Parameter inference and data modelling with deep learning

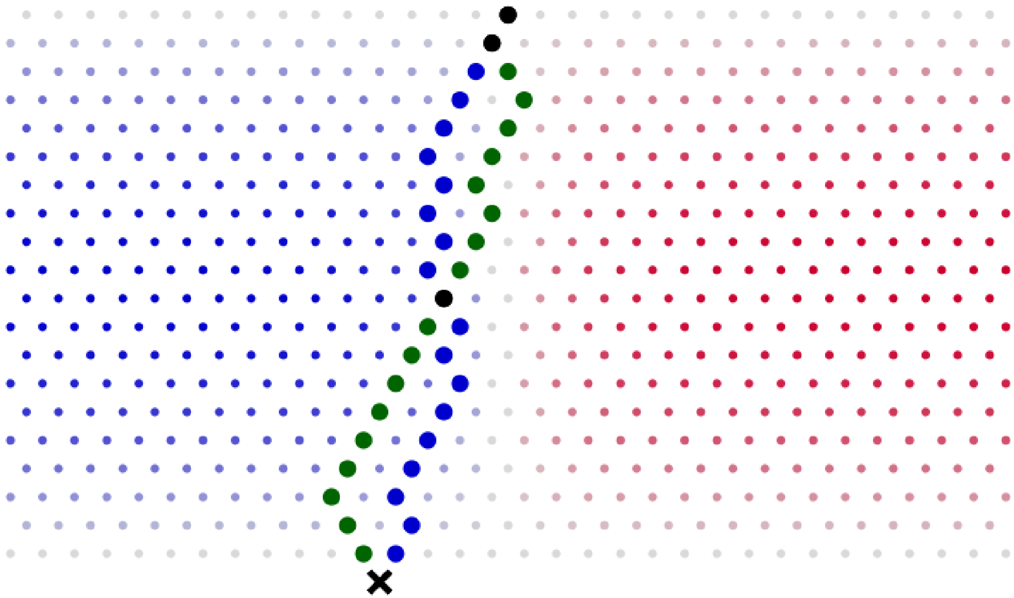
Flexible operation and advanced control workshop

Isaac Newton Institute
January 8, 2019

Gilles Louppe
@glouppe

LIÈGE université


@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?

$$p(x|\theta) = \underbrace{\int}_{\text{intractable!}} p(x, z|\theta) dz$$

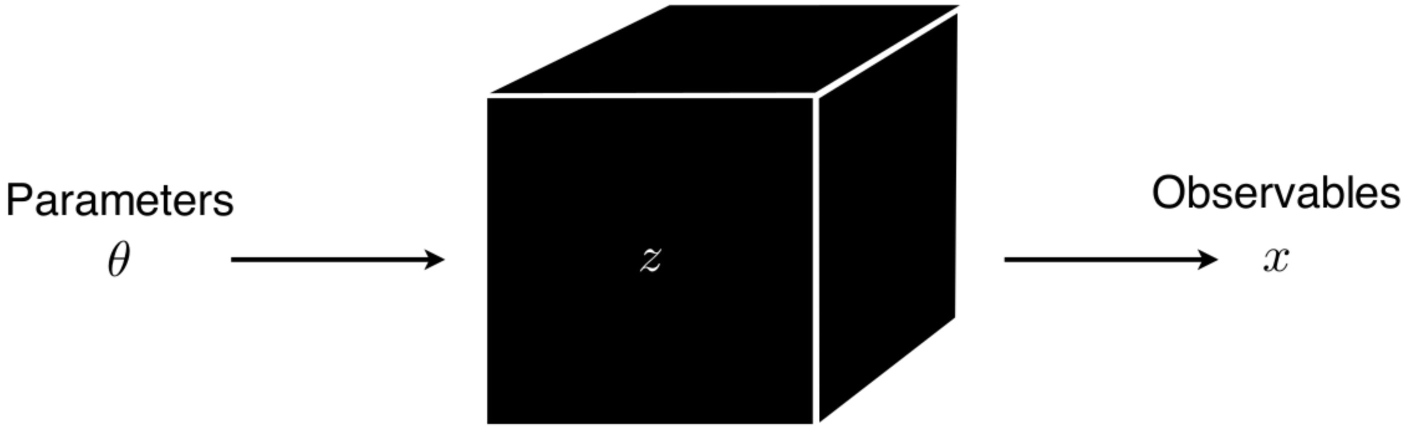
$$\neq \binom{n}{x} \theta^x (1-\theta)^{n-x}$$

Does this mean inference is no longer 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)

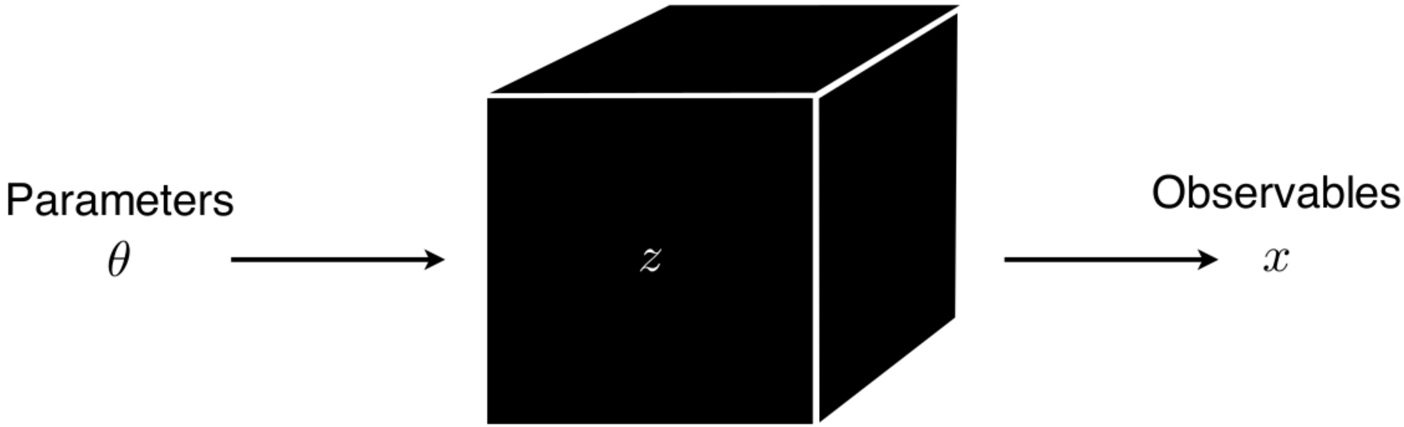
Inference in this context requires likelihood-free algorithms.



Prediction (simulation):

- Well-understood mechanistic model
- Simulator can generate samples

Credits: Johann Brehmer

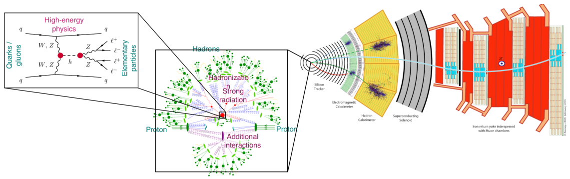


Prediction (simulation):
- Well-understood mechanistic model
- Simulator can generate samples

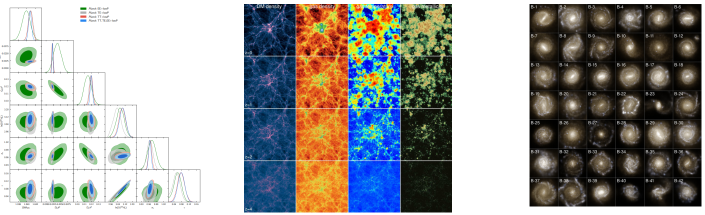
Inference:
- Likelihood function $p(x|\theta)$ is intractable
- Goal: estimator $\hat{p}(x|\theta)$

