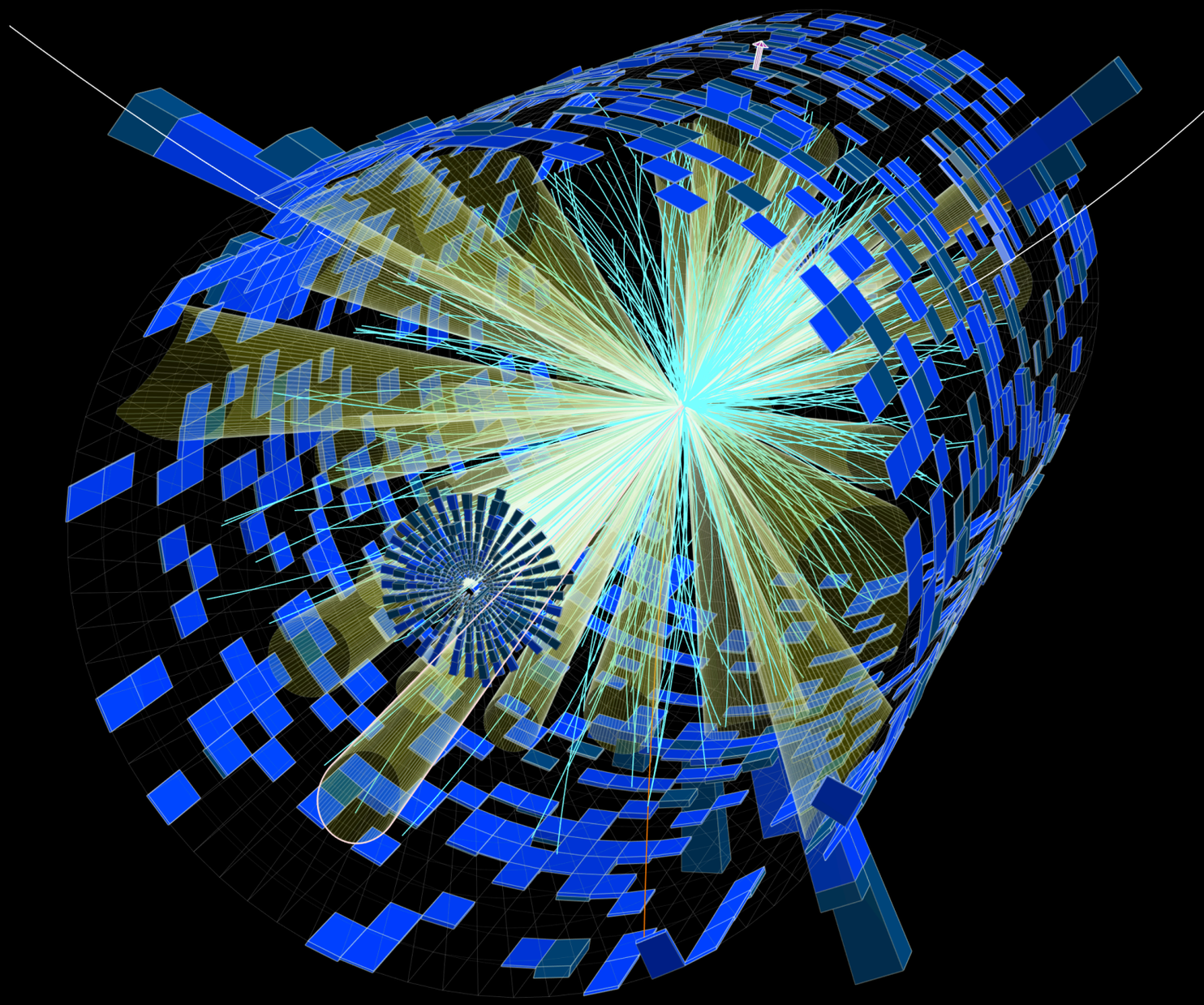
NYU Center for Data Science

Center for Cosmology and particle physics

LIKELIHOOD-FREE INFERENCE

OVERVIEW & COMMENTS

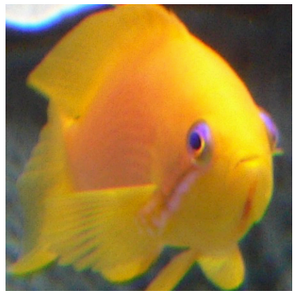


@KyleCranmer
New York University
Department of Physics
Center for Data Science
CILVR Lab

