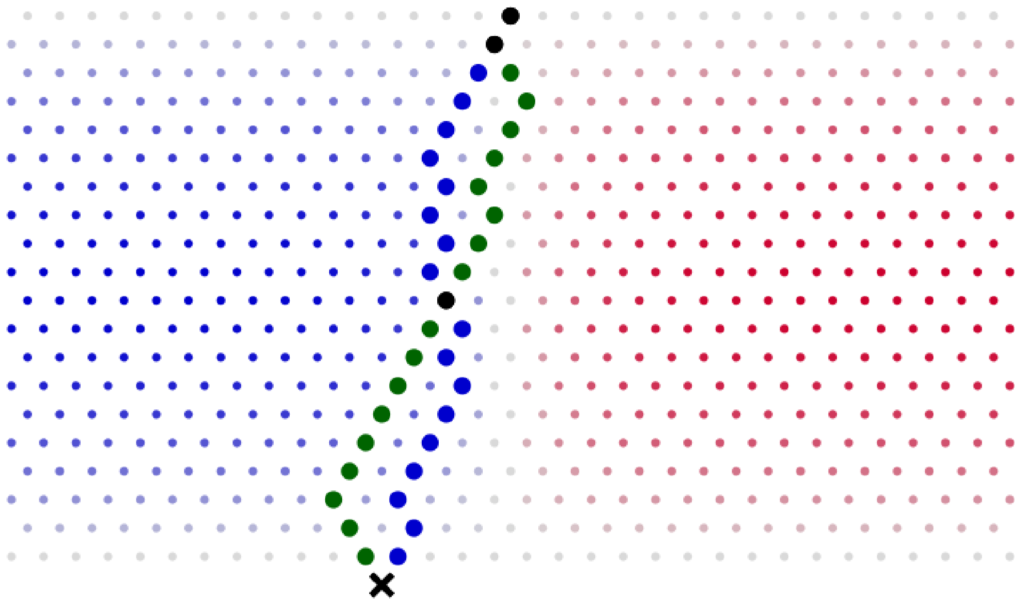
Likelihood-free inference in Physical Sciences

Machine Learning in High Energy Physics Summer School 2019
July 4, DESY

Gilles Louppe
g.louppe@uliege.be



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

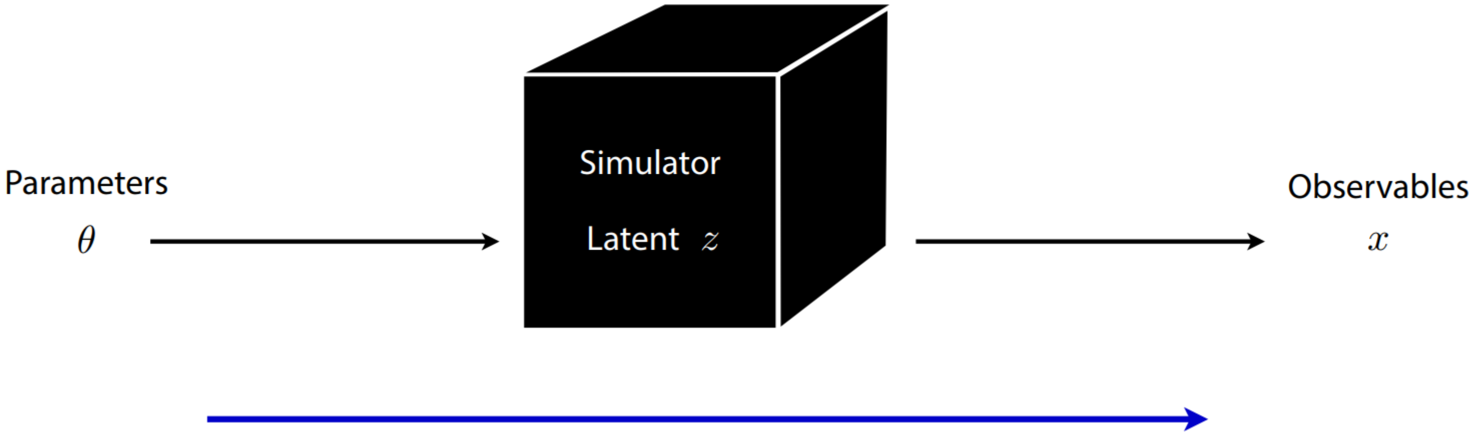
$$p(x|\theta) = \int p(x, z|\theta) dz = \binom{n}{x} \theta^x (1-\theta)^{n-x}$$

What if we shift or remove some of the pins?

The 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)

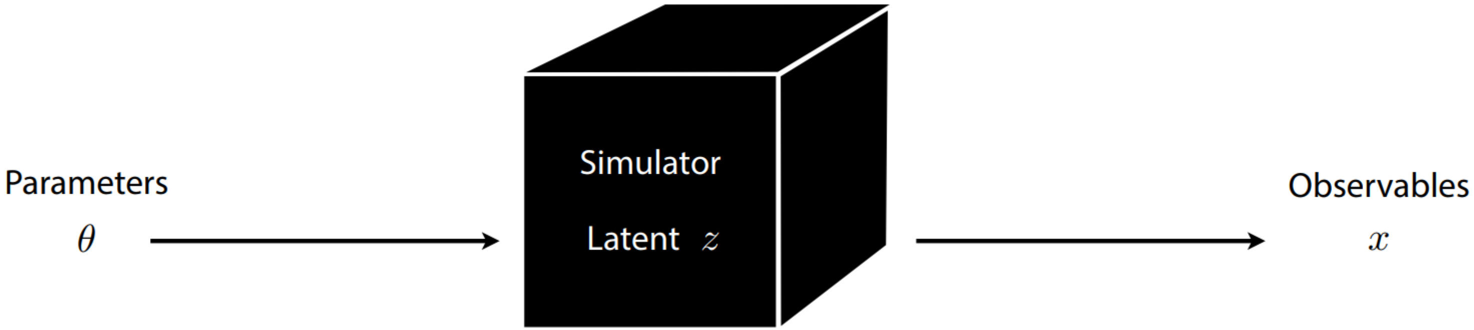
Inference in this context requires likelihood-free algorithms.



Prediction:

- Well-understood mechanistic model
- Simulator can generate samples

Credits: Johann Brehmer.

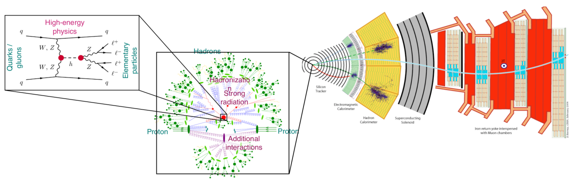


Prediction:
- Well-understood mechanistic model
- Simulator can generate samples

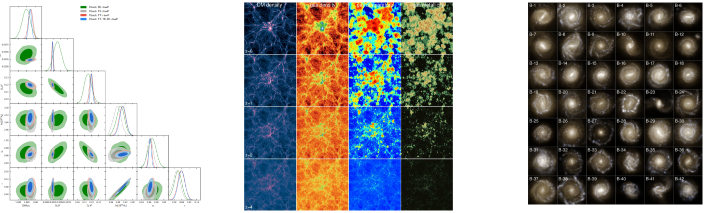
Inference:
- Likelihood function $p(x|\theta)$ is intractable
- Inference based on estimator $\hat{p}(x|\theta)$

Credits: Johann Brehmer.

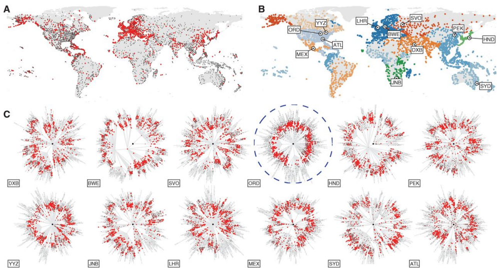
Applications



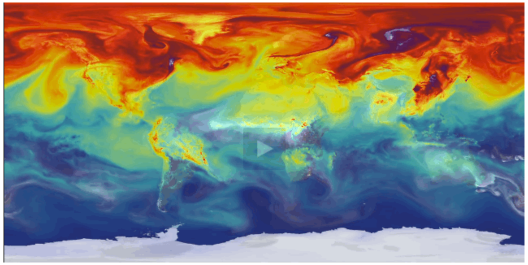
Particle physics



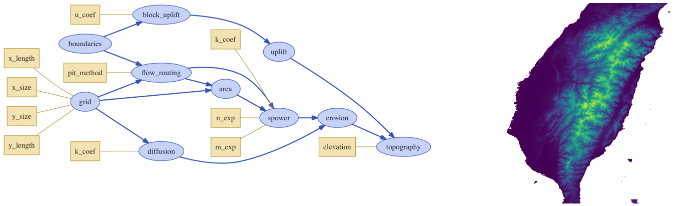
Cosmology



Epidemiology



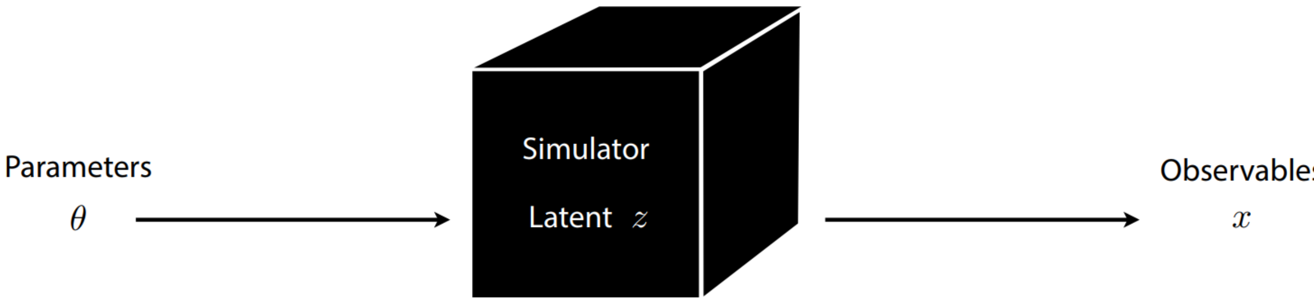
Climatology

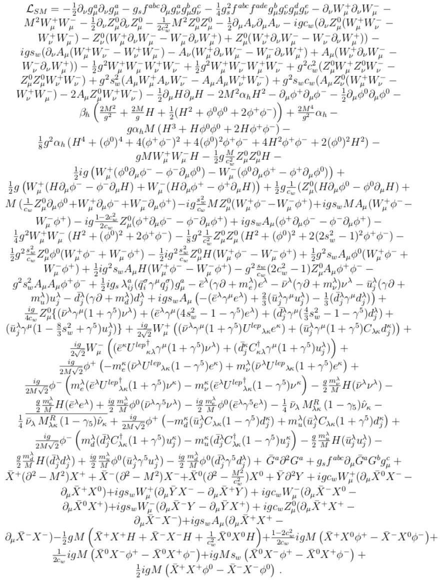


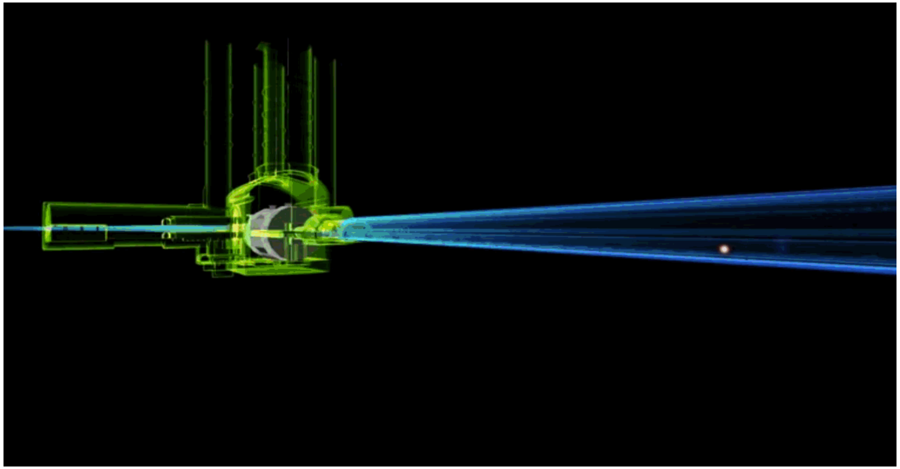
Computational topography

Astronomy

Particle physics







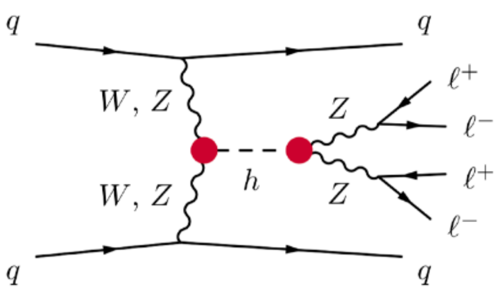
Latent variables

Parameters of interest

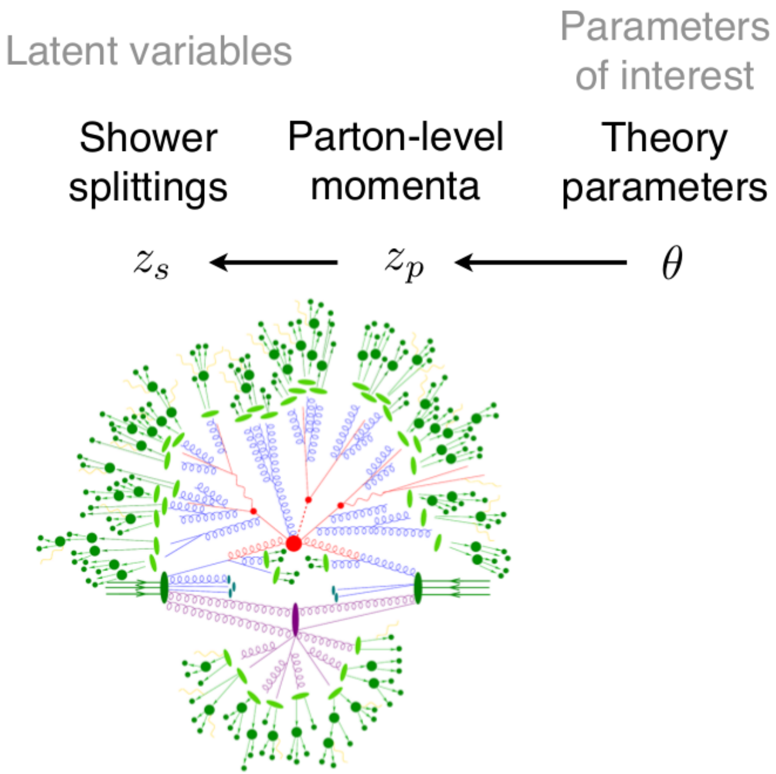
Parton-level momenta

Theory parameters

$$z_p \longleftarrow \theta$$



Credits: Johann Brehmer.