Credits: Johann Brehmer

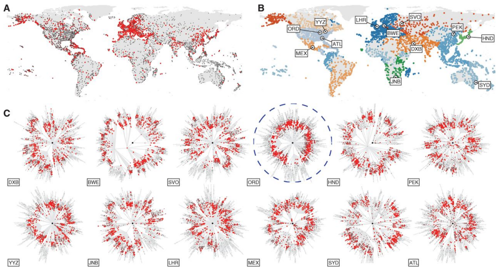
Applications



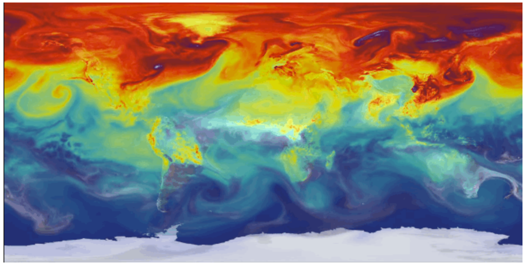
Particle physics



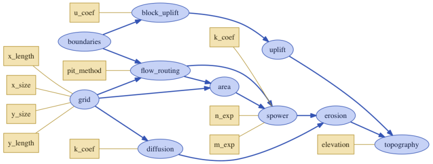
Cosmology

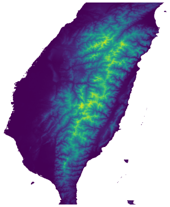


Epidemiology



Climatology

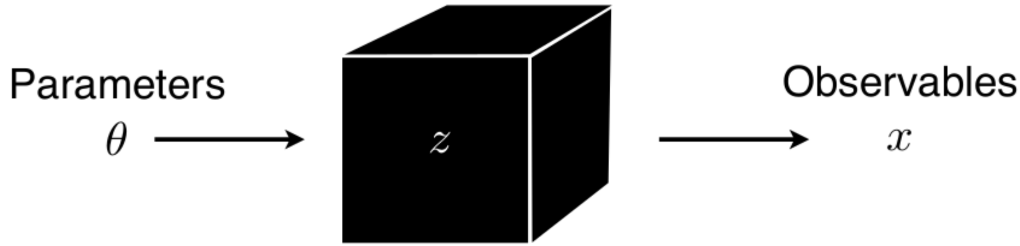




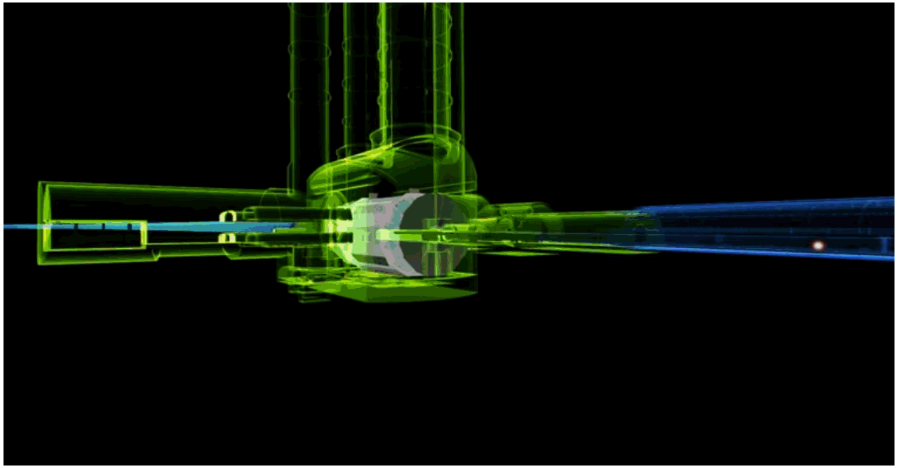
Computational topography

Astronomy

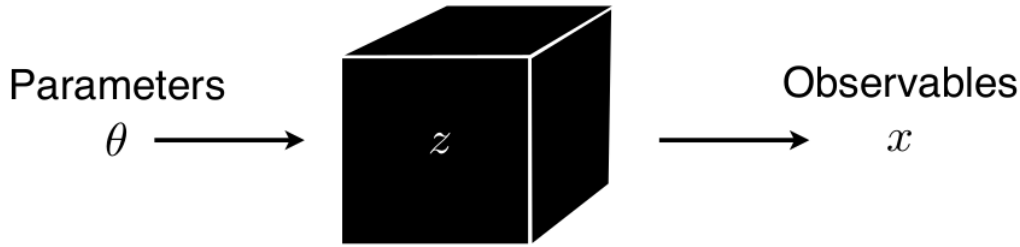
Particle physics



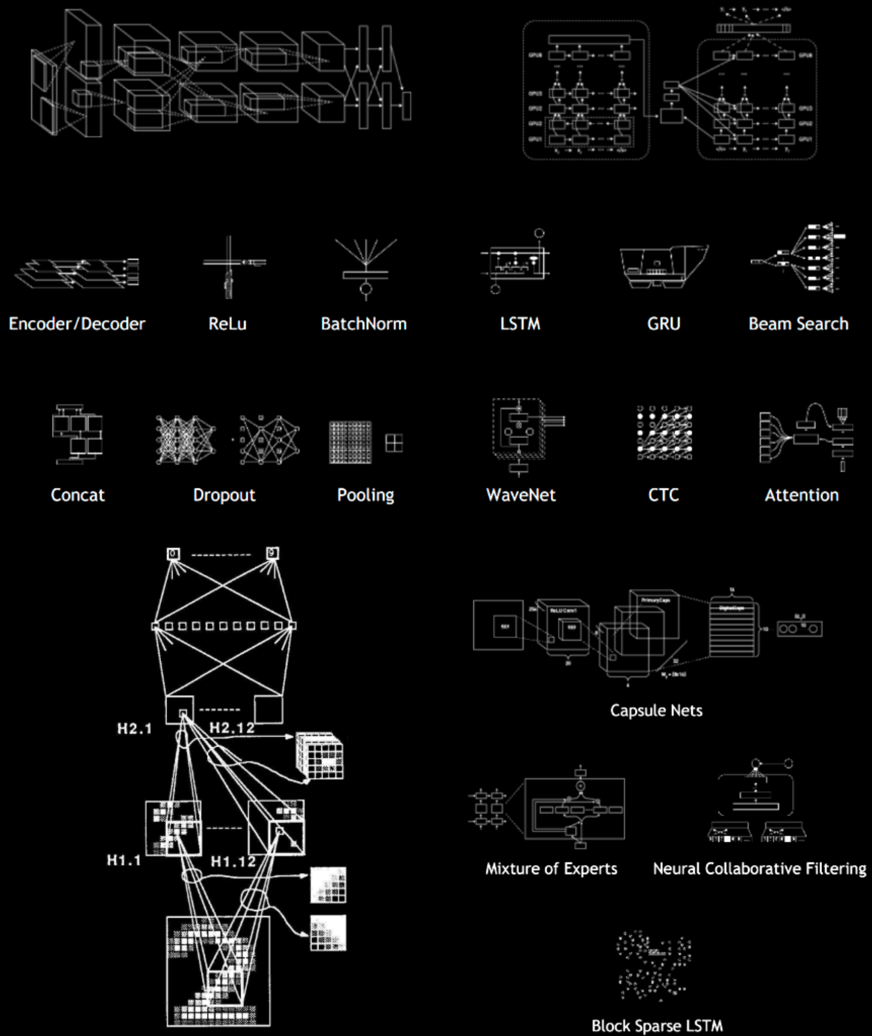
$$
\begin{gathered}
\mathcal{L}_{SM} = -\tfrac{1}{2}\partial_\nu g^a_\mu \partial_\nu g^a_\mu - g_s f^{abc}\partial_\mu g^a_\nu g^b_\mu g^c_\nu - \tfrac{1}{4}g^2_s f^{abc}f^{ade}g^b_\mu g^c_\nu g^d_\mu g^e_\nu - \partial_\nu W^+_\mu \partial_\nu W^-_\mu - \\
M^2 W^+_\mu W^-_\mu - \tfrac{1}{2}\partial_\nu Z^0_\mu \partial_\nu Z^0_\mu - \tfrac{1}{2c^2_w}M^2 Z^0_\mu Z^0_\mu - \tfrac{1}{2}\partial_\mu A_\nu \partial_\mu A_\nu - igc_w(\partial_\nu Z^0_\mu(W^+_\mu W^-_\nu - \\
W^+_\nu W^-_\mu) - Z^0_\nu(W^+_\mu \partial_\nu W^-_\mu - W^-_\mu \partial_\nu W^+_\mu) + Z^0_\mu(W^+_\nu \partial_\nu W^-_\mu - W^-_\nu \partial_\nu W^+_\mu)) - \\
igs_w(\partial_\nu A_\mu(W^+_\mu W^-_\nu - W^+_\nu W^-_\mu) - A_\nu(W^+_\mu \partial_\nu W^-_\mu - W^-_\mu \partial_\nu W^+_\mu) + A_\mu(W^+_\nu \partial_\nu W^-_\mu - \\
W^-_\nu \partial_\nu W^+_\mu)) - \tfrac{1}{2}g^2 W^+_\mu W^-_\mu W^+_\nu W^-_\nu + \tfrac{1}{2}g^2 W^+_\mu W^-_\nu W^+_\mu W^-_\nu + g^2 c^2_w(Z^0_\mu W^+_\mu Z^0_\nu W^-_\nu - \\
Z^0_\mu Z^0_\mu W^+_\nu W^-_\nu) + g^2 s^2_w(A_\mu W^+_\mu A_\nu W^-_\nu - A_\mu A_\mu W^+_\nu W^-_\nu) + g^2 s_w c_w(A_\mu Z^0_\nu(W^+_\mu W^-_\nu - \\
W^+_\nu W^-_\mu) - 2A_\mu Z^0_\mu W^+_\nu W^-_\nu) - \tfrac{1}{2}\partial_\mu H \partial_\mu H - 2M^2\alpha_h H^2 - \partial_\mu \phi^+ \partial_\mu \phi^- - \tfrac{1}{2}\partial_\mu \phi^0 \partial_\mu \phi^0 - \\
\beta_h\left(\tfrac{2M^2}{g^2} + \tfrac{2M}{g}H + \tfrac{1}{2}(H^2 + \phi^0\phi^0 + 2\phi^+\phi^-)\right) + \tfrac{2M^4}{g^2}\alpha_h - \\
g\alpha_h M\left(H^3 + H\phi^0\phi^0 + 2H\phi^+\phi^-\right) - \\
\tfrac{1}{8}g^2\alpha_h\left(H^4 + (\phi^0)^4 + 4(\phi^+\phi^-)^2 + 4(\phi^0)^2\phi^+\phi^- + 4H^2\phi^+\phi^- + 2(\phi^0)^2H^2\right) - \\
gMW^+_\mu W^-_\mu H - \tfrac{1}{2}g\tfrac{M}{c^2_w}Z^0_\mu Z^0_\mu H - \\
\tfrac{1}{2}ig\left(W^+_\mu(\phi^0\partial_\mu\phi^- - \phi^-\partial_\mu\phi^0) - W^-_\mu(\phi^0\partial_\mu\phi^+ - \phi^+\partial_\mu\phi^0)\right) + \\
\tfrac{1}{2}g\left(W^+_\mu(H\partial_\mu\phi^- - \phi^-\partial_\mu H) + W^-_\mu(H\partial_\mu\phi^+ - \phi^+\partial_\mu H)\right) + \tfrac{1}{2}g\tfrac{1}{c_w}(Z^0_\mu(H\partial_\mu\phi^0 - \phi^0\partial_\mu H) + \\
M\left(\tfrac{1}{c_w}Z^0_\mu\partial_\mu\phi^0 + W^+_\mu\partial_\mu\phi^- + W^-_\mu\partial_\mu\phi^+\right) - ig\tfrac{s^2_w}{c_w}MZ^0_\mu(W^+_\mu\phi^- - W^-_\mu\phi^+) + igs_w MA_\mu(W^+_\mu\phi^- - \\
W^-_\mu\phi^+) - ig\tfrac{1-2c^2_w}{2c_w}Z^0_\mu(\phi^+\partial_\mu\phi^- - \phi^-\partial_\mu\phi^+) + igs_w A_\mu(\phi^+\partial_\mu\phi^- - \phi^-\partial_\mu\phi^+) - \\
\tfrac{1}{4}g^2 W^+_\mu W^-_\mu\left(H^2 + (\phi^0)^2 + 2\phi^+\phi^-\right) - \tfrac{1}{8}g^2\tfrac{1}{c^2_w}Z^0_\mu Z^0_\mu\left(H^2 + (\phi^0)^2 + 2(2s^2_w - 1)^2\phi^+\phi^-\right) - \\
\tfrac{1}{2}g^2\tfrac{s^2_w}{c_w}Z^0_\mu\phi^0(W^+_\mu\phi^- + W^-_\mu\phi^+) - \tfrac{1}{2}ig^2\tfrac{s^2_w}{c_w}Z^0_\mu H(W^+_\mu\phi^- - W^-_\mu\phi^+) + \tfrac{1}{2}g^2 s_w A_\mu\phi^0(W^+_\mu\phi^- + \\
W^-_\mu\phi^+) + \tfrac{1}{2}ig^2 s_w A_\mu H(W^+_\mu\phi^- - W^-_\mu\phi^+) - g^2\tfrac{s_w}{c_w}(2c^2_w - 1)Z^0_\mu A_\mu\phi^+\phi^- - \\
g^2 s^2_w A_\mu A_\mu\phi^+\phi^- + \tfrac{1}{2}ig_s\lambda^a_{ij}(\bar{q}^\sigma_i\gamma^\mu q^\sigma_j)g^a_\mu - \bar{e}^\lambda(\gamma\partial + m^\lambda_e)e^\lambda - \bar{\nu}^\lambda(\gamma\partial + m^\lambda_\nu)\nu^\lambda - \bar{u}^\lambda_j(\gamma\partial + \\
m^\lambda_u)u^\lambda_j - \bar{d}^\lambda_j(\gamma\partial + m^\lambda_d)d^\lambda_j + igs_w A_\mu\left(-(\bar{e}^\lambda\gamma^\mu e^\lambda) + \tfrac{2}{3}(\bar{u}^\lambda_j\gamma^\mu u^\lambda_j) - \tfrac{1}{3}(\bar{d}^\lambda_j\gamma^\mu d^\lambda_j)\right) + \\
\tfrac{ig}{4c_w}Z^0_\mu\{(\bar{\nu}^\lambda\gamma^\mu(1+\gamma^5)\nu^\lambda) + (\bar{e}^\lambda\gamma^\mu(4s^2_w - 1 - \gamma^5)e^\lambda) + (\bar{d}^\lambda_j\gamma^\mu(\tfrac{4}{3}s^2_w - 1 - \gamma^5)d^\lambda_j) + \\
(\bar{u}^\lambda_j\gamma^\mu(1 - \tfrac{8}{3}s^2_w + \gamma^5)u^\lambda_j)\} + \tfrac{ig}{2\sqrt{2}}W^+_\mu\left((\bar{\nu}^\lambda\gamma^\mu(1+\gamma^5)U^{lep}{}_{\lambda\kappa}e^\kappa) + (\bar{u}^\lambda_j\gamma^\mu(1+\gamma^5)C_{\lambda\kappa}d^\kappa_j)\right) + \\
\tfrac{ig}{2\sqrt{2}}W^-_\mu\left((\bar{e}^\kappa U^{lep\dagger}{}_{\kappa\lambda}\gamma^\mu(1+\gamma^5)\nu^\lambda) + (\bar{d}^\kappa_j C^\dagger_{\kappa\lambda}\gamma^\mu(1+\gamma^5)u^\lambda_j)\right) + \\
\tfrac{ig}{2M\sqrt{2}}\phi^+\left(-m^\kappa_e(\bar{\nu}^\lambda U^{lep}{}_{\lambda\kappa}(1-\gamma^5)e^\kappa) + m^\lambda_\nu(\bar{\nu}^\lambda U^{lep}{}_{\lambda\kappa}(1+\gamma^5)e^\kappa\right) + \\
\tfrac{ig}{2M\sqrt{2}}\phi^-\left(m^\lambda_e(\bar{e}^\lambda U^{lep\dagger}{}_{\lambda\kappa}(1+\gamma^5)\nu^\kappa) - m^\kappa_\nu(\bar{e}^\lambda U^{lep\dagger}{}_{\lambda\kappa}(1-\gamma^5)\nu^\kappa\right) - \tfrac{g}{2}\tfrac{m^\lambda_\nu}{M}H(\bar{\nu}^\lambda\nu^\lambda) - \\
\tfrac{g}{2}\tfrac{m^\lambda_e}{M}H(\bar{e}^\lambda e^\lambda) + \tfrac{ig}{2}\tfrac{m^\lambda_\nu}{M}\phi^0(\bar{\nu}^\lambda\gamma^5\nu^\lambda) - \tfrac{ig}{2}\tfrac{m^\lambda_e}{M}\phi^0(\bar{e}^\lambda\gamma^5 e^\lambda) - \tfrac{1}{4}\bar{\nu}_\lambda M^R_{\lambda\kappa}(1-\gamma_5)\hat{\nu}_\kappa - \\
\tfrac{1}{4}\bar{\nu}_\lambda M^R_{\lambda\kappa}(1-\gamma_5)\hat{\nu}_\kappa + \tfrac{ig}{2M\sqrt{2}}\phi^+\left(-m^\kappa_d(\bar{u}^\lambda_j C_{\lambda\kappa}(1-\gamma^5)d^\kappa_j) + m^\lambda_u(\bar{u}^\lambda_j C_{\lambda\kappa}(1+\gamma^5)d^\kappa_j\right) + \\
\tfrac{ig}{2M\sqrt{2}}\phi^-\left(m^\lambda_d(\bar{d}^\lambda_j C^\dagger_{\lambda\kappa}(1+\gamma^5)u^\kappa_j) - m^\kappa_u(\bar{d}^\lambda_j C^\dagger_{\lambda\kappa}(1-\gamma^5)u^\kappa_j\right) - \tfrac{g}{2}\tfrac{m^\lambda_u}{M}H(\bar{u}^\lambda_j u^\lambda_j) - \\
\tfrac{g}{2}\tfrac{m^\lambda_d}{M}H(\bar{d}^\lambda_j d^\lambda_j) + \tfrac{ig}{2}\tfrac{m^\lambda_u}{M}\phi^0(\bar{u}^\lambda_j\gamma^5 u^\lambda_j) - \tfrac{ig}{2}\tfrac{m^\lambda_d}{M}\phi^0(\bar{d}^\lambda_j\gamma^5 d^\lambda_j) + \bar{G}^a\partial^2 G^a + g_s f^{abc}\partial_\mu\bar{G}^a G^b g^c_\mu + \\
\bar{X}^+(\partial^2 - M^2)X^+ + \bar{X}^-(\partial^2 - M^2)X^- + \bar{X}^0(\partial^2 - \tfrac{M^2}{c^2_w})X^0 + \bar{Y}\partial^2 Y + igc_w W^+_\mu(\partial_\mu\bar{X}^0 X^- - \\
\partial_\mu\bar{X}^+ X^0) + igs_w W^+_\mu(\partial_\mu\bar{Y}X^- - \partial_\mu\bar{X}^+ Y) + igc_w W^-_\mu(\partial_\mu\bar{X}^- X^0 - \\
\partial_\mu\bar{X}^0 X^+) + igs_w W^-_\mu(\partial_\mu\bar{X}^- Y - \partial_\mu\bar{Y}X^+) + igc_w Z^0_\mu(\partial_\mu\bar{X}^+ X^+ - \\
\partial_\mu\bar{X}^- X^-) + igs_w A_\mu(\partial_\mu\bar{X}^+ X^+ - \\
\partial_\mu\bar{X}^- X^-) - \tfrac{1}{2}gM\left(\bar{X}^+ X^+ H + \bar{X}^- X^- H + \tfrac{1}{c^2_w}\bar{X}^0 X^0 H\right) + \tfrac{1-2c^2_w}{2c_w}igM\left(\bar{X}^+ X^0\phi^+ - \bar{X}^- X^0\phi^-\right) + \\
\tfrac{1}{2c_w}igM\left(\bar{X}^0 X^-\phi^+ - \bar{X}^0 X^+\phi^-\right) + igMs_w\left(\bar{X}^0 X^-\phi^+ - \bar{X}^0 X^+\phi^-\right) + \\
\tfrac{1}{2}igM\left(\bar{X}^+ X^+\phi^0 - \bar{X}^- X^-\phi^0\right).
\end{gathered}
$$