SUPPORT

The SCAILFIN Project

scailfin.github.io



COLLABORATORS

Gilles Louppe
U. Liège

Kyunghyun Cho

Joan Bruna

Brenden Lake

Meghan Frate

Juan Pavez

Tilman Plehn

Johann Brehmer

Felix Kling

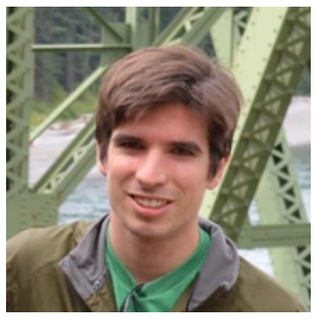
Lukas Heinrich

Markus Store

Tim Head

Michael Kagan

Irina Espejo



Peter Sadowski

Daniel Whiteson

Pierre Baldi

Lezcano Casado

Atılım Güneş Baydin
University of Oxford

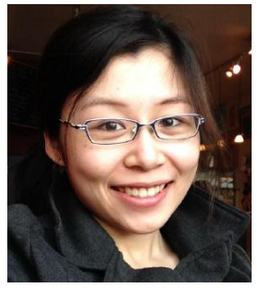
Prabhat
NERSC, Berkeley Lab

Wahid Bhimji
NERSC, Berkeley Lab

Frank Wood
University of Oxford

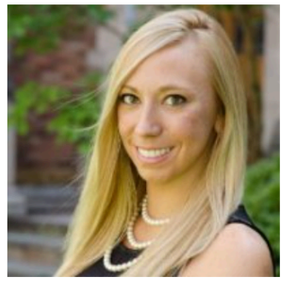
Phiala Shanahan

William Detmold



Karen Ng

Tuan Anh Le



Michela Paganini
Yale University

Daniela Huppenkothen
New York University

Savannah Thais
Yale University

Ruth Angus
Columbia University

REFERENCES







Johann Brehmer Gilles Louppe Juan Pavez Markus Stoye Felix Kling Tilman Plehn Tim Tait

KC, GL, JP: Approximating Likelihood Ratios with Calibrated Discriminative Classifiers [1506.02169]

JB, KC, FK, TP: Better Higgs Measurements Through Information Geometry [1612.05261]

JB, FK, TP, TT: Better Higgs-CP Measurements Through Information Geometry [1712.02350]

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

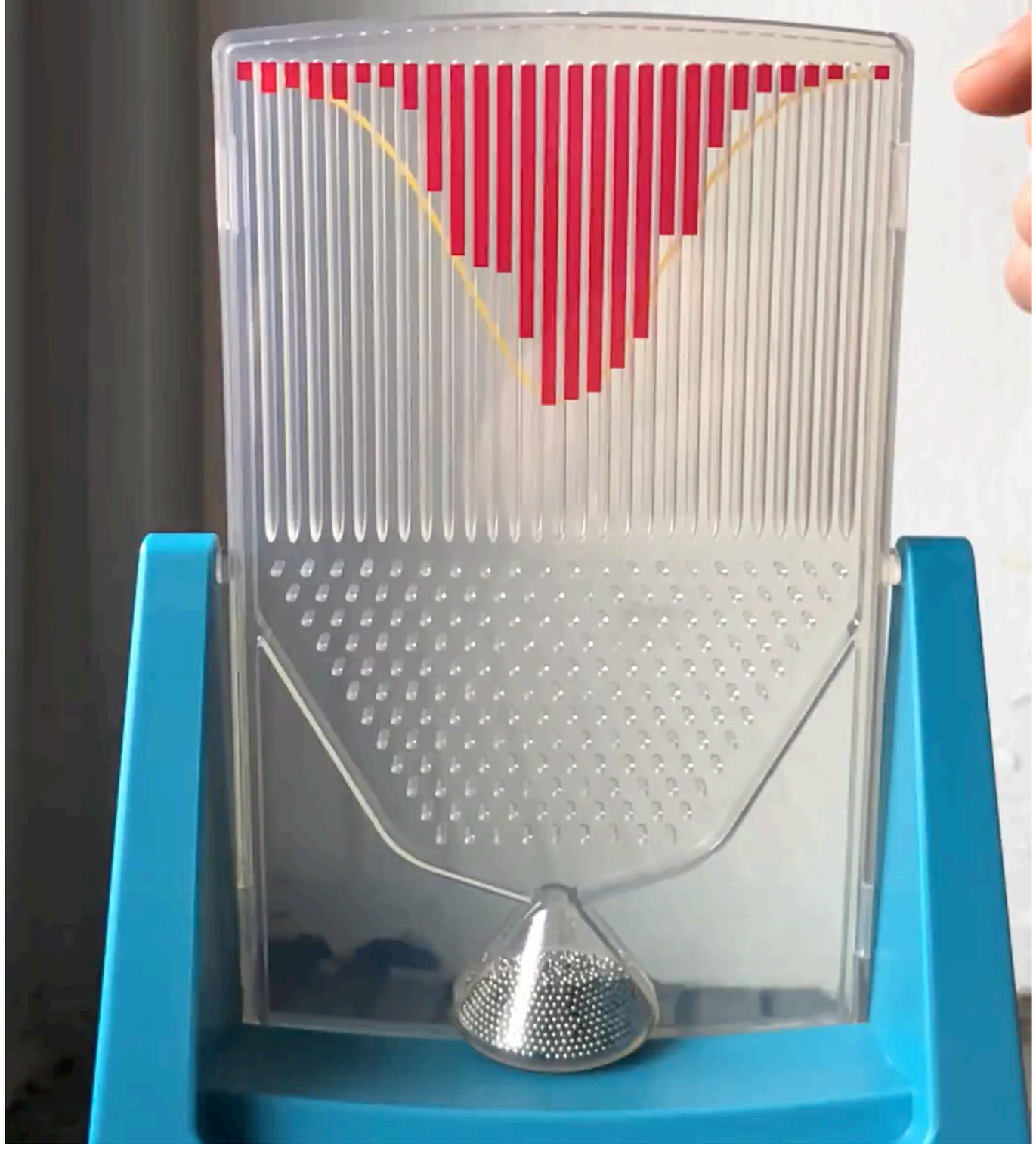
JB, KC, GL, JP: A Guide to Constraining Effective Field Theories with Machine Learning [1805.00020]