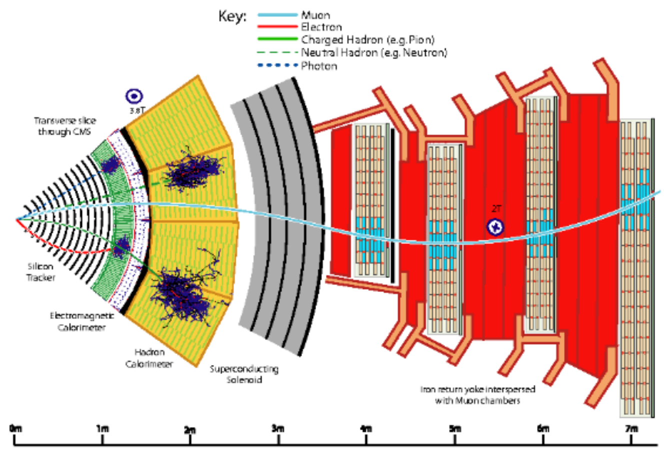
Credits: Johann Brehmer.

Latent variables

Parameters of interest

Detector interactions	Shower splittings	Parton-level momenta	Theory parameters

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

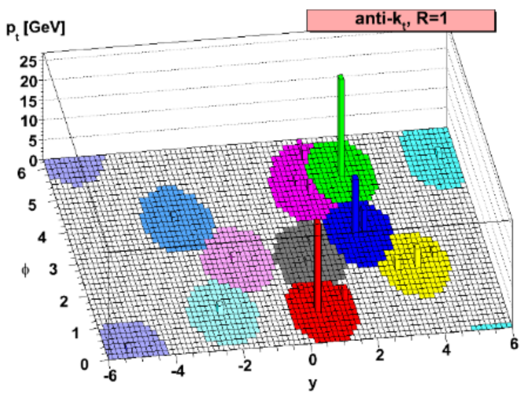


Credits: Johann Brehmer.

Features | Latent variables | Parameters of interest

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

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



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

Credits: Johann Brehmer.

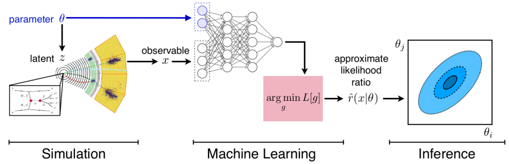
$$p(x|\theta) = \underbrace{\iiint}_{\text{intractable}} p(z_p|\theta)p(z_s|z_p)p(z_d|z_s)p(x|z_d)dz_p dz_s dz_d$$

Likelihood-free inference algorithms

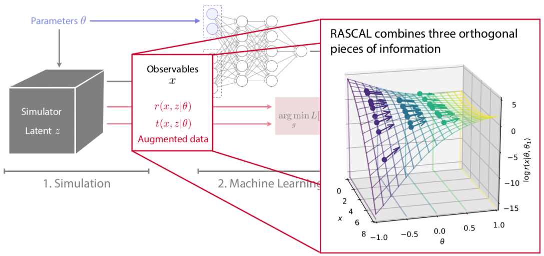
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

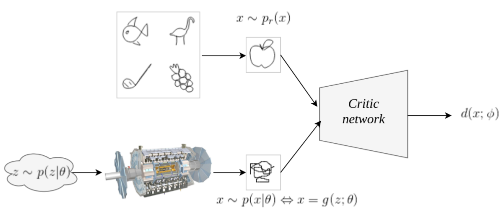


Histograms of observables
Neural density (ratio) estimation

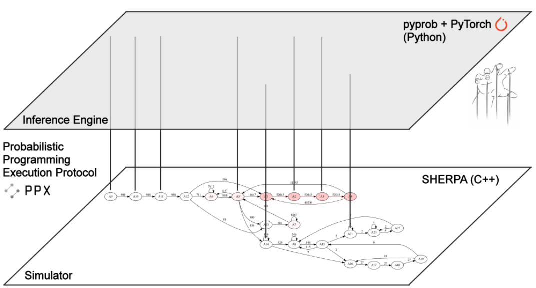


Mining gold from implicit models

Learn to control the simulator



Adversarial variational optimization

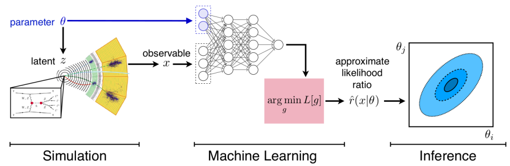


Probabilistic programming

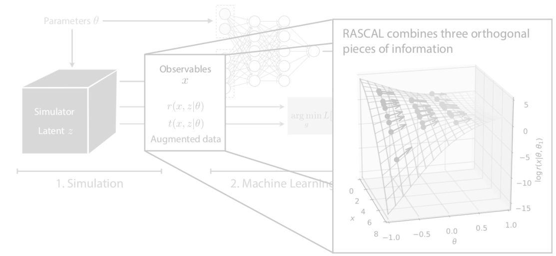
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

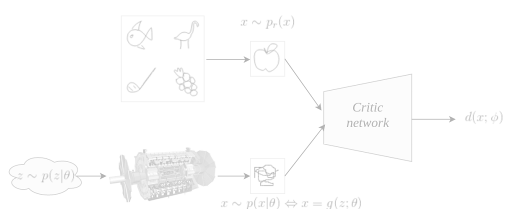


Histograms of observables
Neural density (ratio) estimation

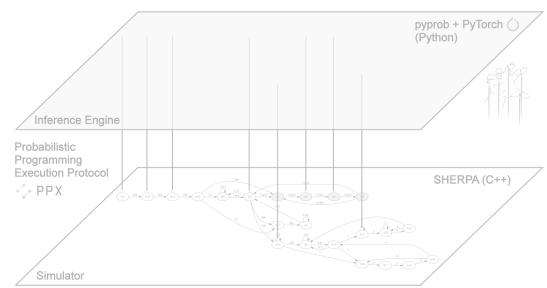


Mining gold from implicit models

Learn to control the simulator



Adversarial variational optimization



Probabilistic programming

Likelihood ratio

The likelihood ratio

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

is the quantity that is central to many statistical inference procedures.

Examples

- Frequentist hypothesis testing
- Supervised learning
- Bayesian posterior sampling with MCMC
- Bayesian posterior inference through Variational Inference
- Generative adversarial networks
- Empirical Bayes with Adversarial Variational Optimization
- Optimal compression

When solving a problem of interest, do not solve a more general problem as an intermediate step. – Vladimir Vapnik

$$\frac{p_{\mathbf{X}}(\mathbf{x}|\theta_0)}{p_{\mathbf{X}}(\mathbf{x}|\theta_1)} = r(\mathbf{x};\theta_0,\theta_1)$$

Direct likelihood ratio estimation is simpler than density estimation.

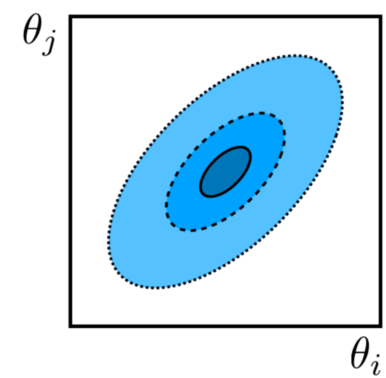
(This is fortunate, we are in the likelihood-free scenario!)

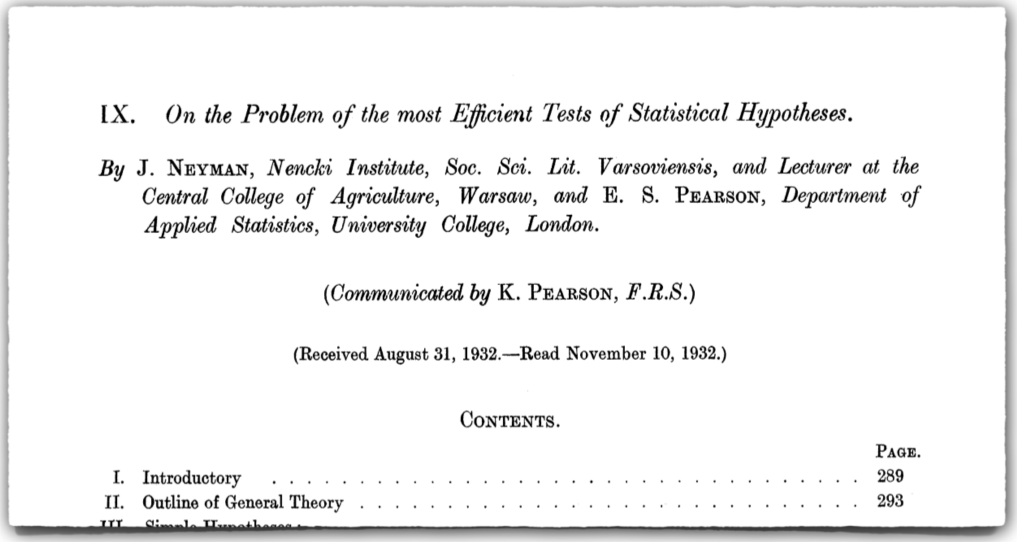
The frequentist physicist's way

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.





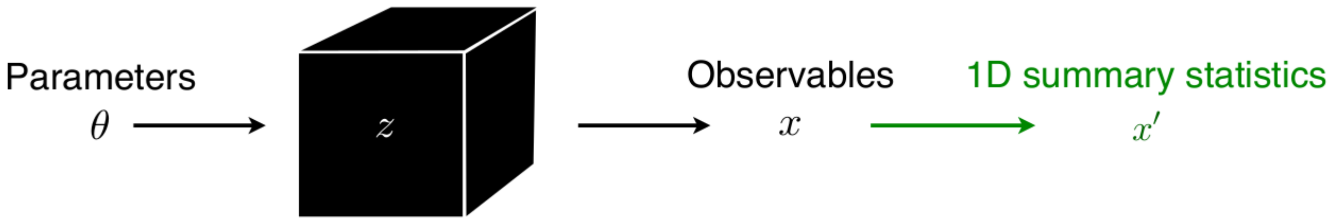
IX. *On the Problem of the most Efficient Tests of Statistical Hypotheses.*

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.



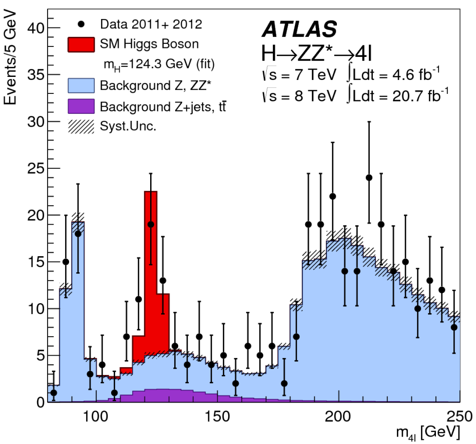
Define a projection function $s : \mathcal{X} \to \mathbb{R}$ mapping observables x to a summary statistics $x' = s(x)$.

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

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

From this it comes

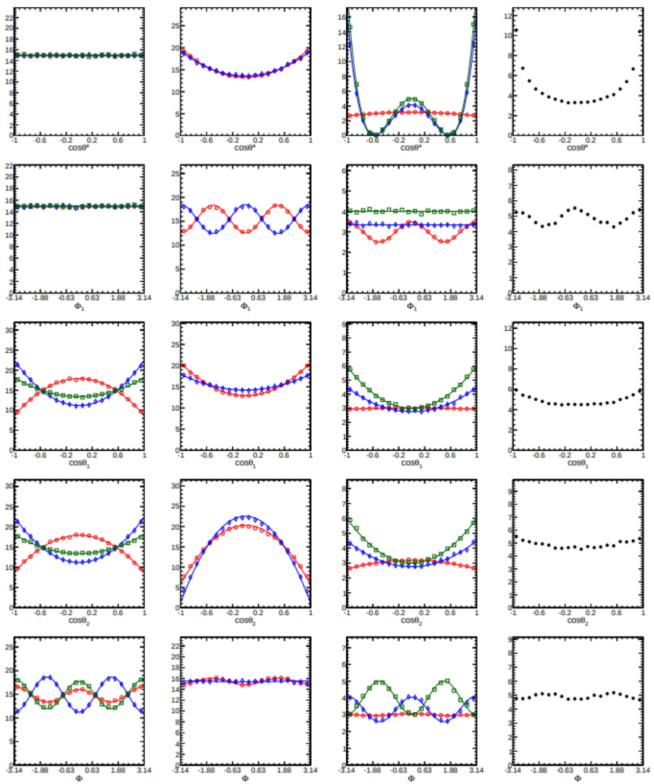
$$\frac{p(x|\theta_0)}{p(x|\theta_1)} \approx \frac{\hat{p}(x|\theta_0)}{\hat{p}(x|\theta_1)} = \hat{r}(x|\theta_0, \theta_1).$$



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.