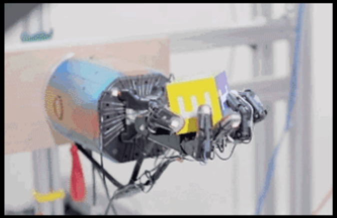


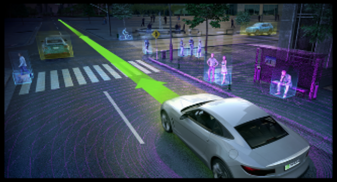
Energy?



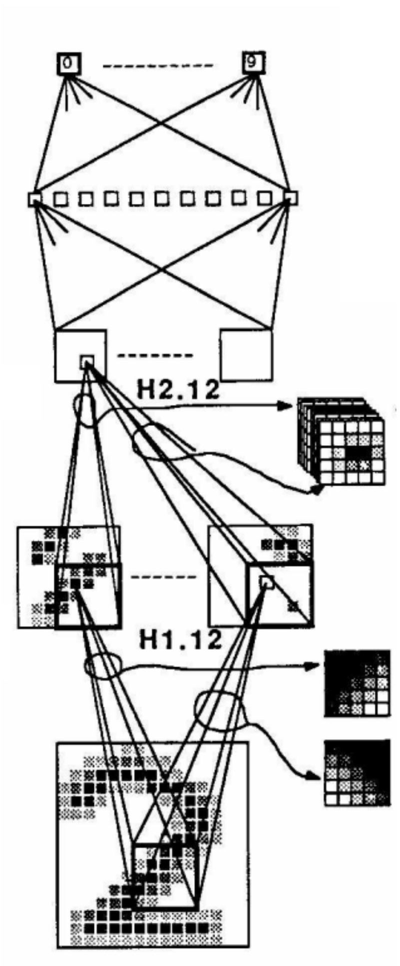
Likelihood-free inference algorithms







Can we harness deep learning for inference and generation?



Neural networks are

- function approximators with a gazillion of parameters,
- tuned with stochastic gradient descent

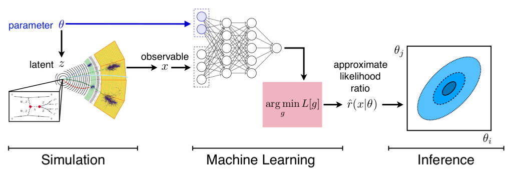
$$\theta_{t+1} = \theta_t - \gamma \hat{\nabla}_\theta \mathcal{L}(\theta_t),$$

- are flexible enough to be structured by domain knowledge.

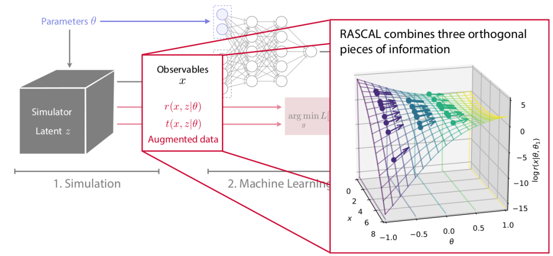
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

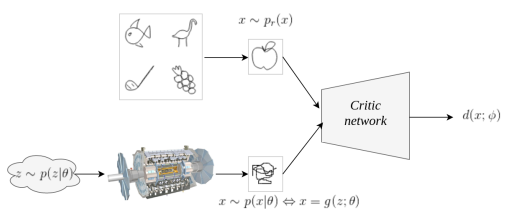


Histograms of observables
Neural density (ratio) estimation

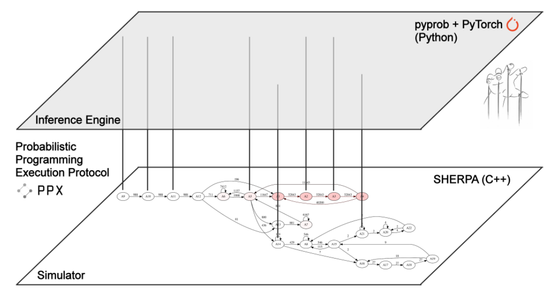


Mining gold from implicit models

Learn to control the simulator



Adversarial variational optimization

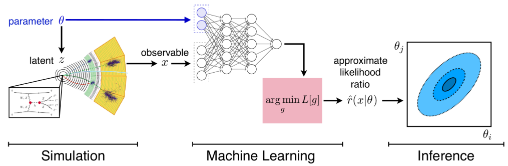


Probabilistic programming

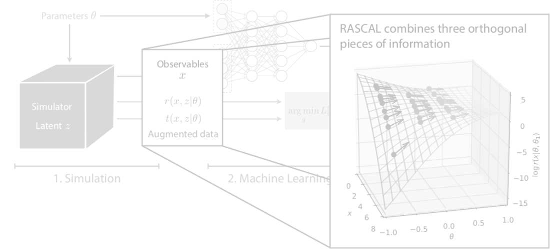
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

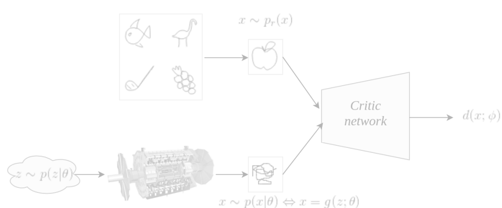


Histograms of observables
Neural density (ratio) estimation

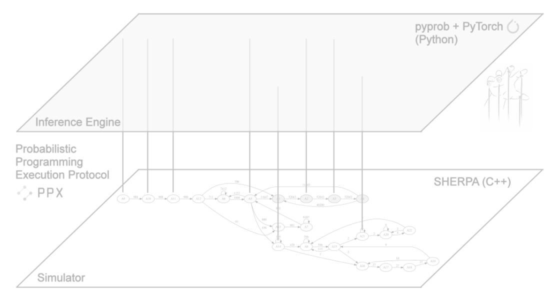


Mining gold from implicit models

Learn to control the simulator



Adversarial variational optimization



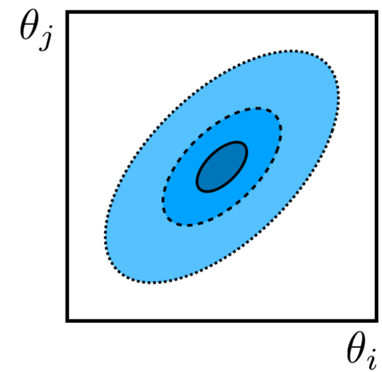
Probabilistic programming

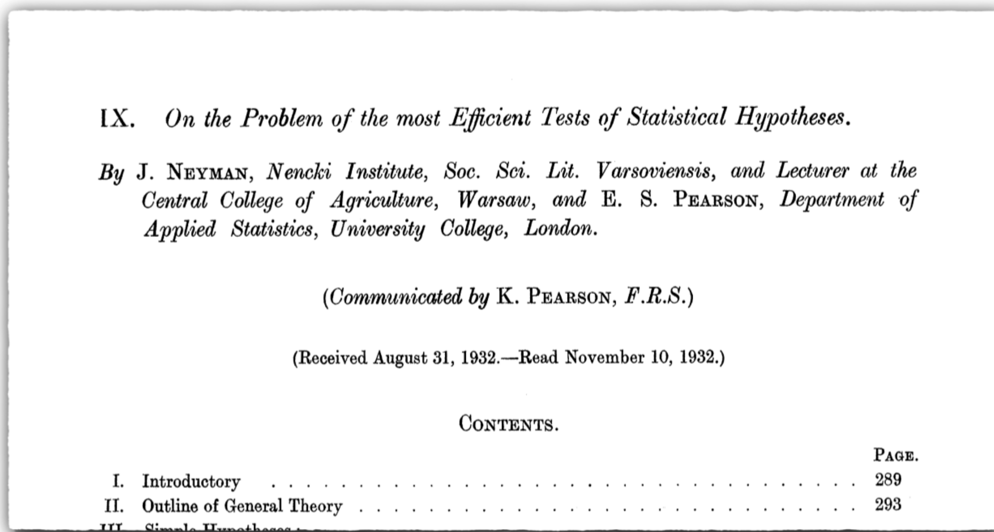
The physicist's way

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.





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

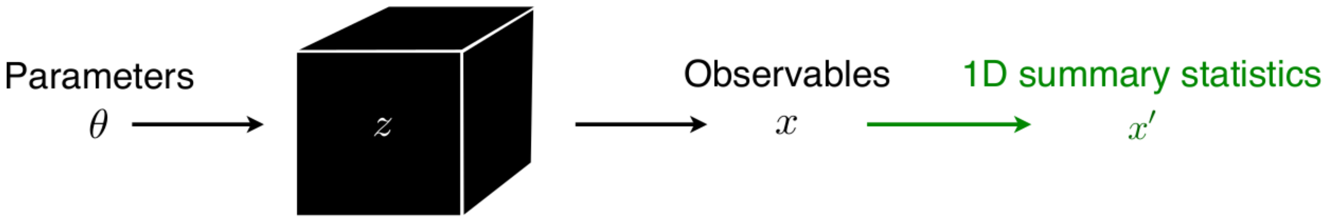
By J. NEYMAN, *Nencki Institute, Soc. Sci. Lit. Varsoviensis, and Lecturer at the Central College of Agriculture, Warsaw, and* E. S. PEARSON, *Department of Applied Statistics, University College, London.*

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

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

CONTENTS.

		PAGE.
I.	Introductory	289
II.	Outline of General Theory	293



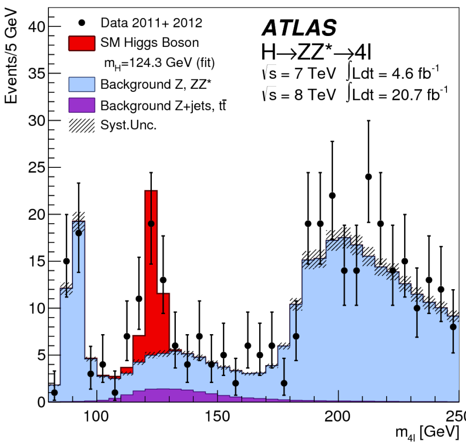
Define a projection function $s : \mathcal{X} \to \mathbb{R}$ mapping observables x to a summary statistics $x' = s(x)$.

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

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

From this it comes

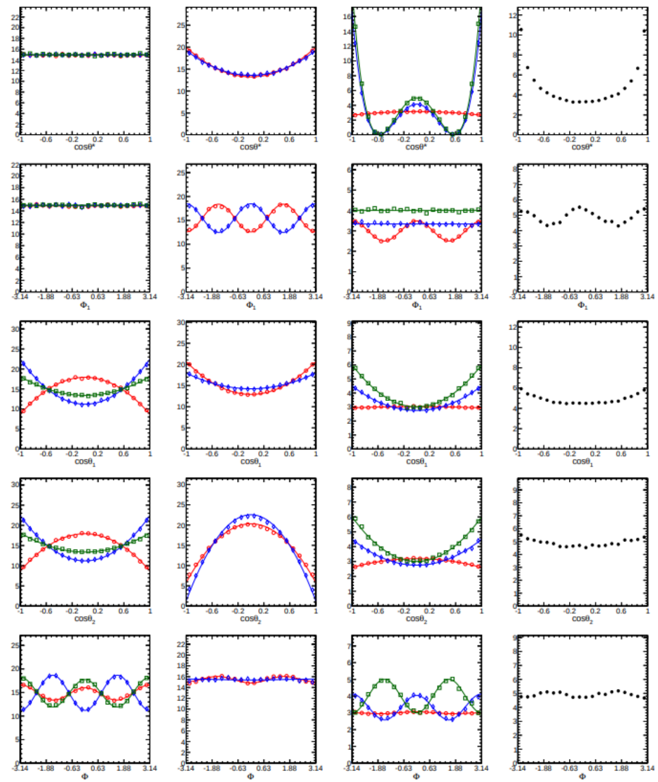
$$\frac{p(x|\theta_0)}{p(x|\theta_1)} \approx \frac{\hat{p}(x|\theta_0)}{\hat{p}(x|\theta_1)} = \hat{r}(x|\theta_0, \theta_1).$$



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.

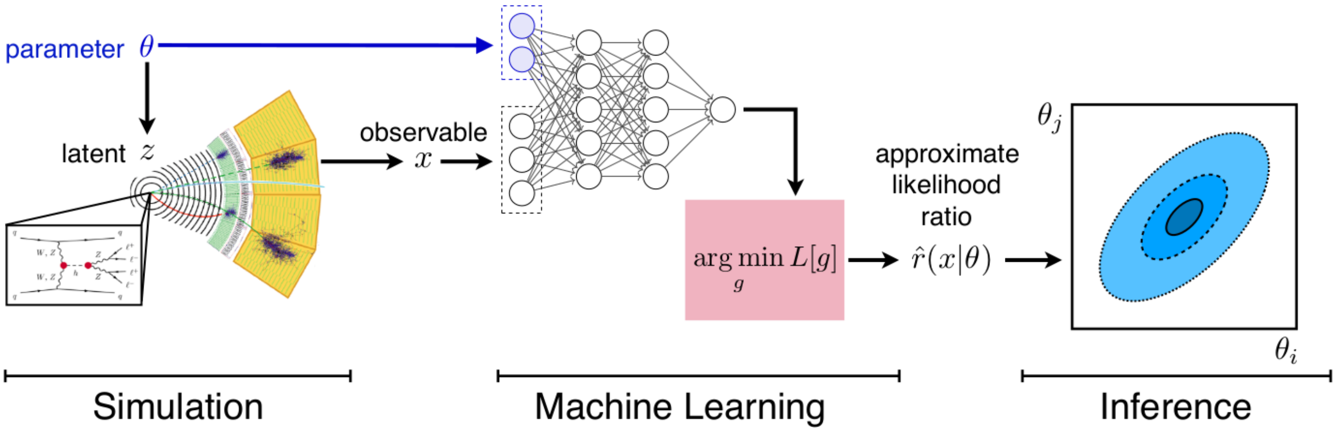
Unfortunately, filling high-dimensional histograms is not tractable.



Who you gonna call? Machine learning!

Refs: Bolognesi et al, 2012 (arXiv:1208.4018)

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,

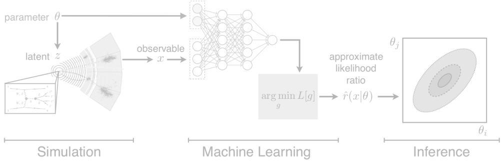
$$r(x|\theta_0, \theta_1) \approx \hat{r}(x|\theta_0, \theta_1) = \frac{1 - \hat{s}(x)}{\hat{s}(x)}$$

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

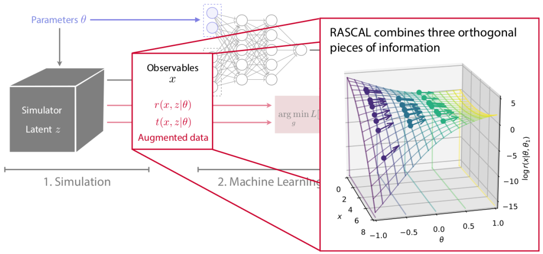
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

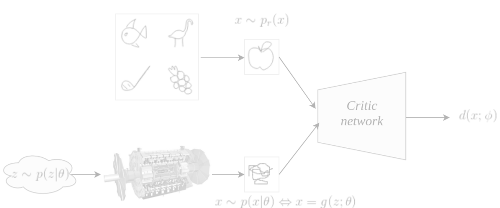


Histograms of observables
Neural density (ratio) estimation

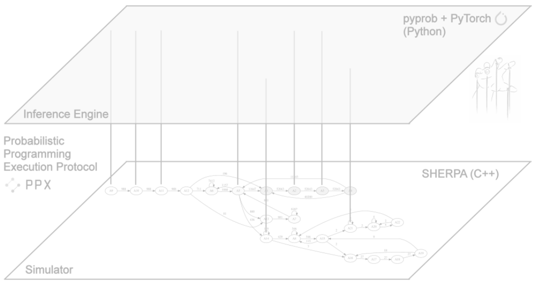


Mining gold from implicit models

Learn to control the simulator

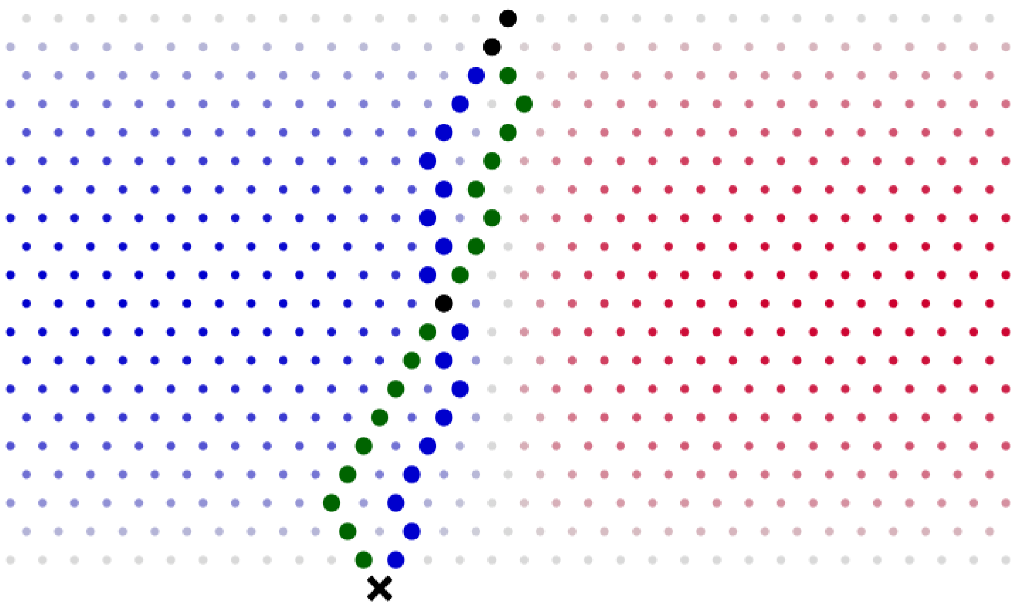


Adversarial variational optimization



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

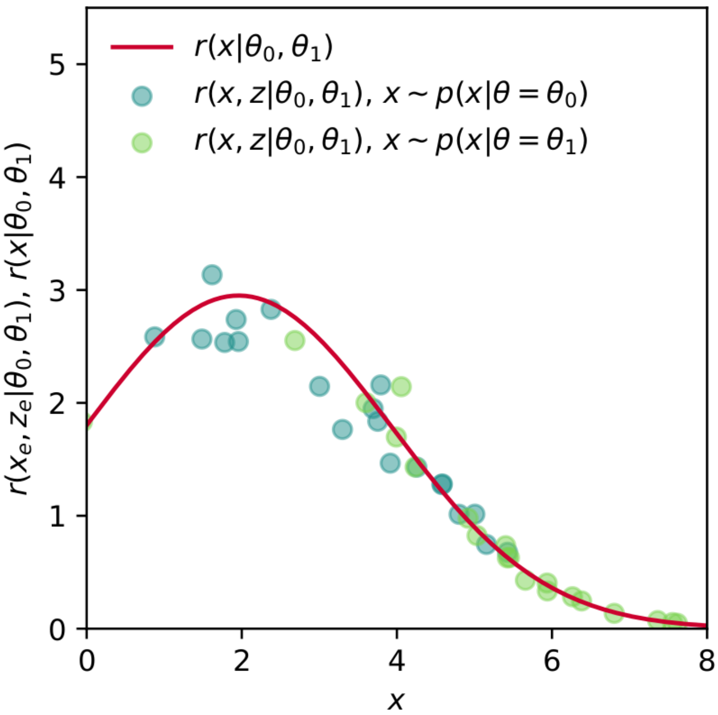
We call this process mining gold from your simulator!