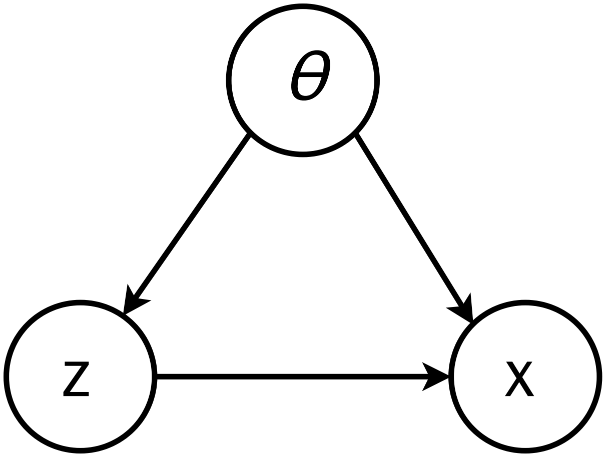


Bolognesi et al, 2012 (arXiv:1208.4018).

Bayesian inference

Bayesian inference usually consists in computing the posterior

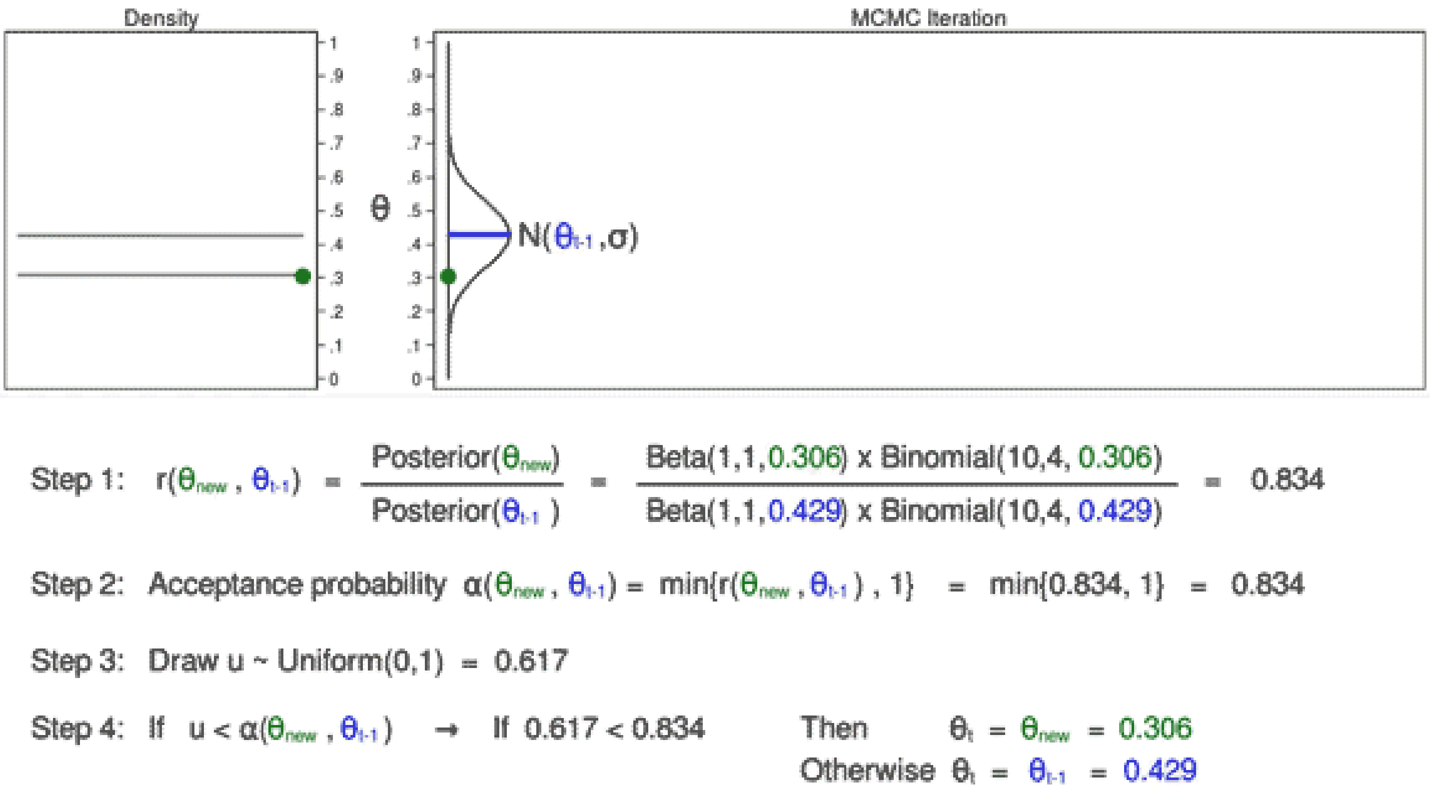
$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}.$$



Doubly intractable in the likelihood-free scenario:

- Cannot evaluate the evidence $p(x) = \int p(x|\theta)p(\theta)d\theta$.
- Cannot evaluate the likelihood $p(x|\theta) = \int p(x, z|\theta)dz$.

Posterior sampling



MCMC algorithms can be made likelihood-free by plugging in the likelihood ratio.

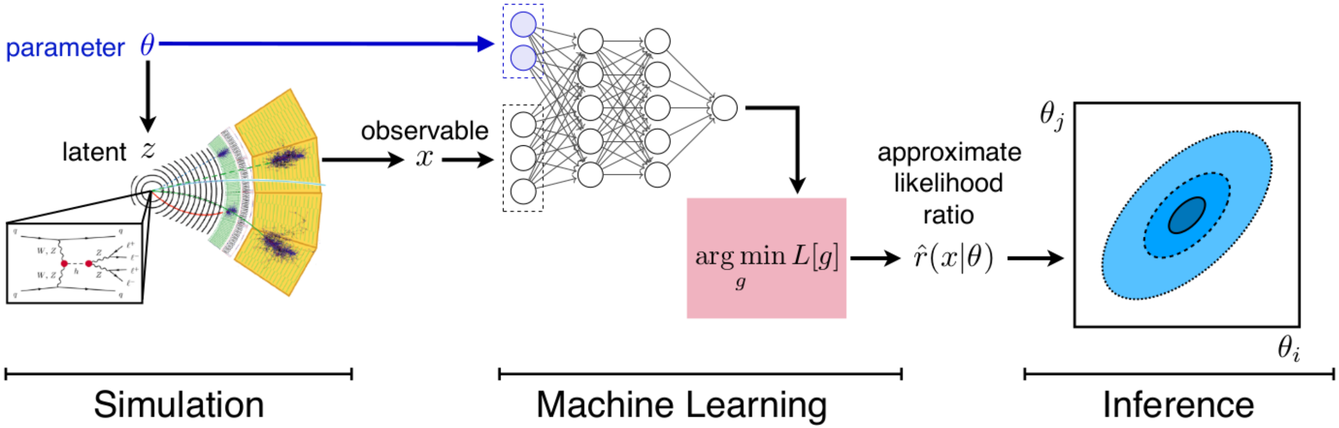
Chuck Huber, 2016; Hermans and Louppe, 2019 [arXiv:1903.04057].

CARL

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

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



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,

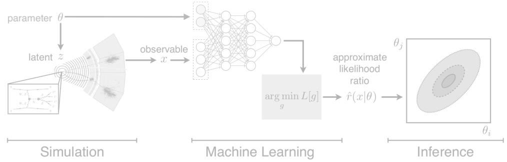
$$r(x|\theta_0, \theta_1) \approx \hat{r}(x|\theta_0, \theta_1) = \frac{1 - \hat{s}(x)}{\hat{s}(x)}$$

That is, supervised classification is equivalent to likelihood ratio estimation.

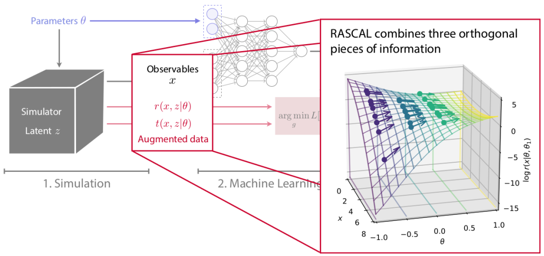
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

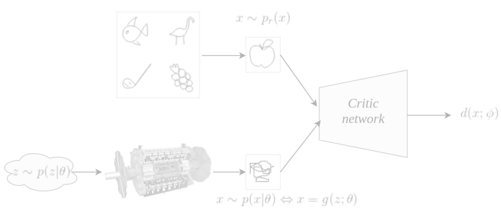


Histograms of observables
Neural density (ratio) estimation

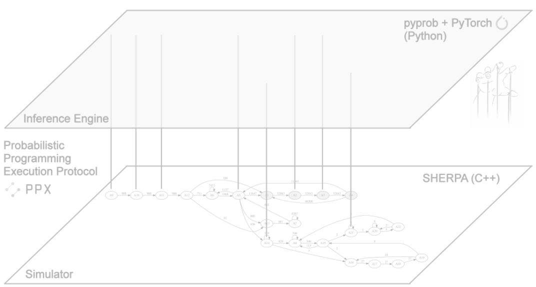


Mining gold from implicit models

Learn to control the simulator

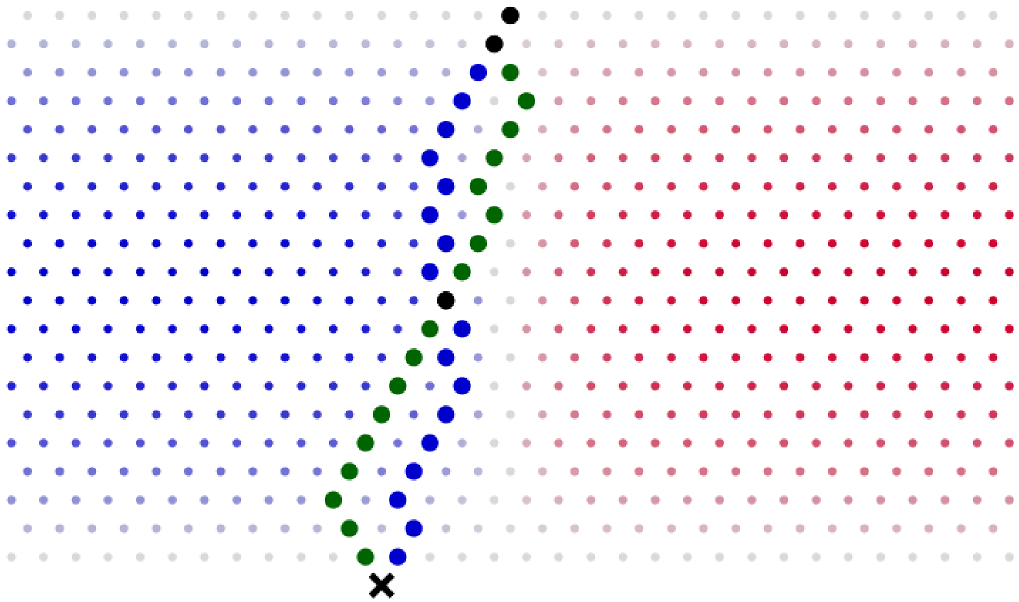


Adversarial variational optimization



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

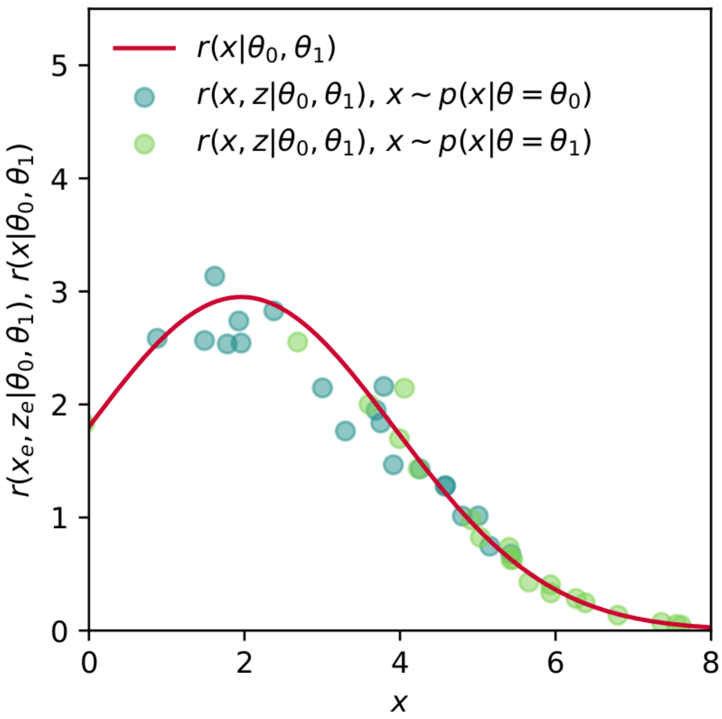
We call this process mining gold from your simulator!

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:

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

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

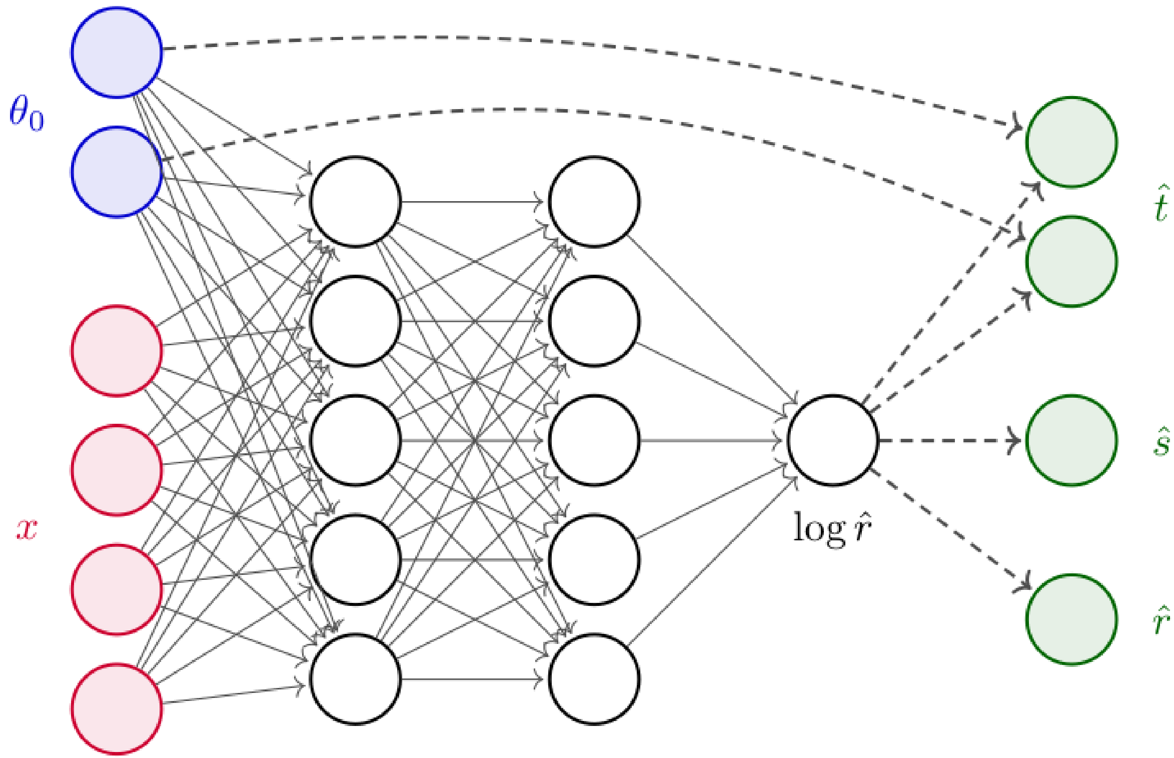
$$\begin{aligned} g^*(x) &= \frac{1}{p(x|\theta)} \int p(x,z|\theta) g(x,z) dz \\ &= \mathbb{E}_{p(z|x,\theta)}\left[g(x,z)\right] \end{aligned}$$

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

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

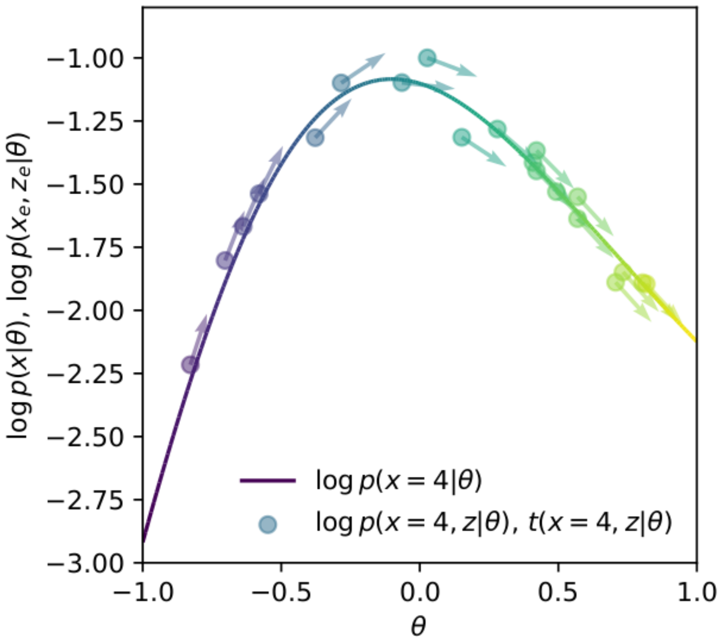


$$r^*(x|\theta_0, \theta_1) = \arg\min_{\hat{r}} L_r[\hat{r}]$$

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

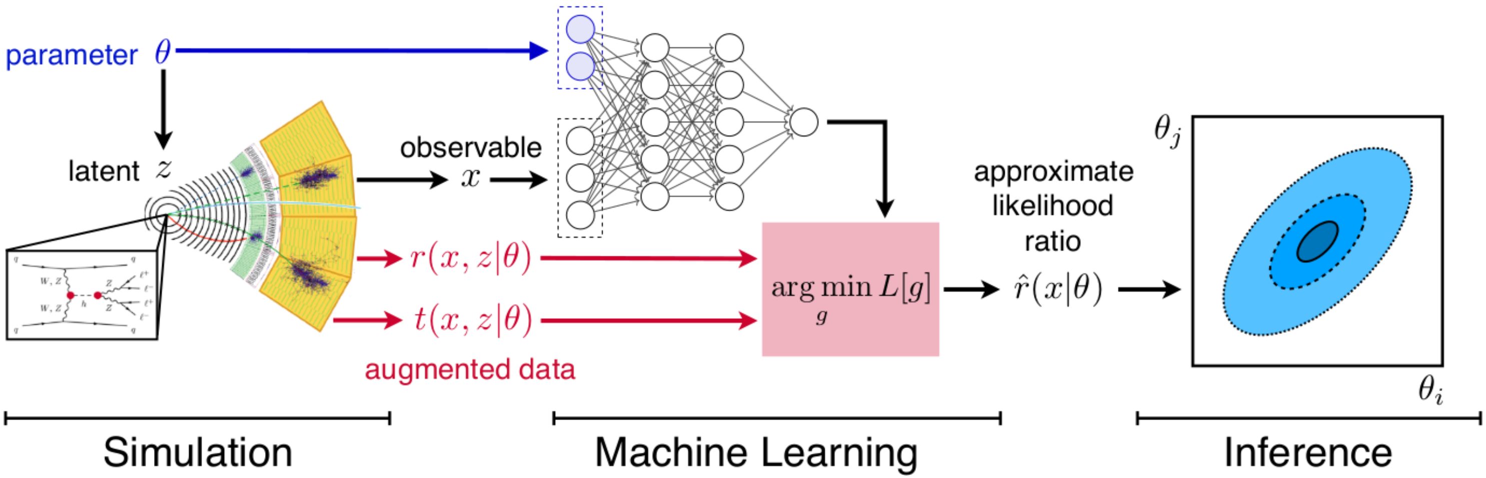
$$L_t = \mathbb{E}_{p(x,z|\theta_0)}\left[(t(x, z|\theta_0) - \hat{t}(x))^2\right],$$

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

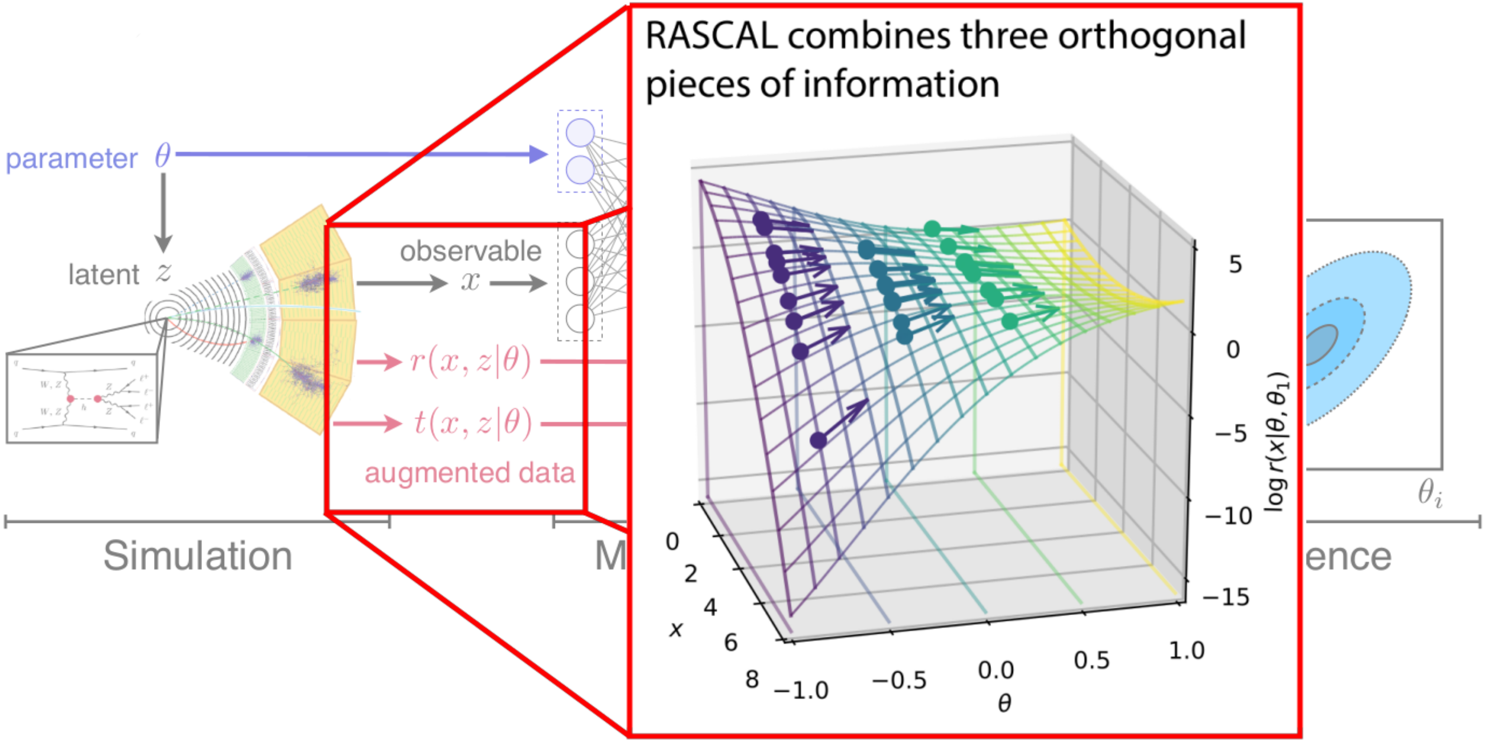
$$L_{RASCAL} = L_r + L_t$$



Brehmer et al, 2018 (arXiv:1805.12244)

RASCAL

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



Brehmer et al, 2018 (arXiv:1805.12244)

SALLY (= optimal compression)

The likelihood ratio r relates to the score

$$t(x|\theta_{\text{ref}}) = \nabla_\theta \log p(x|\theta)|_{\theta_{\text{ref}}} = \nabla_\theta r(x|\theta, \theta_{\text{ref}})|_{\theta_{\text{ref}}}.$$

- It quantifies the relative change of the likelihood under infinitesimal changes.
- It can be seen as a local equivalent of the likelihood ratio.

In a small patch around θ_{ref}, we have the approximation

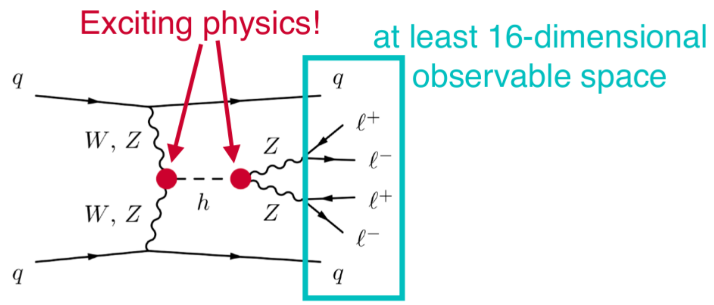
$$p_{\text{local}}(x|\theta) = \frac{1}{Z(\theta)} p(t(x|\theta_{\text{ref}})|\theta_{\text{ref}}) \exp(t(x|\theta_{\text{ref}}) \cdot (\theta - \theta_{\text{ref}}))$$

where the score $t(x|\theta_{\text{ref}})$ are its sufficient statistics. Therefore,

- in the local model the likelihood ratio between θ and θ_{ref} only depends on the product between the score and $\theta - \theta_{\text{ref}}$.
- That is, x can be compressed into a single scalar without loss of power.

Brehmer et al, 2018 [arXiv:1805.12244].

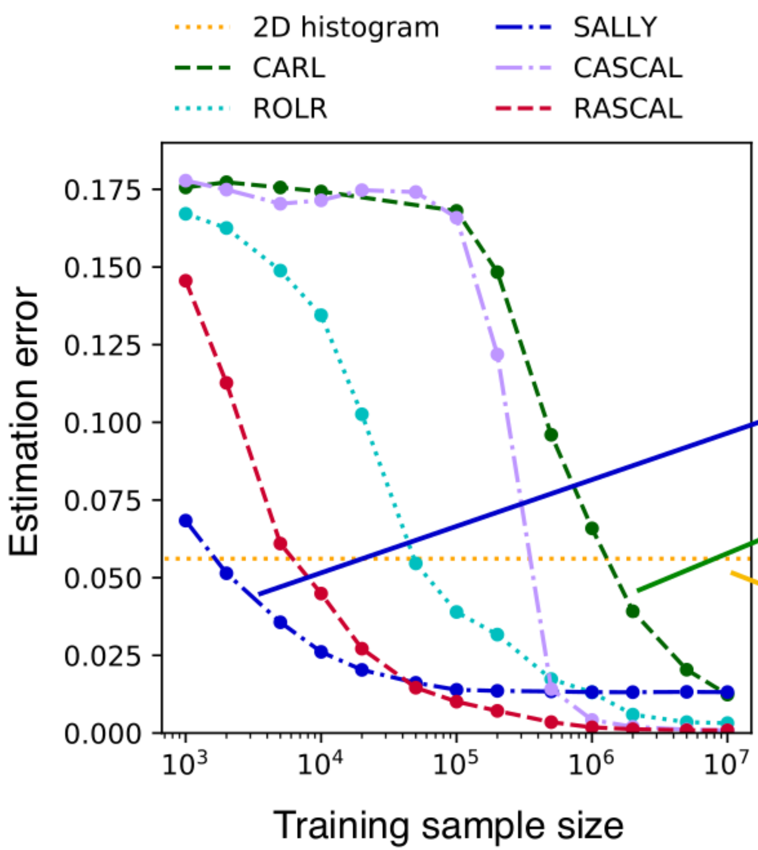
Results?



Experimental setup

- Higgs production in weak boson fusion.
- Goal: constraints on two theory parameters.