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 $r(x|\theta_0, \theta_1)$?

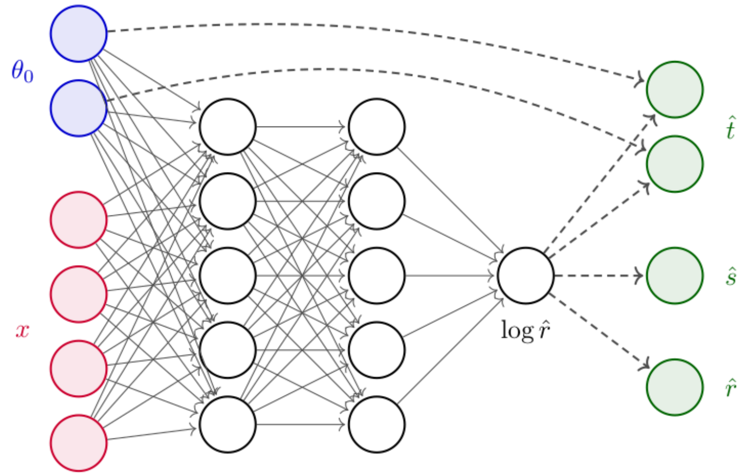


Let us define

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

Via calculus of variations, we find that this functional 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^*?

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

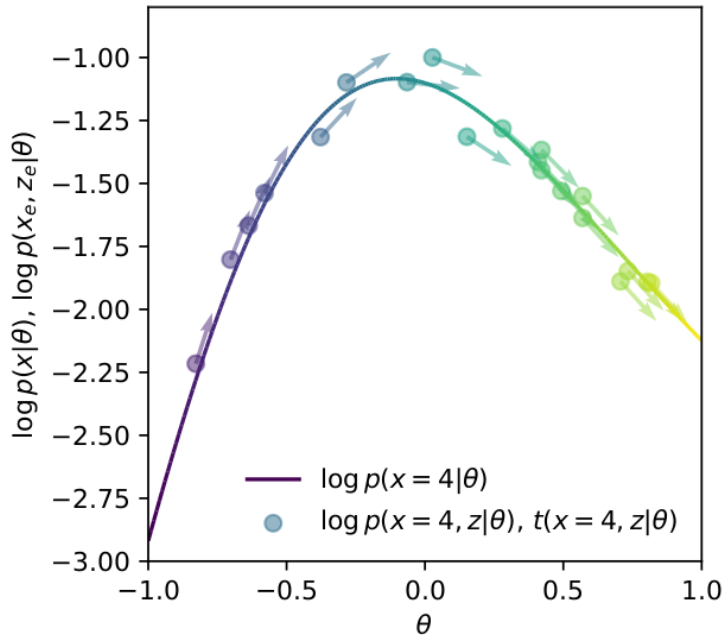
Minimizing functionals is exactly what machine learning does. In our case,

- $\hat{r}$ 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.



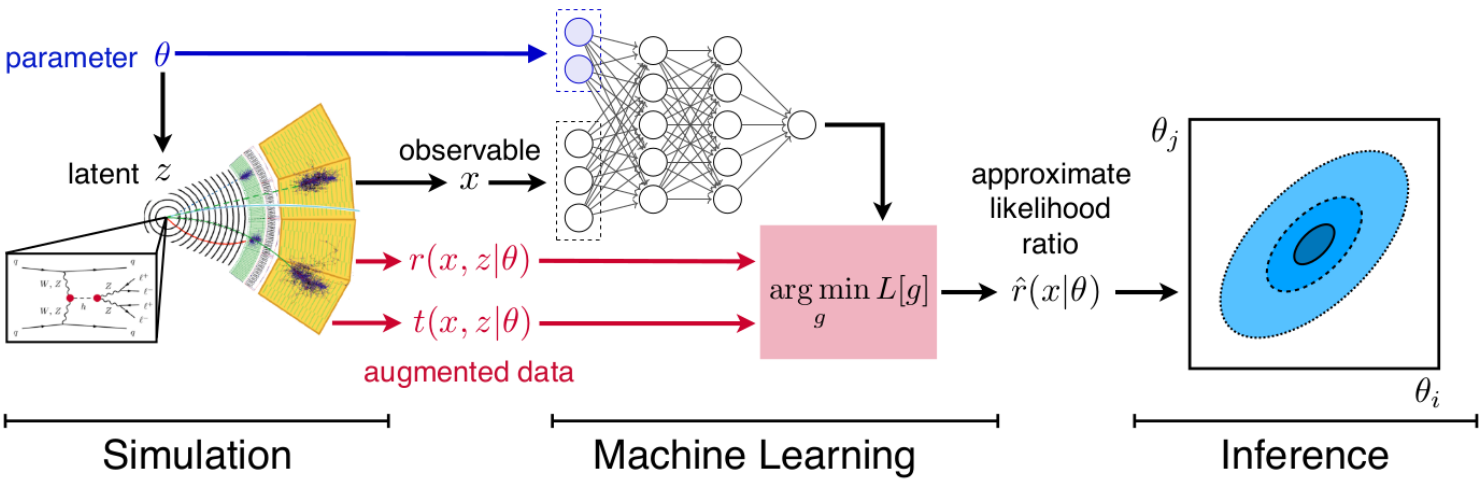
We define

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

which can be shown to be minimized by $t^*(x) = t(x|\theta_0)$.

Rascal

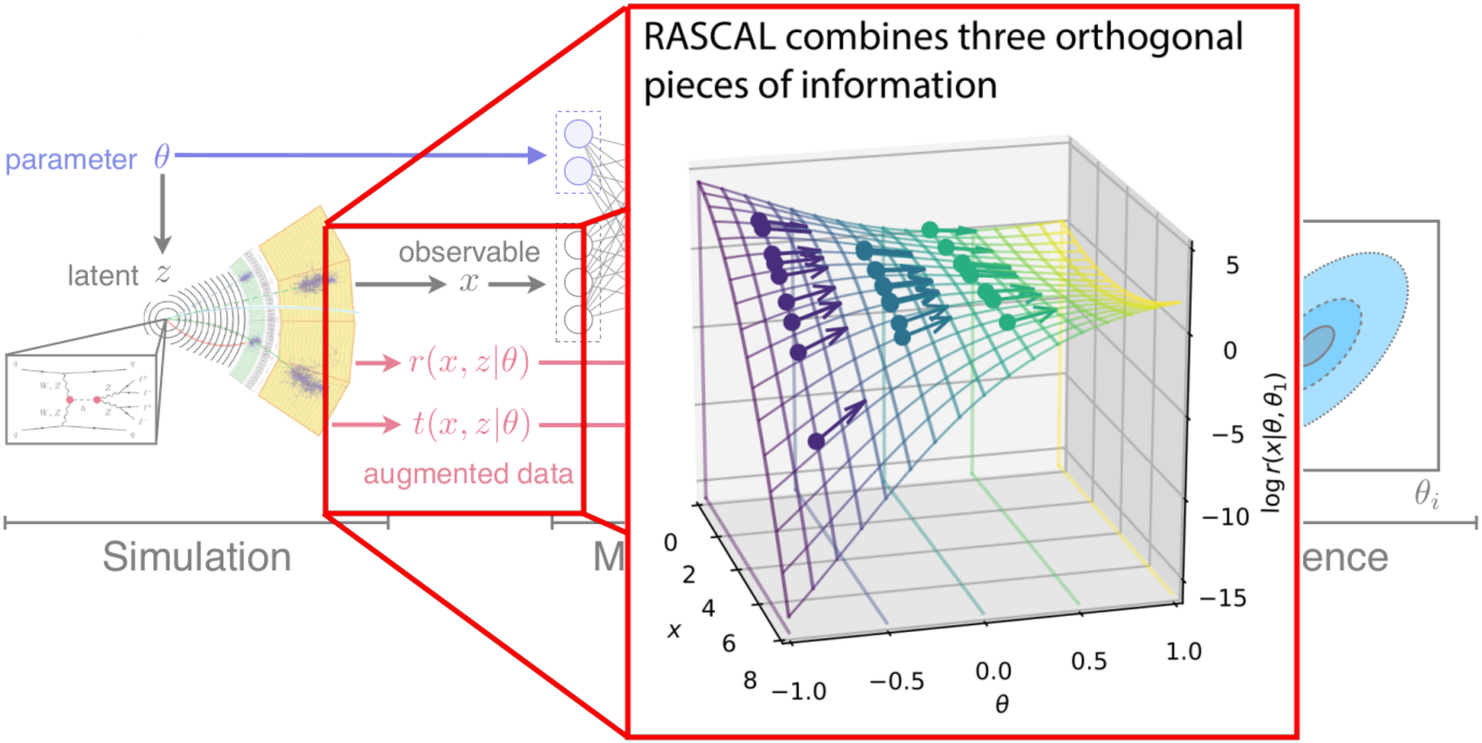
$$L_{RASCAL} = L_r + L_t$$



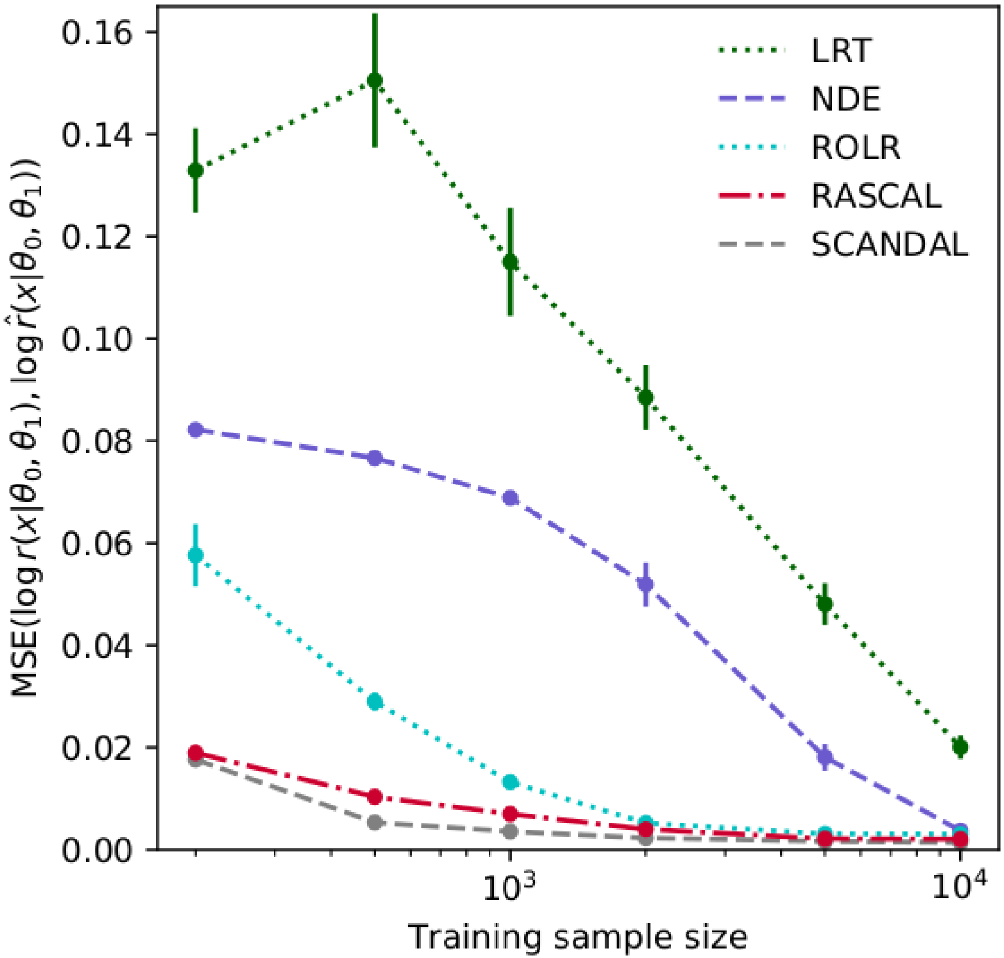
Refs: Brehmer et al, 2018 (arXiv:1805.12244)