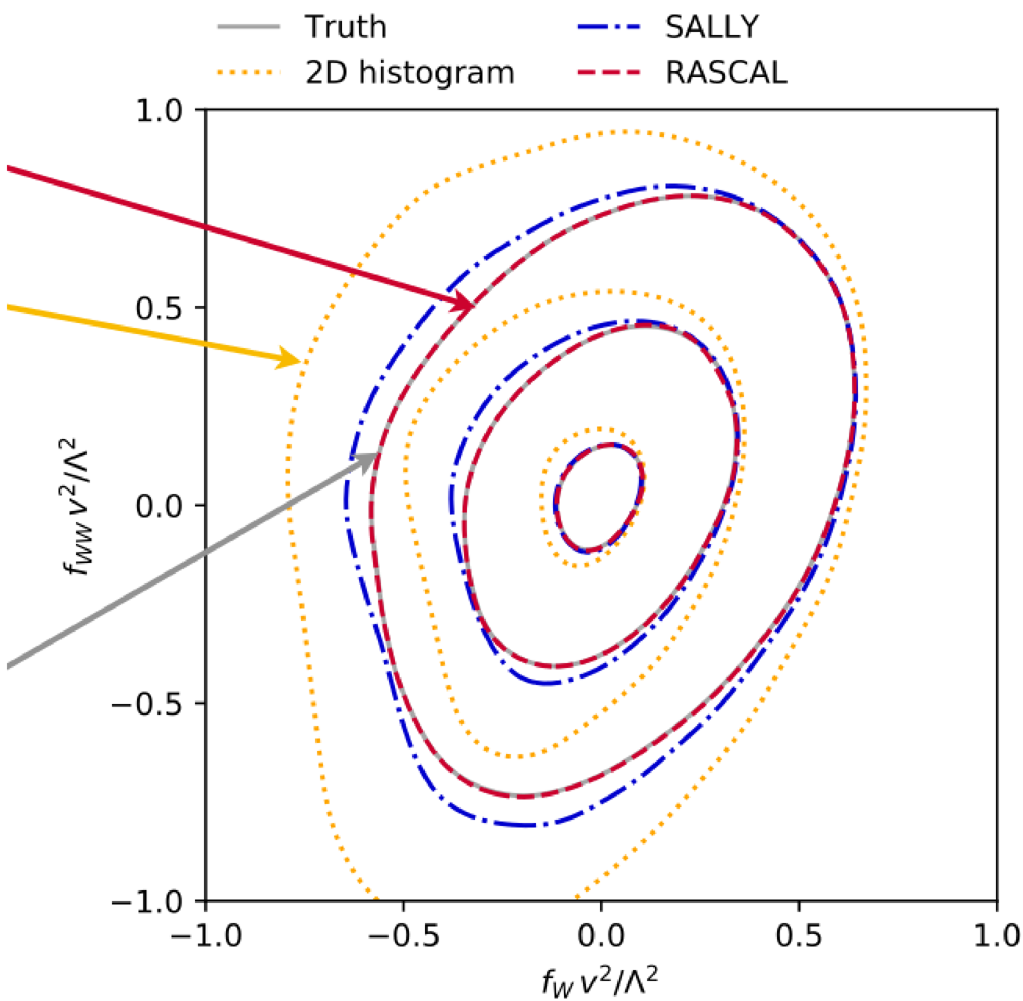
$$\mathcal{L} = \mathcal{L}_{SM} + \underbrace{\frac{f_W}{\Lambda^2}}_{} \frac{ig}{2} (D^\mu \phi)^\dagger \sigma^a D^\nu \phi \, W^a_{\mu\nu} - \underbrace{\frac{f_{WW}}{\Lambda^2}}_{} \frac{g^2}{4} (\phi^\dagger \phi) \, W^a_{\mu\nu} W^{\mu\nu\, a}$$



New techniques require less data than generic ML method
[CARL, see K. Cranmer, G. Louppe, J. Pavez 1506.02169]

Traditional Histogram

Brehmer et al, 2018 (arXiv:1805.12244)

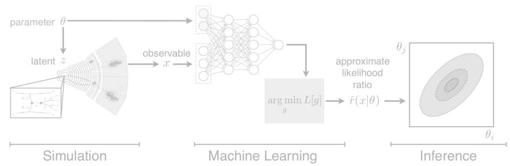


Brehmer et al, 2018 (arXiv:1805.12244)

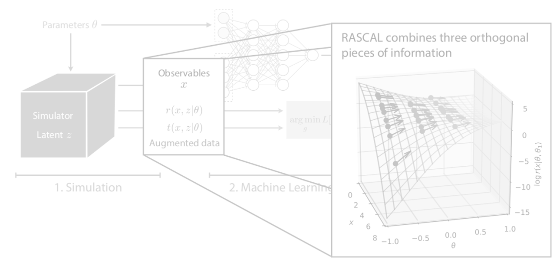
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

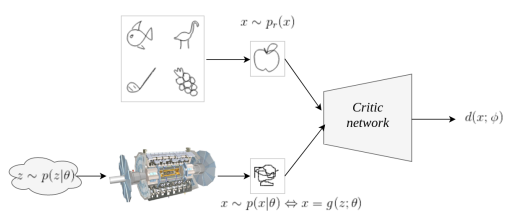


Histograms of observables
Neural density (ratio) estimation

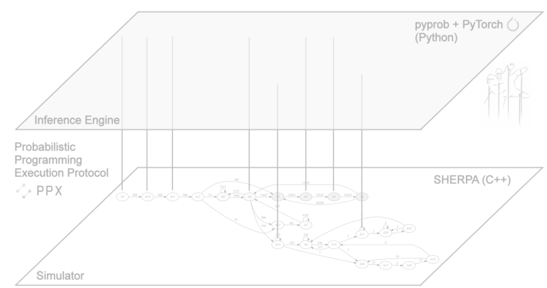


Mining gold from implicit models

Learn to control the simulator

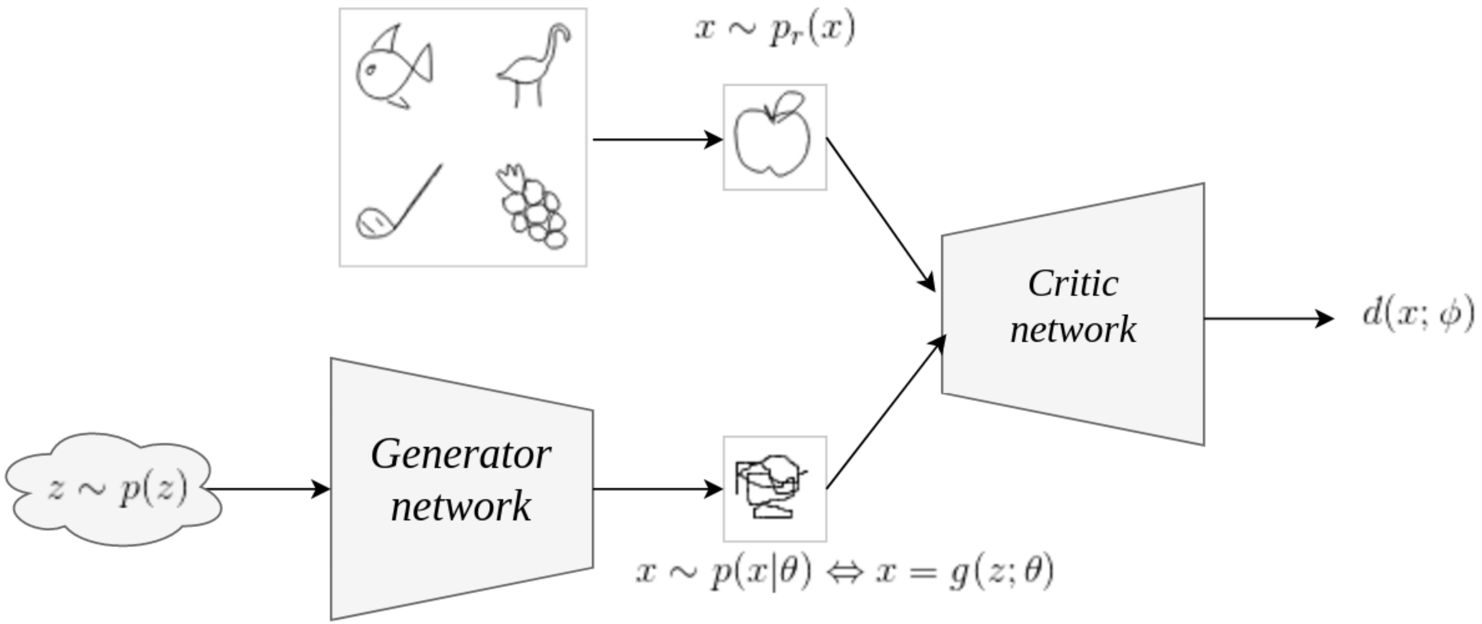


Adversarial variational optimization



Probabilistic programming

Generative adversarial networks



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

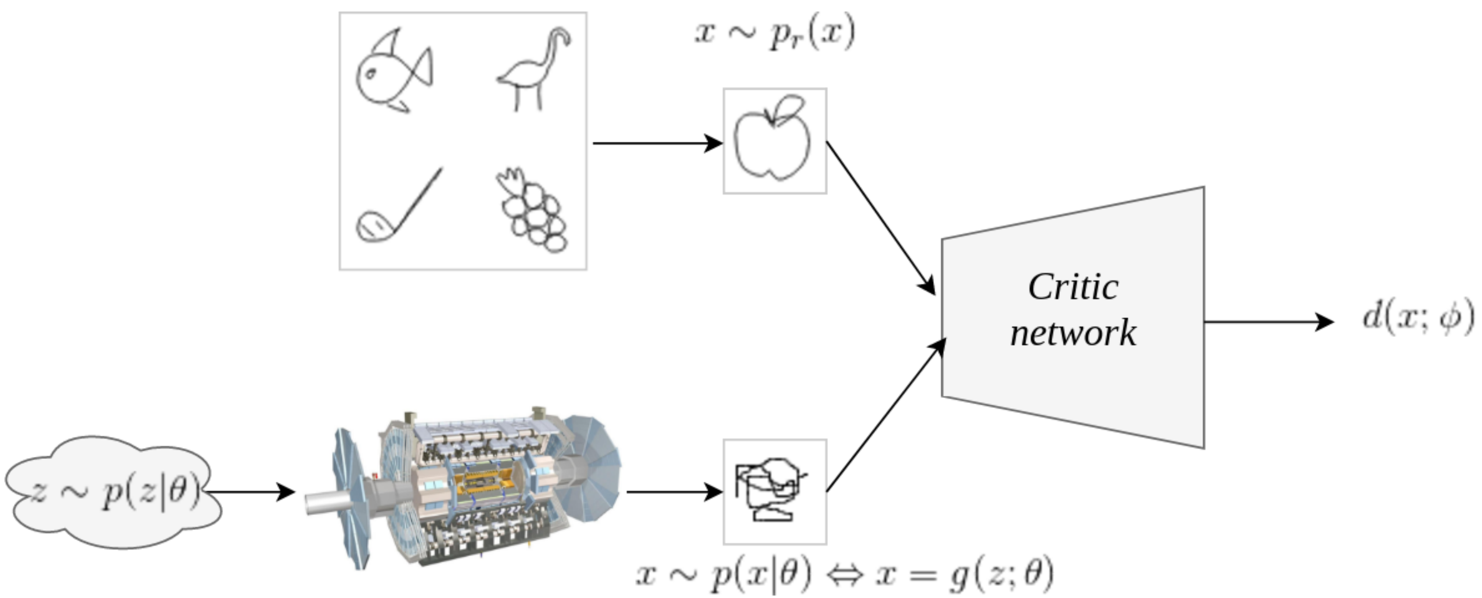
Odena et al
2016

Miyato et al
2017

Zhang et al
2018

Brock et al
2018

AVO



Replace g with an actual scientific simulator!

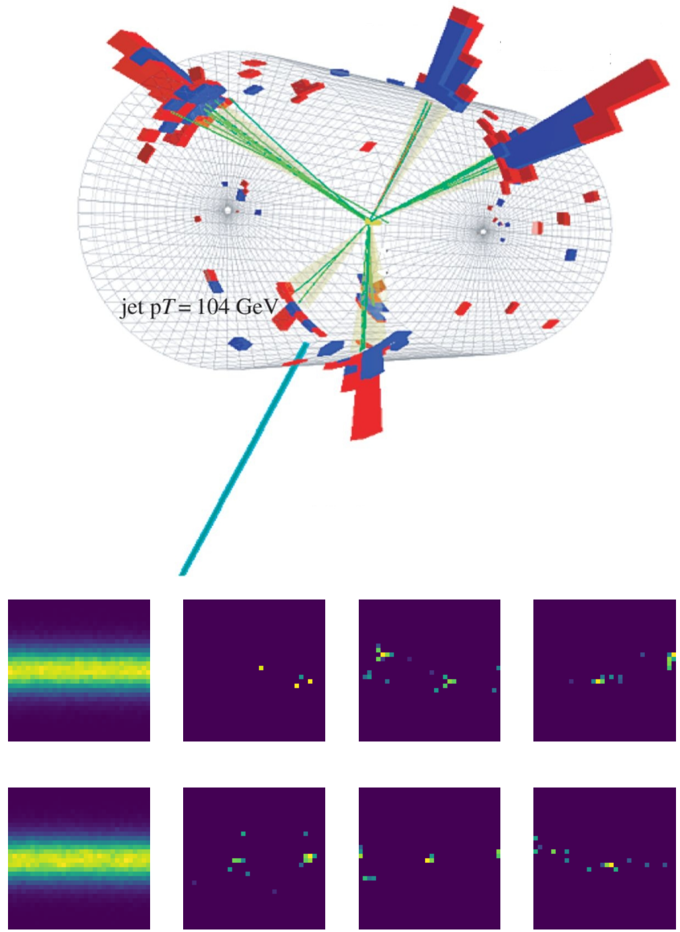
Louppe et al, 2017 (arXiv:1707.07113)

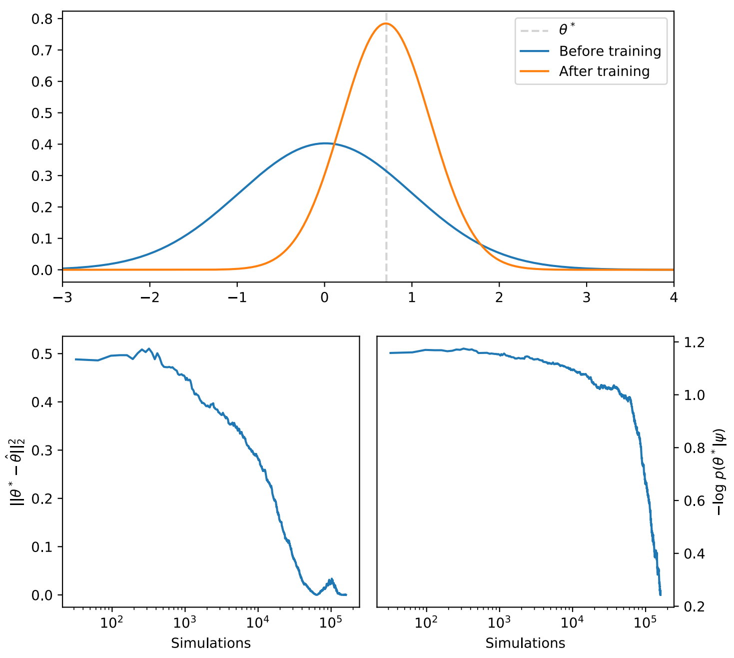
Key insights

- 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.



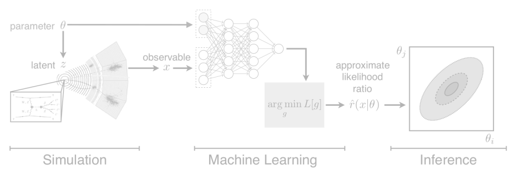


Samples for $\theta = 0$ (top) vs.
samples for $\theta = 0.81$ (bottom).

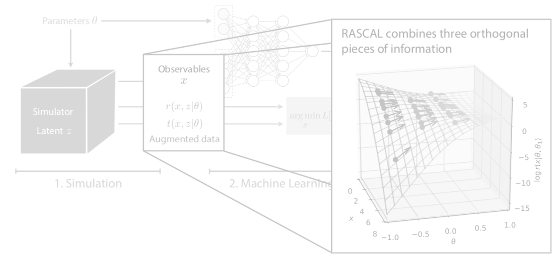
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

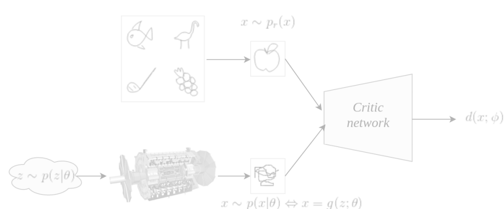


Histograms of observables
Neural density (ratio) estimation

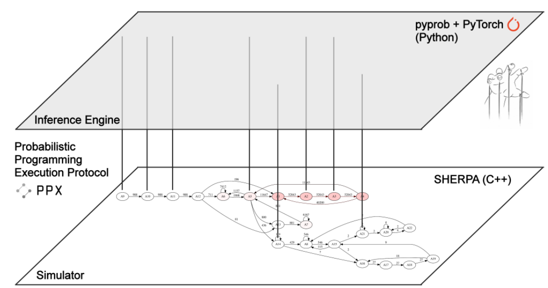


Mining gold from implicit models

Learn to control the simulator



Adversarial variational optimization



Probabilistic programming

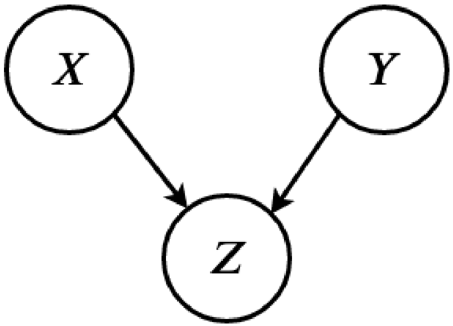
Probabilistic programming

Probabilistic models define a set of random variables and their relationships.

- Observed variables
- Unobserved (hidden, latent) variables

Probabilistic graphical models use graphs to express conditional dependence.

- Bayesian networks
- Markov random fields



$$p(x, y, z) = p(x)p(y)p(z|x, y)$$

Probabilistic programming extends this to ordinary programming with two added constructs:

- Sampling from distributions
- Conditioning random variables by specifying observed values

Example

```
bool c1, c2;
c1 = Bernoulli(0.5);
c2 = Bernoulli(0.5);
observe(c1 || c2);
return(c1, c2);
```

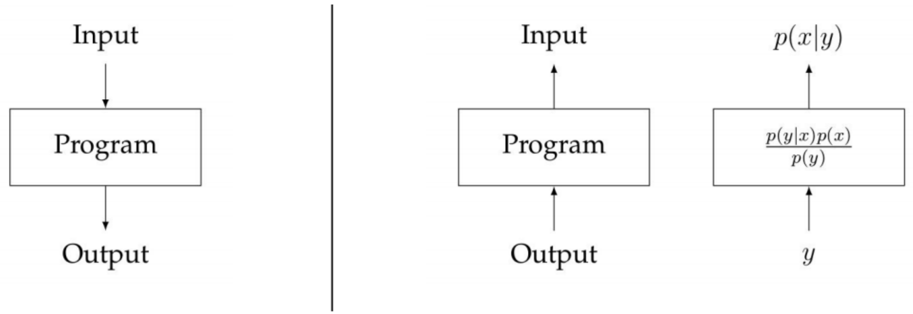
Inference

With a probabilistic program, we define a joint distribution of unobserved and observed variables $p(x, y)$.

Inference engines give us distributions over unobserved variables, given observed variables (data)

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

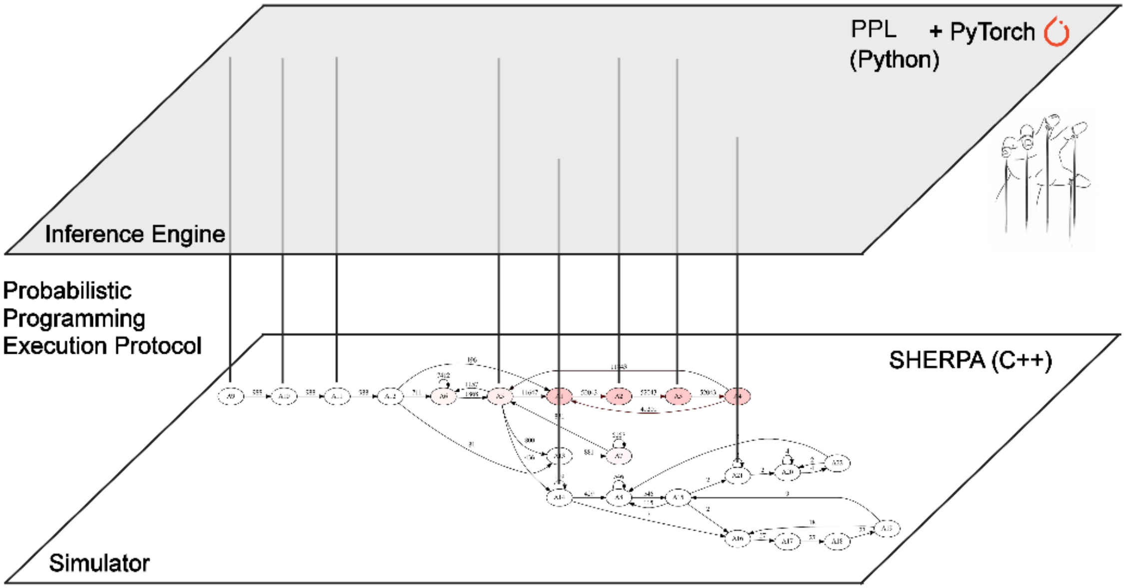
Ordinary program



Probabilistic program

Credits: Atılım Güneş Baydin.

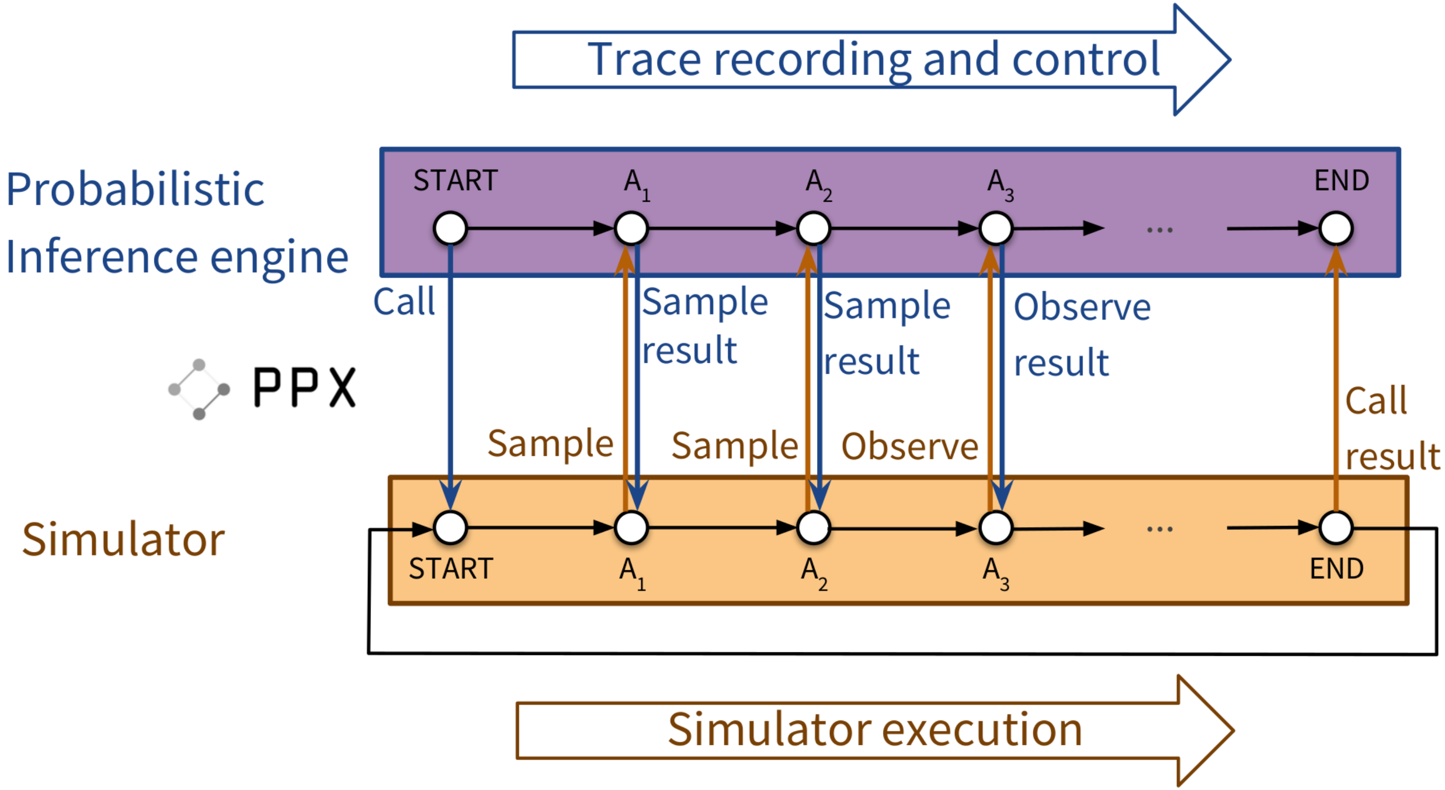
A stochastic simulator implicitly defines a probability distribution by sampling pseudo-random numbers. **Scientific simulators are probabilistic programs!**.

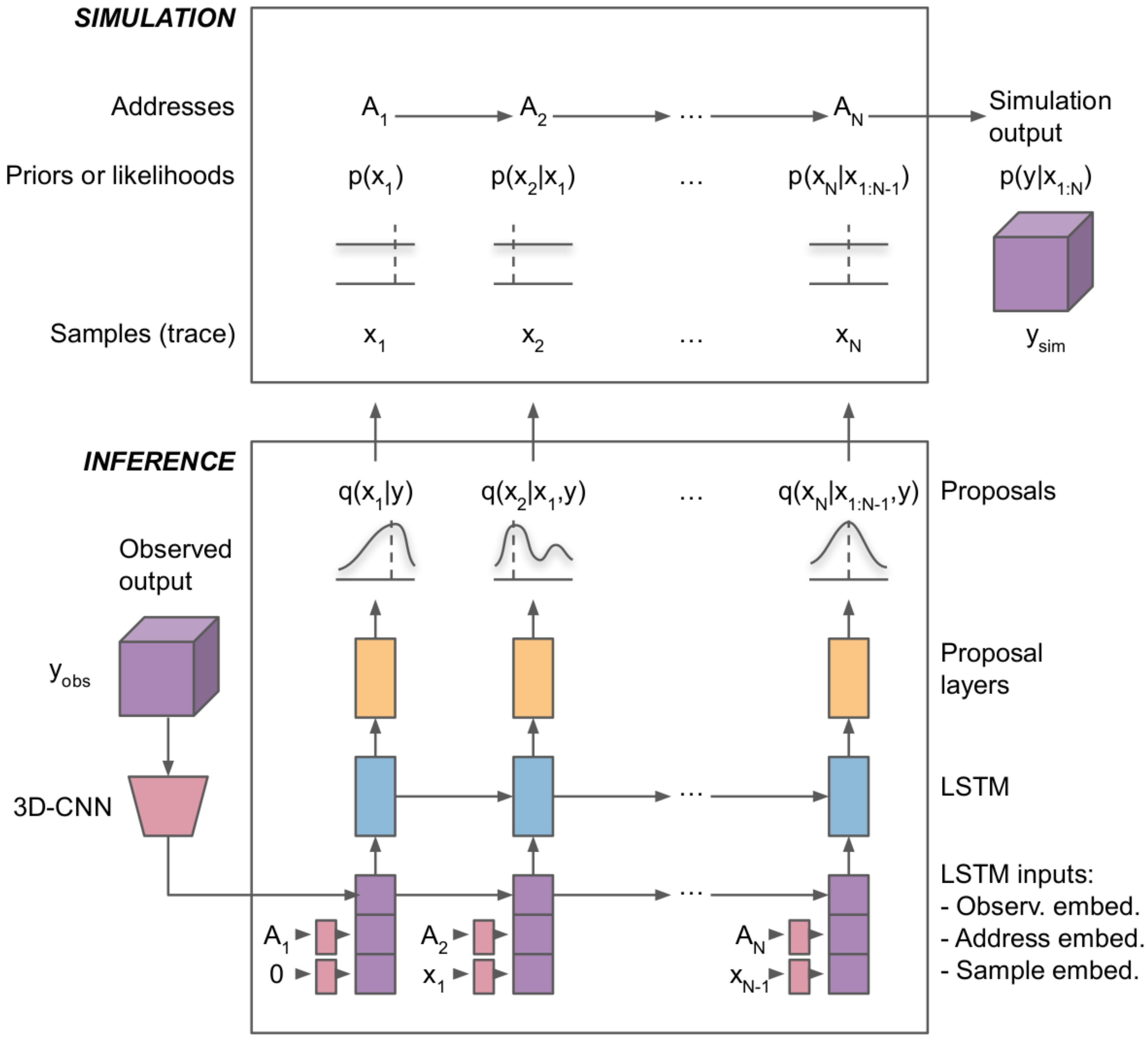


Key insights

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

Le et al, 2016 (arXiv:1610.09900); Baydin et al, 2018 (arXiv:1807.07706).