RASCAL

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



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

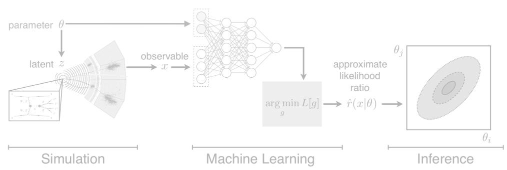


LRT
NDE
ROLR
RASCAL
SCANDAL
MSE(log r(x|θ0, θ1), log r̂(x|θ0, θ1))
Training sample size

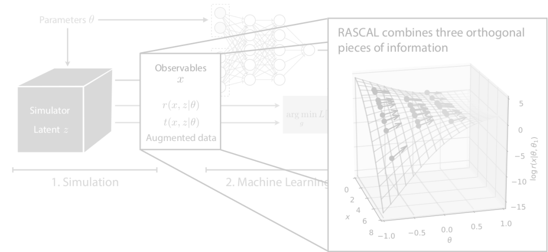
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

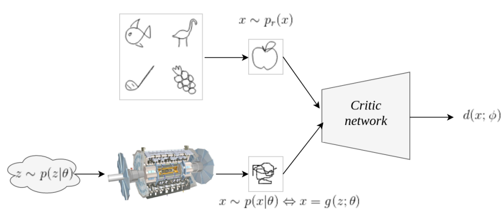


Histograms of observables
Neural density (ratio) estimation

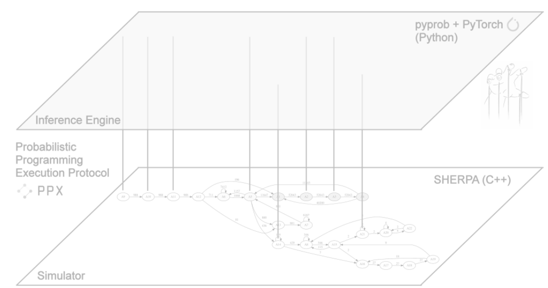


Mining gold from implicit models

Learn to control the simulator

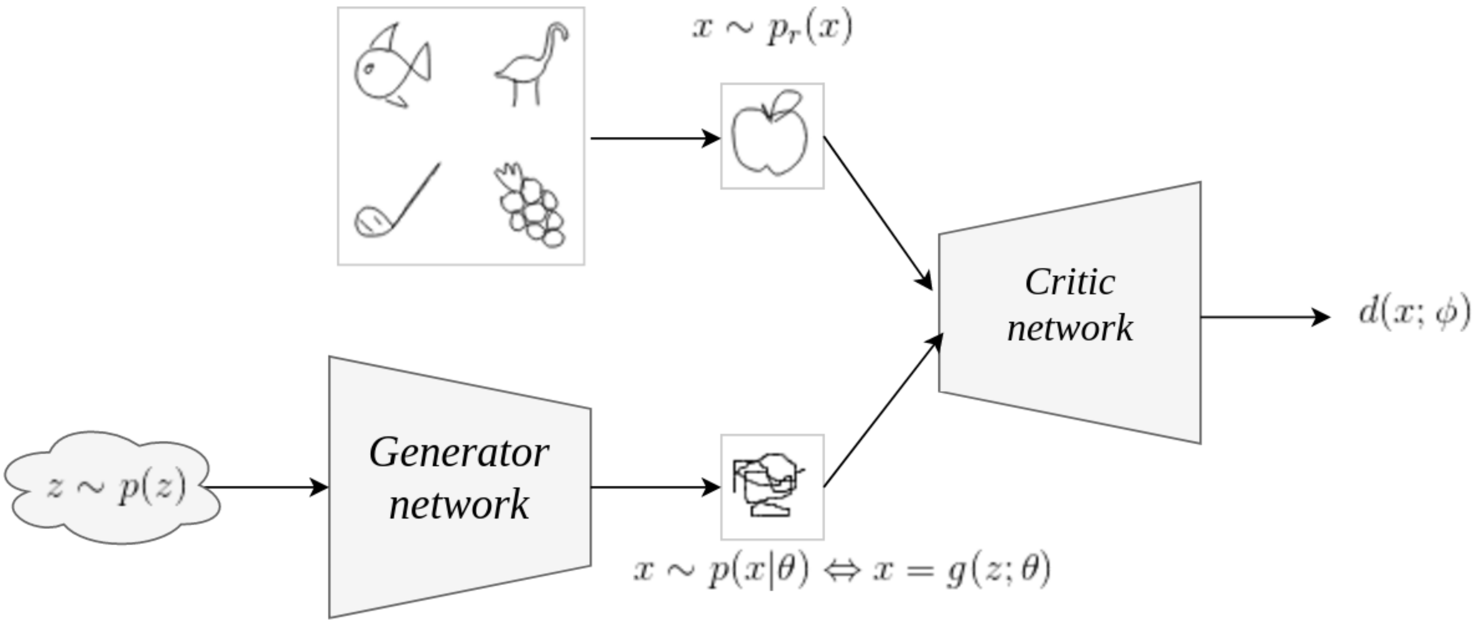


Adversarial variational optimization



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

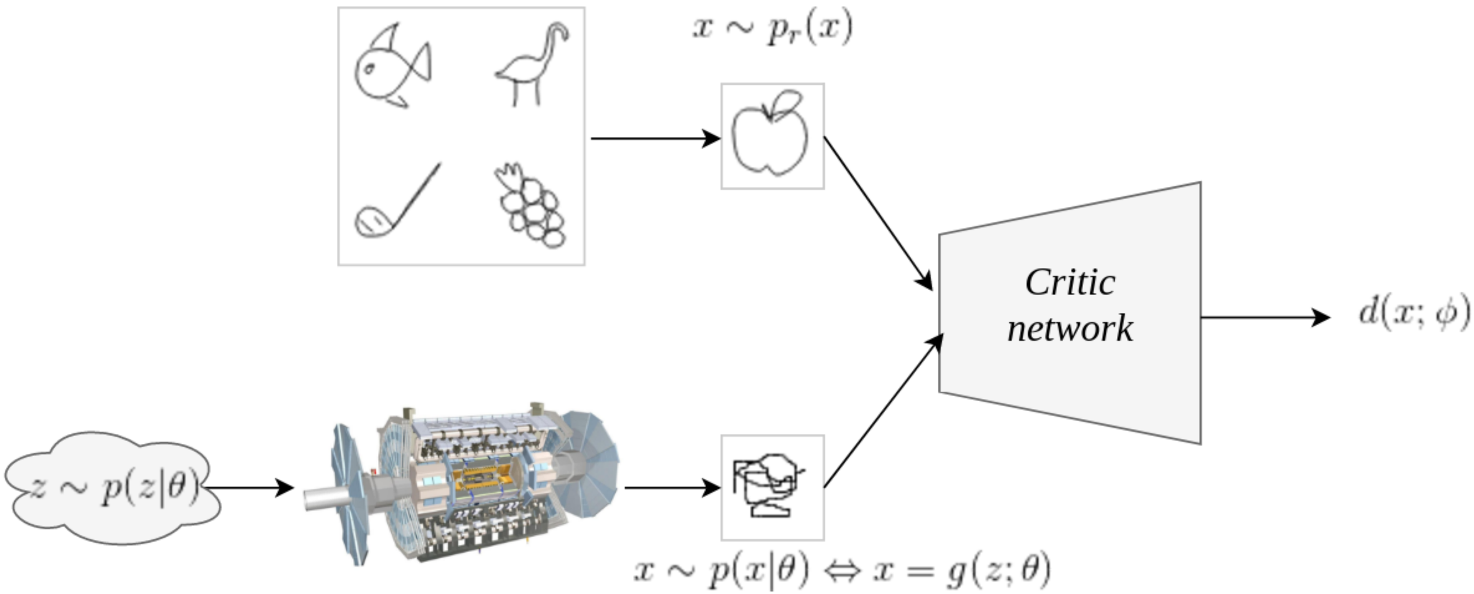
Zhang et al 2018

Brock et al 2018

Figure 2. Uncurated set of images produced by our style-based generator (config F) with the FFHQ dataset. Here we used a variation of the truncation trick [5, 29] with $\psi = 0.7$ for resolutions $4^2 - 32^2$. Please see the accompanying video for more results.

Karras et al, 2018.

AVO



Replace g with an actual scientific simulator!

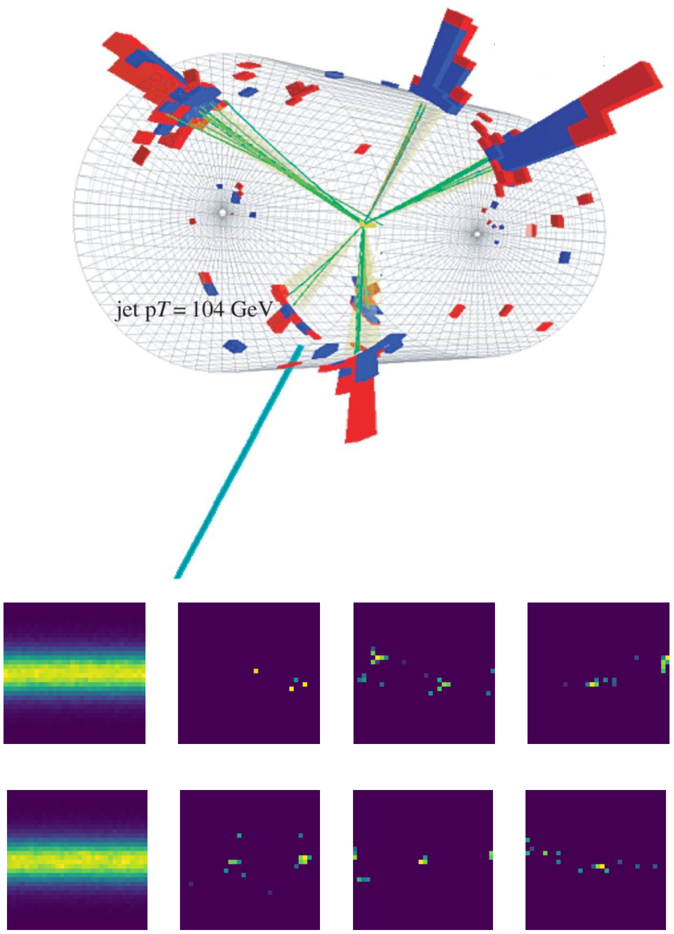
Refs: Louppe et al, 2017 (arXiv:1707.07113)

Key insights

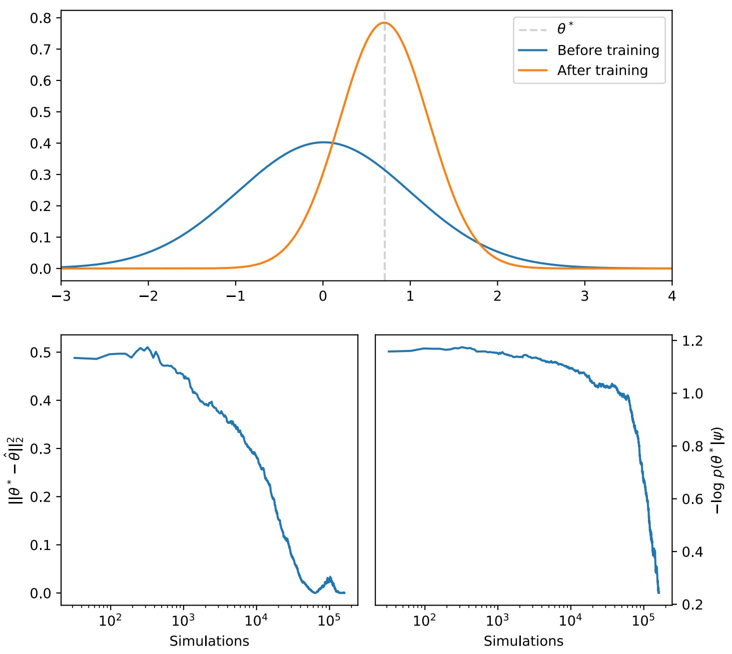
- Replace the generative network with a non-differentiable forward simulator $g(\mathbf{z}; \theta)$.
- Let the neural network critic figure out how to adjust the simulator parameters.
- Combine with variational optimization to bypass the non-differentiability by optimizing upper bounds of the adversarial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

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

respectively over ϕ and ψ.