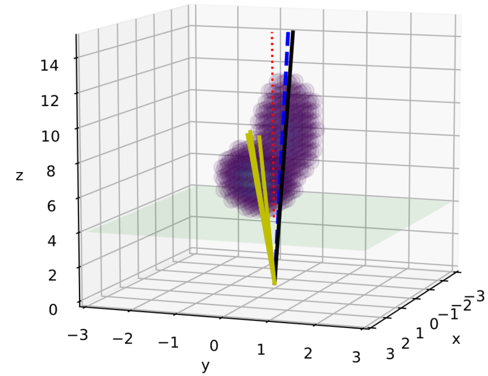

Trace recording and control
Probabilistic
Inference engine
PPX
START
A_1
A_2
A_3
...
END
Call
Sample
result
Sample
result
Observe
result
Call
result
Sample
Sample
Observe
Simulator
START
A_1
A_2
A_3
...
END
Simulator execution

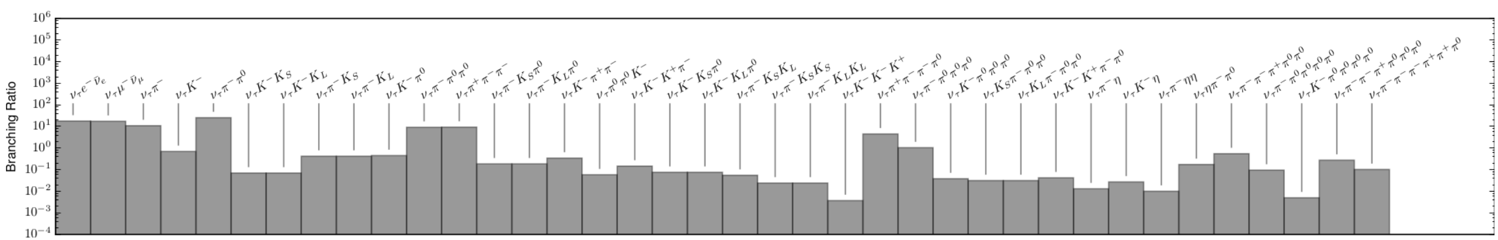

SIMULATION
Addresses
A_1
A_2
...
A_N
Simulation output
Priors or likelihoods
$p(x_1)$
$p(x_2|x_1)$
$p(x_N|x_{1:N-1})$
$p(y|x_{1:N})$
Samples (trace)
x_1
x_2
x_N
y_{sim}
INFERENCE
$q(x_1|y)$
$q(x_2|x_1,y)$
$q(x_N|x_{1:N-1},y)$
Proposals
Observed output
y_{obs}
Proposal layers
LSTM
3D-CNN
LSTM inputs:
- Observ. embed.
- Address embed.
- Sample embed.
A_1
0
A_2
x_1
A_N
x_{N-1}

Taking control of Sherpa

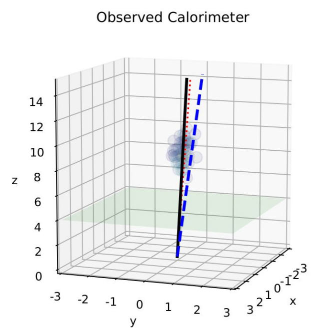
Experimental setup

- τ decay in Sherpa, 38 decay channels, coupled with an approximate calorimeter simulation in C++.
- Observations are 3D calorimeter depositions.
- Latent variables (Monte Carlo truth) of interest: decay channel, px, py, pz momenta, final state momenta and IDs.

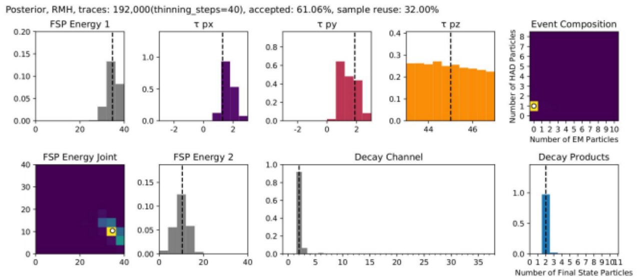




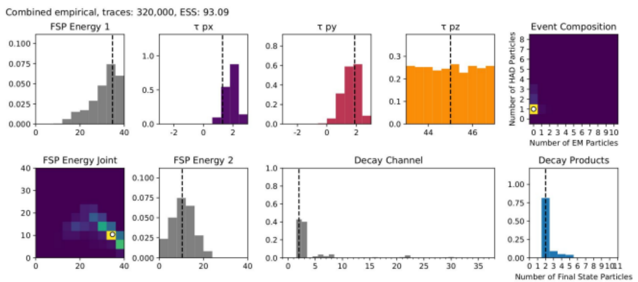
Inference results



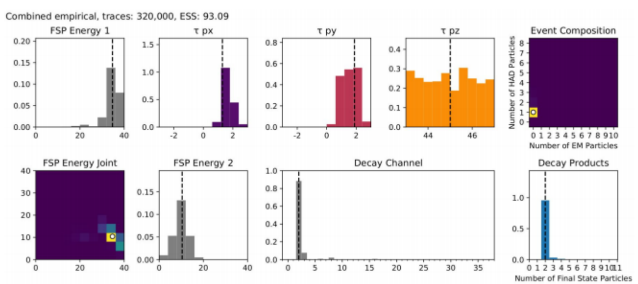
MCMC true posterior (7.7M single node)



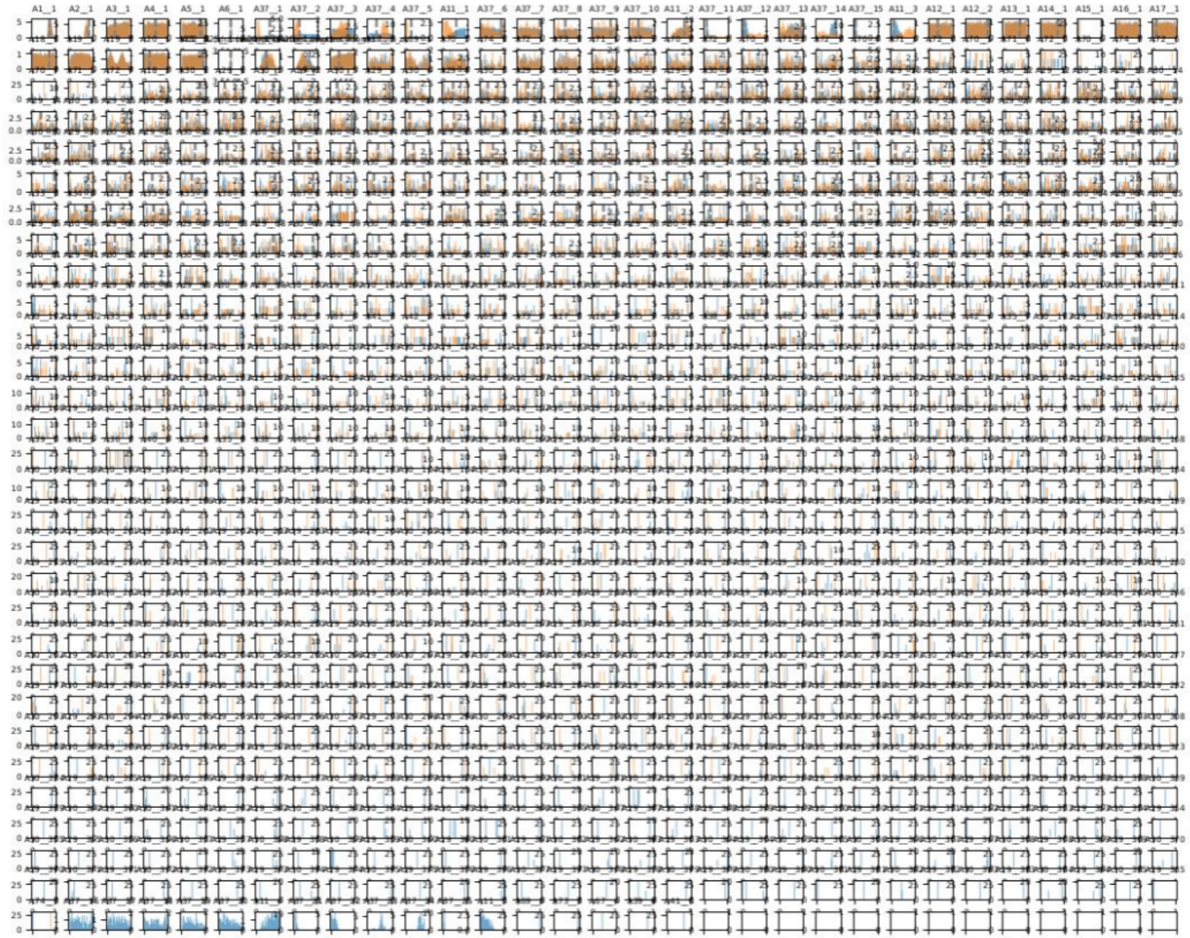
IC proposal
from trained NN



IC posterior
after importance weighting



Credits: Atılım Güneş Baydin.

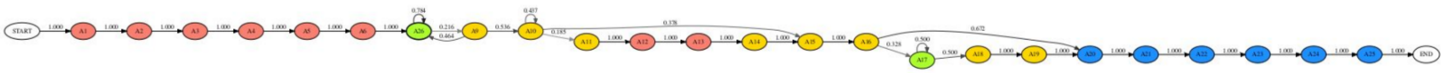


We obtain posteriors over the whole Sherpa address space, 1000s of addresses.

Credits: Atılım Güneş Baydin.

Interpretability

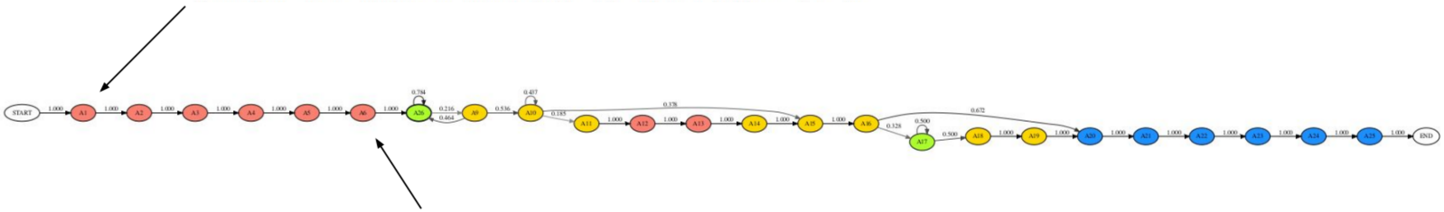
Latent probabilistic structure of the 10 most frequent trace types:



Interpretability

Latent probabilistic structure of the 10 most frequent trace types:

[forward(xt:: xarray_container<xt:: uvector<double, std:: allocator<double> >, (xt:: layout_type)1, xt:: svector<unsigned long, 4ul, std:: allocator<unsigned long>, true>, xt:: xtensor_expression_tag>)+0x5f; SherpaGenerator:: Generate()+0x36; SHERPA:: Sherpa:: GenerateOneEvent(bool)+0x2fa; SHERPA:: Event_Handler:: GenerateEvent(SHERPA:: eventtype:: code)+0x44d; SHERPA:: Event_Handler:: GenerateHadronDecayEvent(SHERPA:: eventtype:: code&)+0x45f; ATOOLS:: Random:: Get(bool, bool)+0x1d5; probprog_RNG:: Get(bool, bool)+0xf9]_Uniform_1

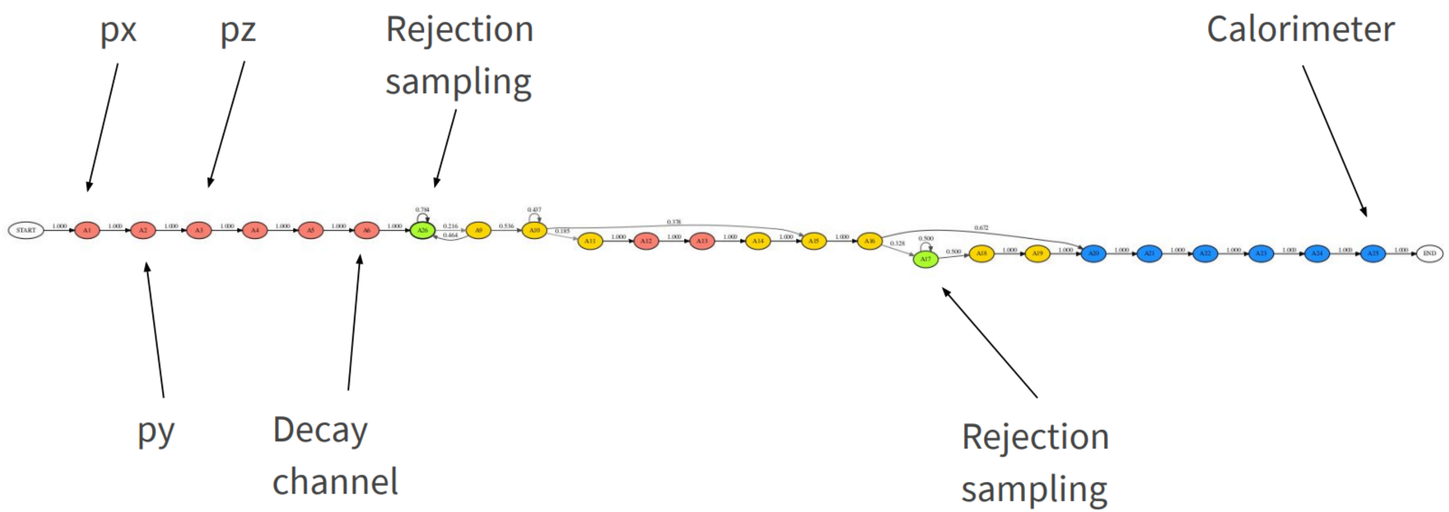


[forward(xt:: xarray_container<xt:: uvector<double, std:: allocator<double> >, (xt:: layout_type)1, xt:: svector<unsigned long, 4ul, std:: allocator<unsigned long>, true>, xt:: xtensor_expression_tag>)+0x5f; SherpaGenerator:: Generate()+0x36; SHERPA:: Sherpa:: GenerateOneEvent(bool)+0x2fa; SHERPA:: Event_Handler:: GenerateEvent(SHERPA:: eventtype:: code)+0x44d; SHERPA:: Event_Handler:: GenerateHadronDecayEvent(SHERPA:: eventtype:: code&)+0x982; SHERPA:: Event_Handler:: IterateEventPhases(SHERPA:: eventtype:: code&, double&)+0x1d2; SHERPA:: Hadron_Decays:: Treat(ATOOLS:: Blob_List*, double&)+0x975; SHERPA:: Decay_Handler_Base:: TreatInitialBlob(ATOOLS:: Blob*, METOOLS:: Amplitude2_Tensor*, std:: vector<ATOOLS:: Particle*, std:: allocator<ATOOLS:: Particle*> > const&)+0x1ab1; SHERPA:: Hadron_Decay_Handler:: CreateDecayBlob(ATOOLS:: Particle*)+0x4cd; PHASIC:: Decay_Table:: Select() const+0x9d7; ATOOLS:: Random:: GetCategorical(std:: vector<double, std:: allocator<double> > const&, bool, bool)+0x1a5; probprog_RNG:: GetCategorical(std:: vector<double, std:: allocator<double> > const&, bool, bool)+0x111]_Categorical(length_categories:38)_1

Credits: Atılım Güneş Baydin.

Interpretability

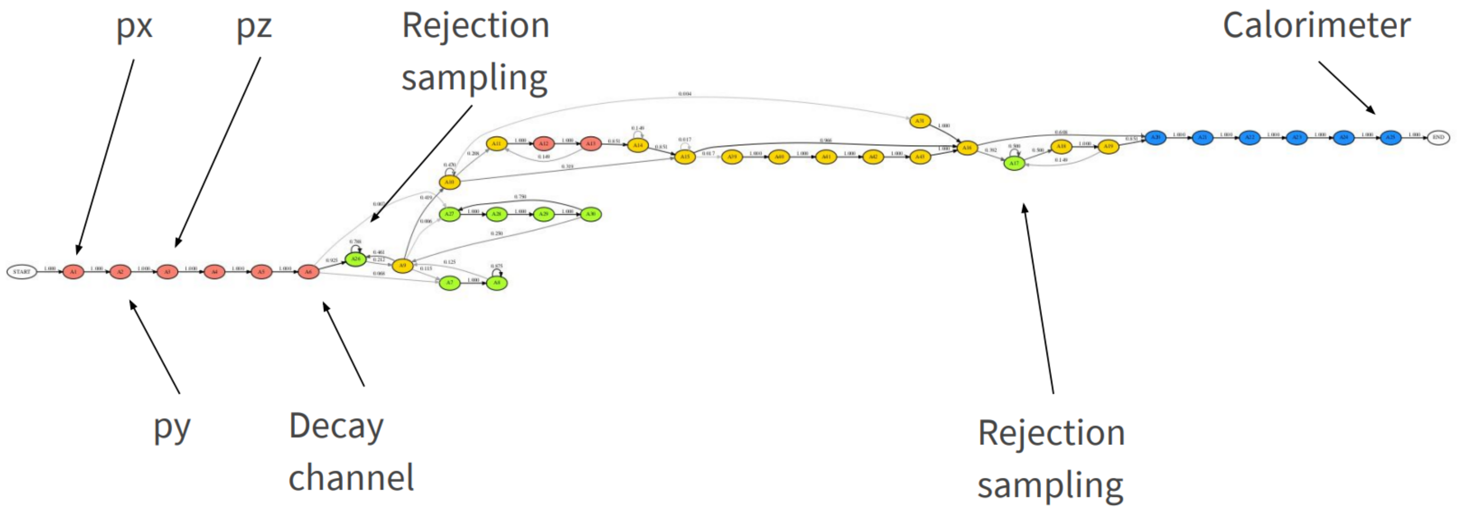
Latent probabilistic structure of the 10 most frequent trace types:



Credits: Atılım Güneş Baydin.

Interpretability

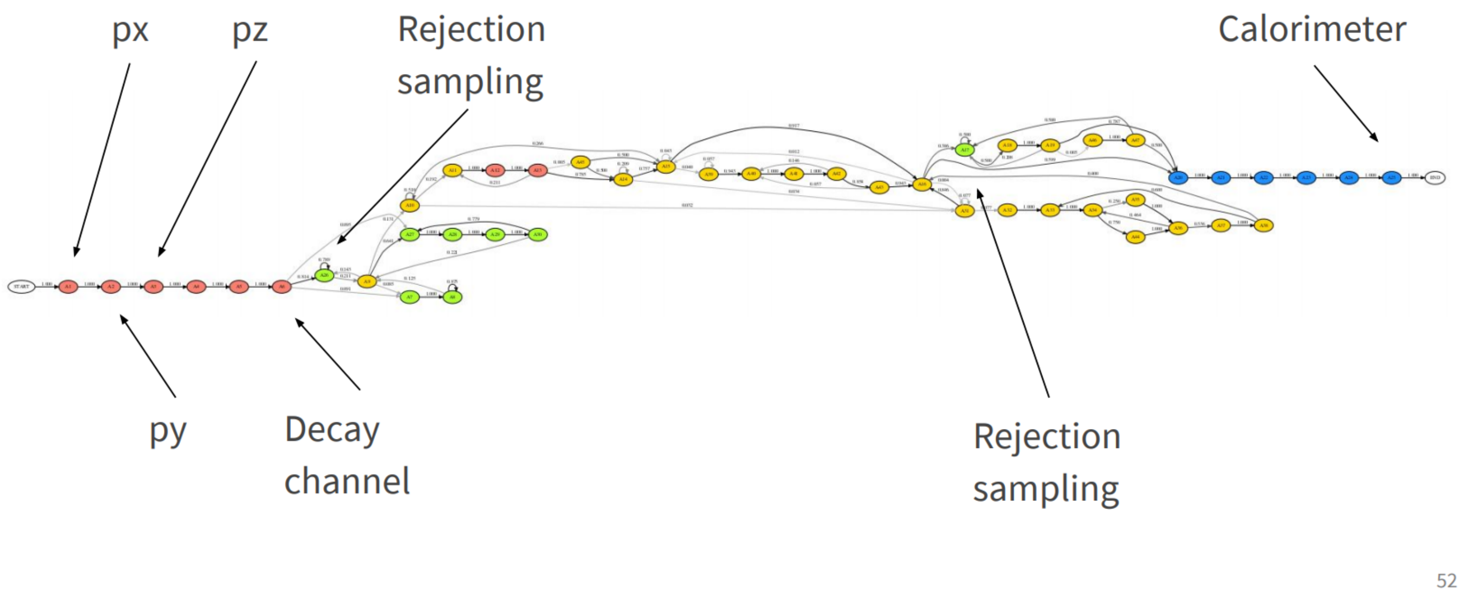
Latent probabilistic structure of the 25 most frequent trace types:



Credits: Atılım Güneş Baydin.

Interpretability

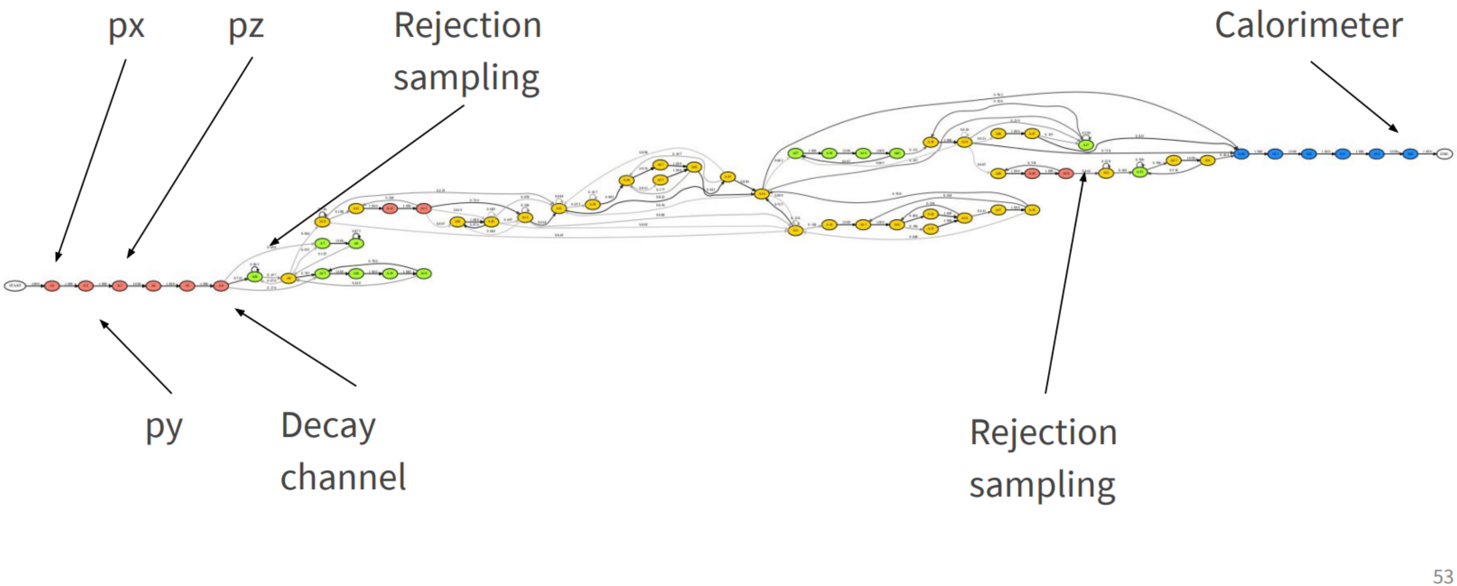
Latent probabilistic structure of the 100 most frequent trace types:



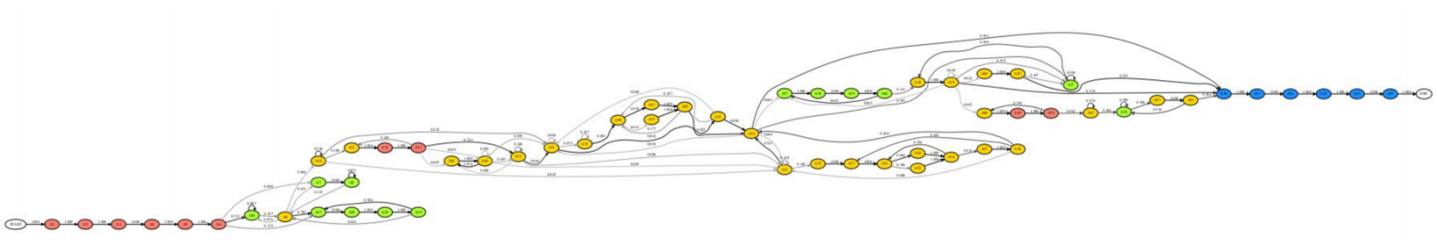
Credits: Atılım Güneş Baydin.

Interpretability

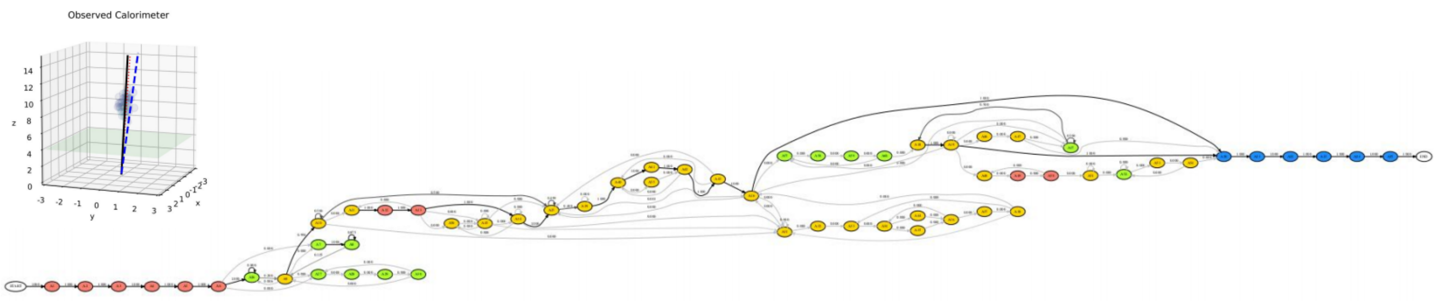
Latent probabilistic structure of the 250 most frequent trace types:



Interpretability



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



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

Summary

Summary

- Much of modern science is based on "likelihood-free" simulations.
- The likelihood-ratio is central to many statistical inference procedures.
- Supervised learning enables likelihood-ratio estimation.
- Better likelihood-ratio estimates can be achieved by mining simulators.
- Probabilistic programming enables posterior inference in scientific simulators.

Collaborators


Kyle Cranmer


Juan Pavez


Johann Brehmer


Joeri Hermans


Lukas Heinrich


Atılım Güneş Baydin


Wahid Bhimji


Frank Wood

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., Hermans, J., & 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.