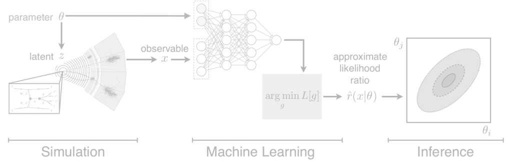
Samples for $\theta = 0$ (top) vs.
samples for $\theta = 0.81$ (bottom).



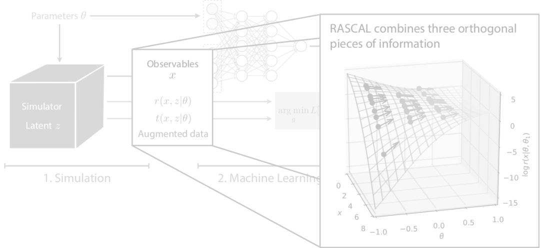
Treat the simulator as a black box

Make use of the inner structure

Learn a proxy for inference

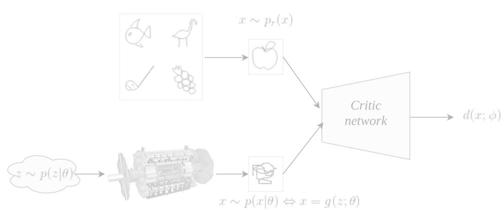


Histograms of observables
Neural density (ratio) estimation

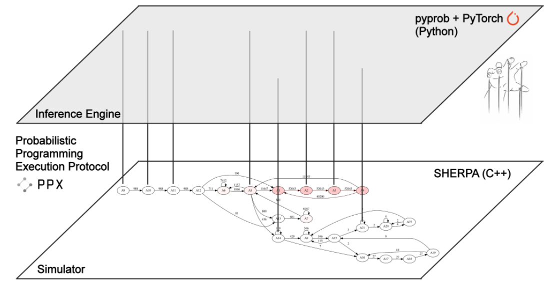


Mining gold from implicit models

Learn to control the simulator

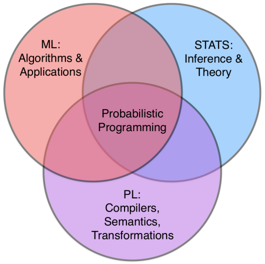


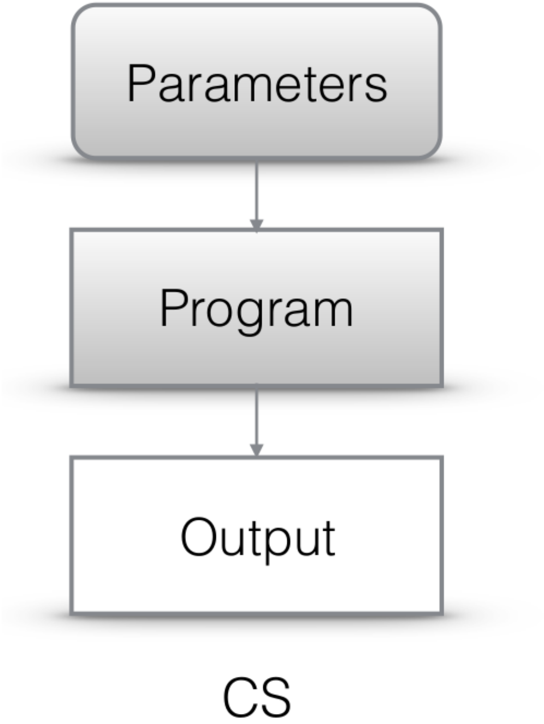
Adversarial variational optimization



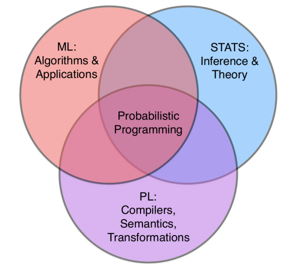
Probabilistic programming

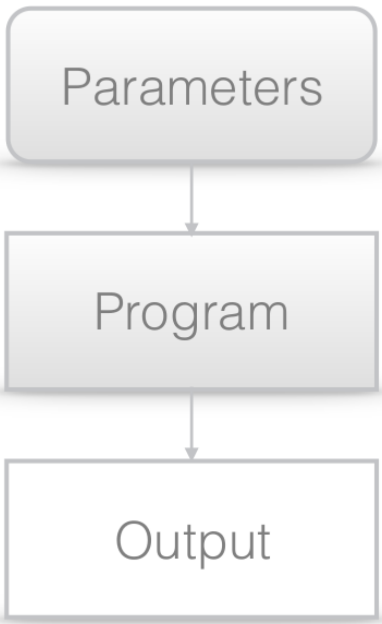
Probabilistic programming



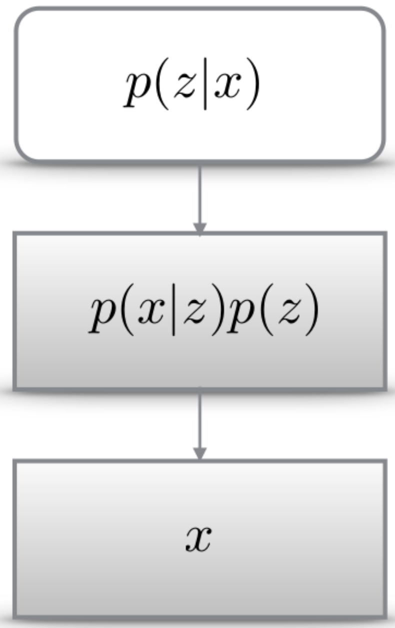


Probabilistic programming



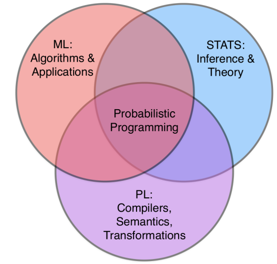


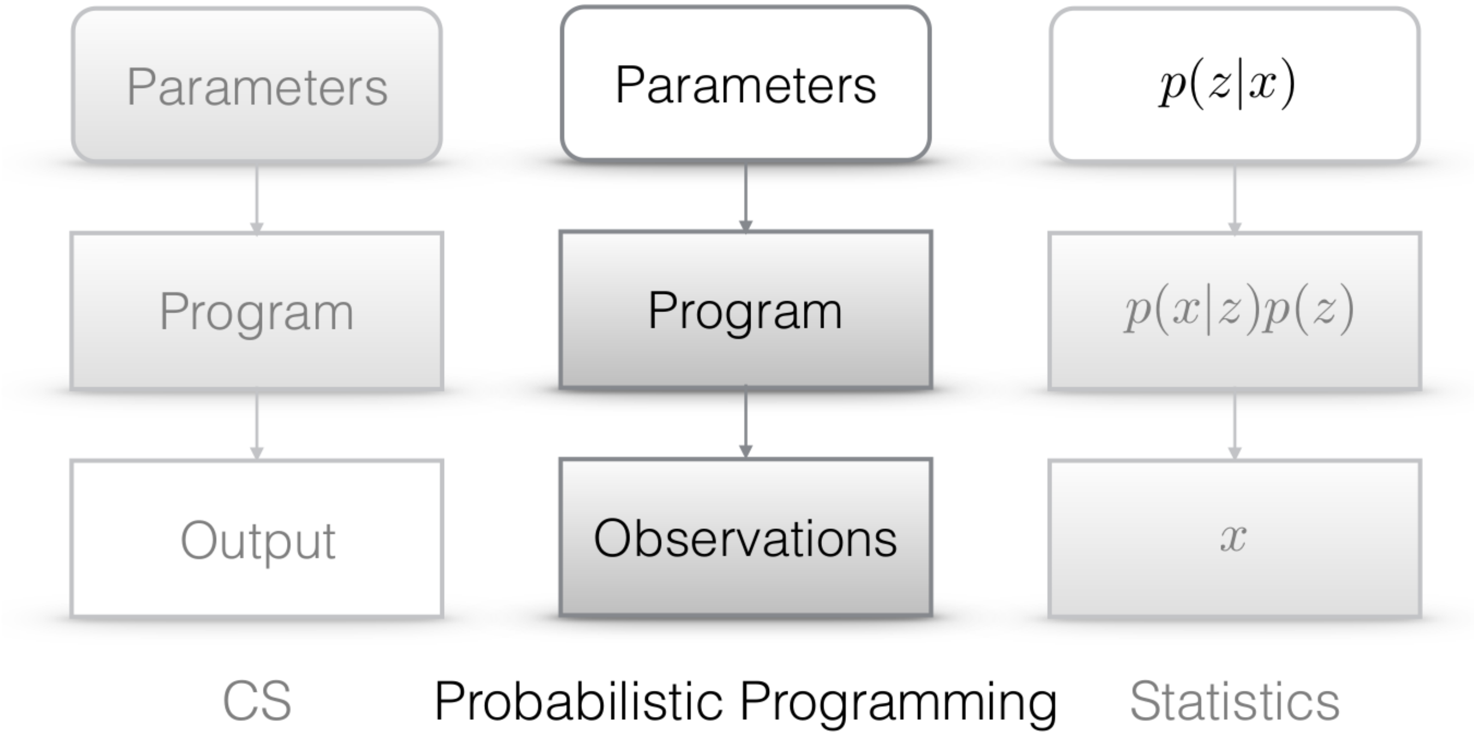
CS



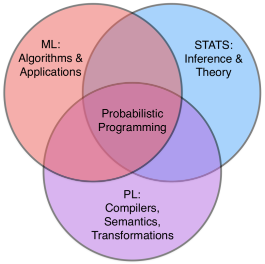
Statistics

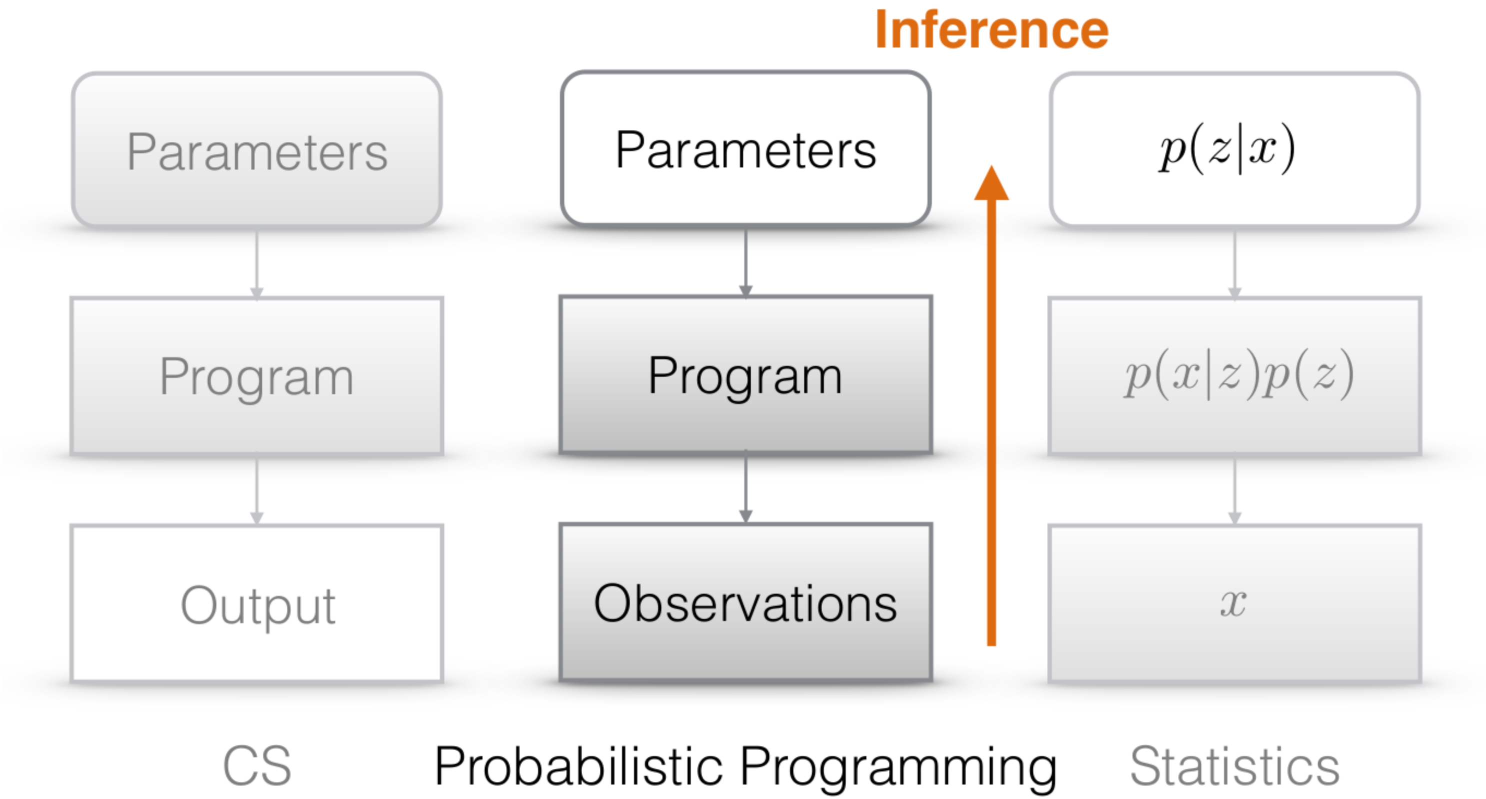
Probabilistic programming

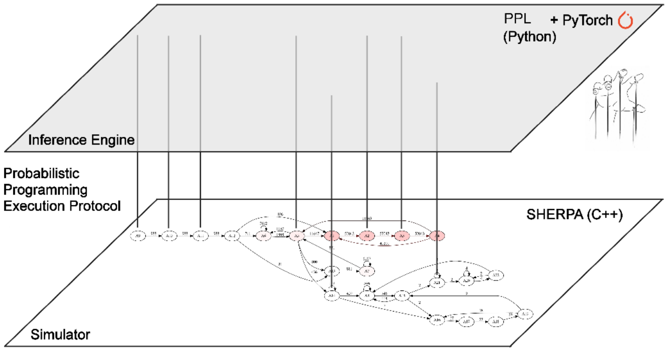


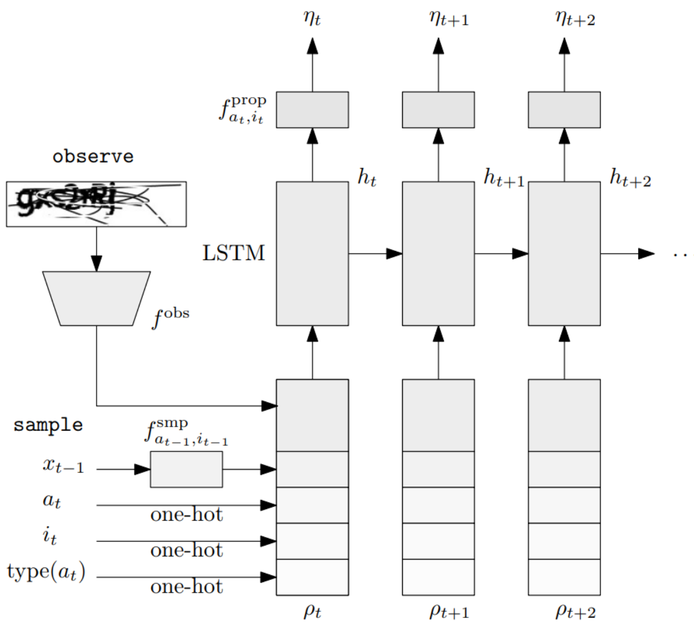


Probabilistic programming









Key insights

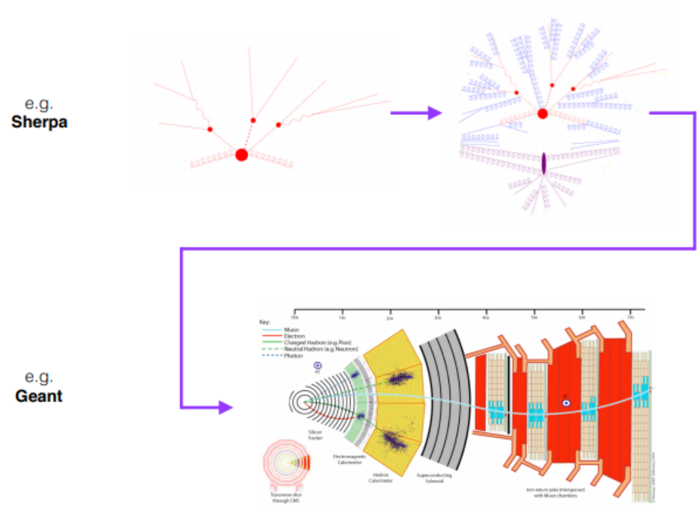
Let a neural network take full control of the internals of the simulation program by hijacking all calls to the random number generator.

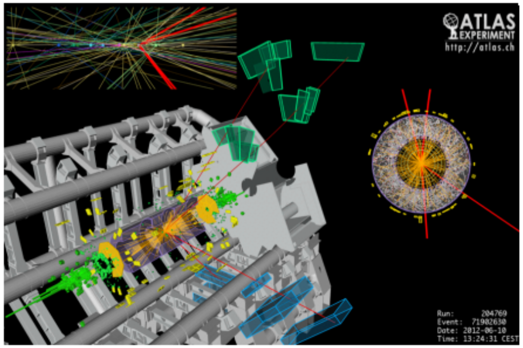
Refs: Le et al, 2016 (arXiv:1610.09900); Baydin et al, 2018 (arXiv:1807.07706)

```
(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

Refs: Le et al, 2016 (arXiv:1610.09900)

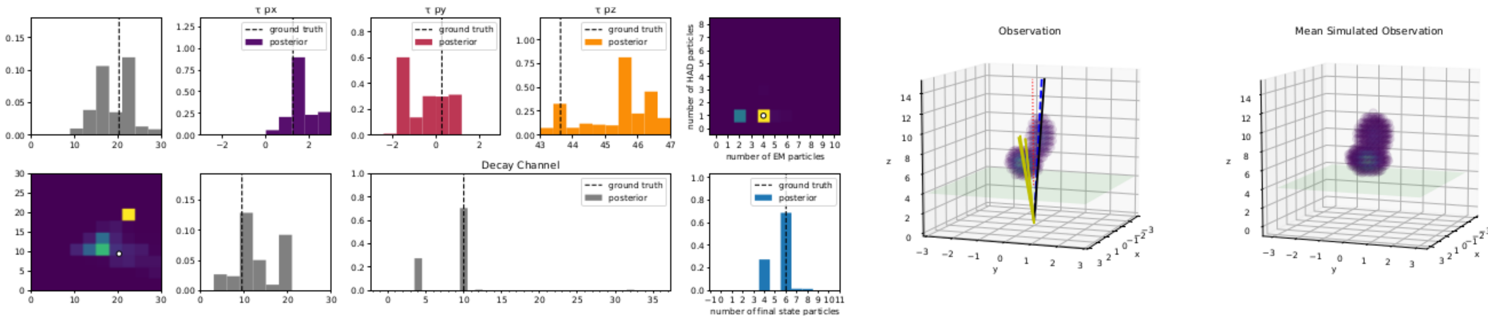


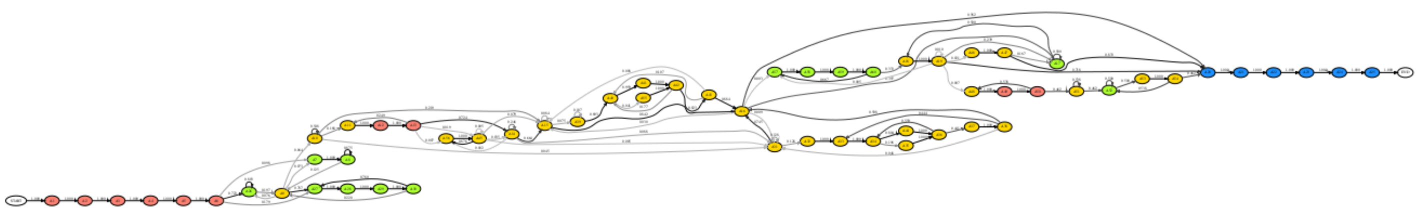


x	y
event & detector simulators	ATLAS detector output

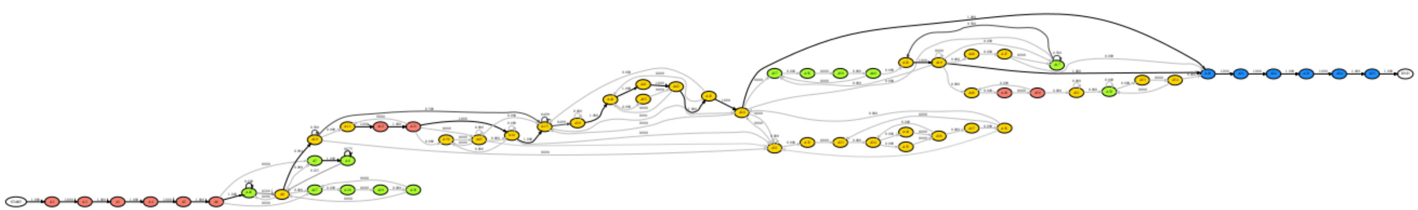
Probabilistic programming hooked to particle physics simulators
(work in progress)

Refs: Baydin et al, 2018 (arXiv:1807.07706)





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



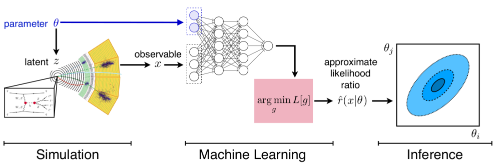
(b) Posterior execution $p(\mathbf{x}|\mathbf{y})$ conditioned on a given calorimeter observation $\mathbf{y}$.

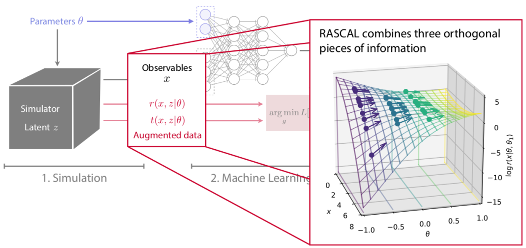
Refs: Baydin et al, 2018 (arXiv:1807.07706)

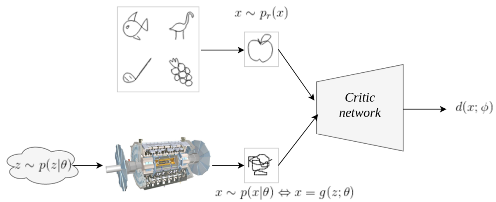
Summary

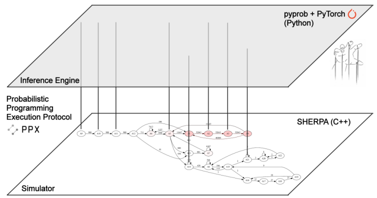
Summary

- 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., 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.