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

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

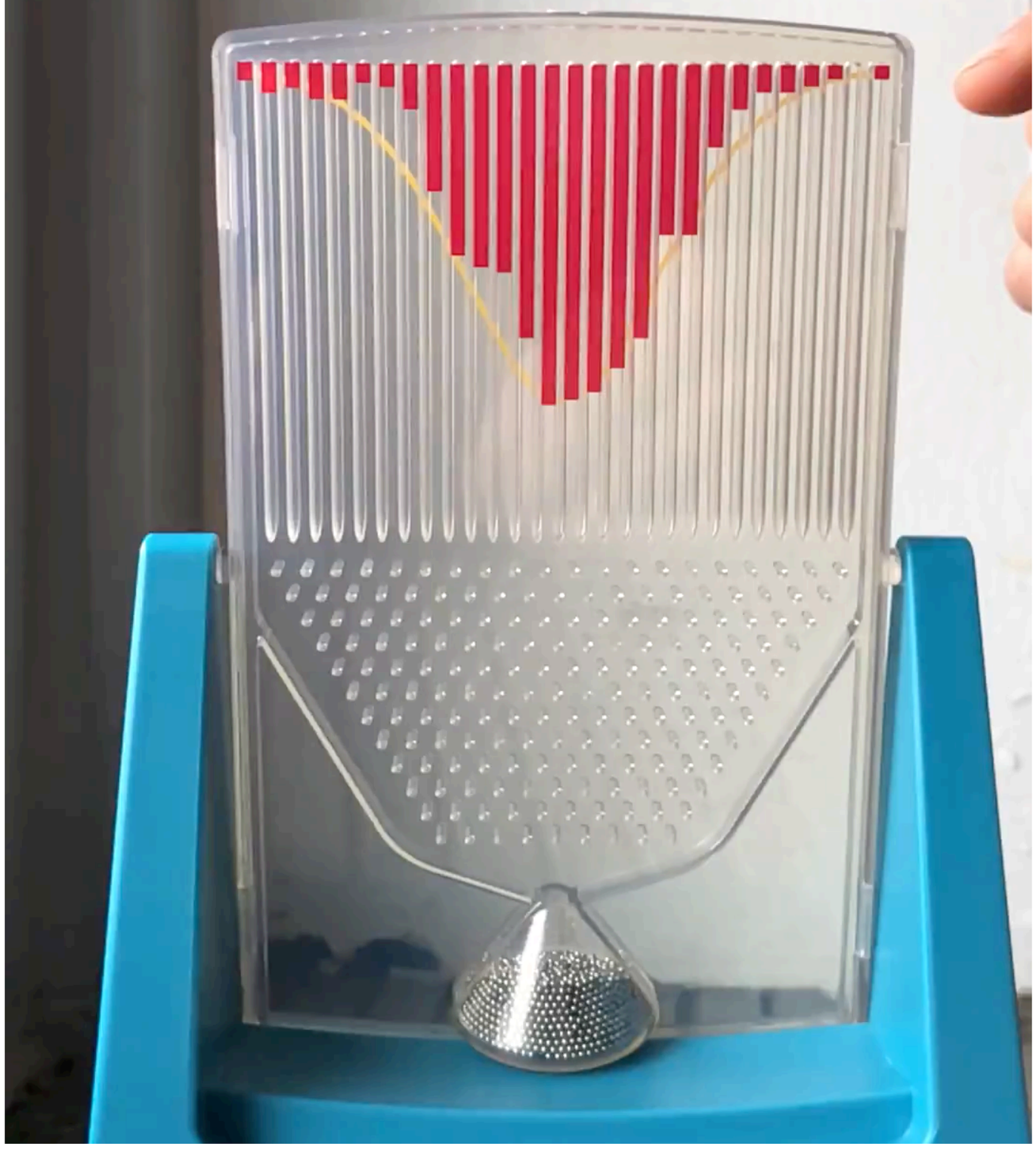
JB, KC, FK: MadMiner In preparation

Thanks to **Johann** and Gilles for help with slides!



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

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



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

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

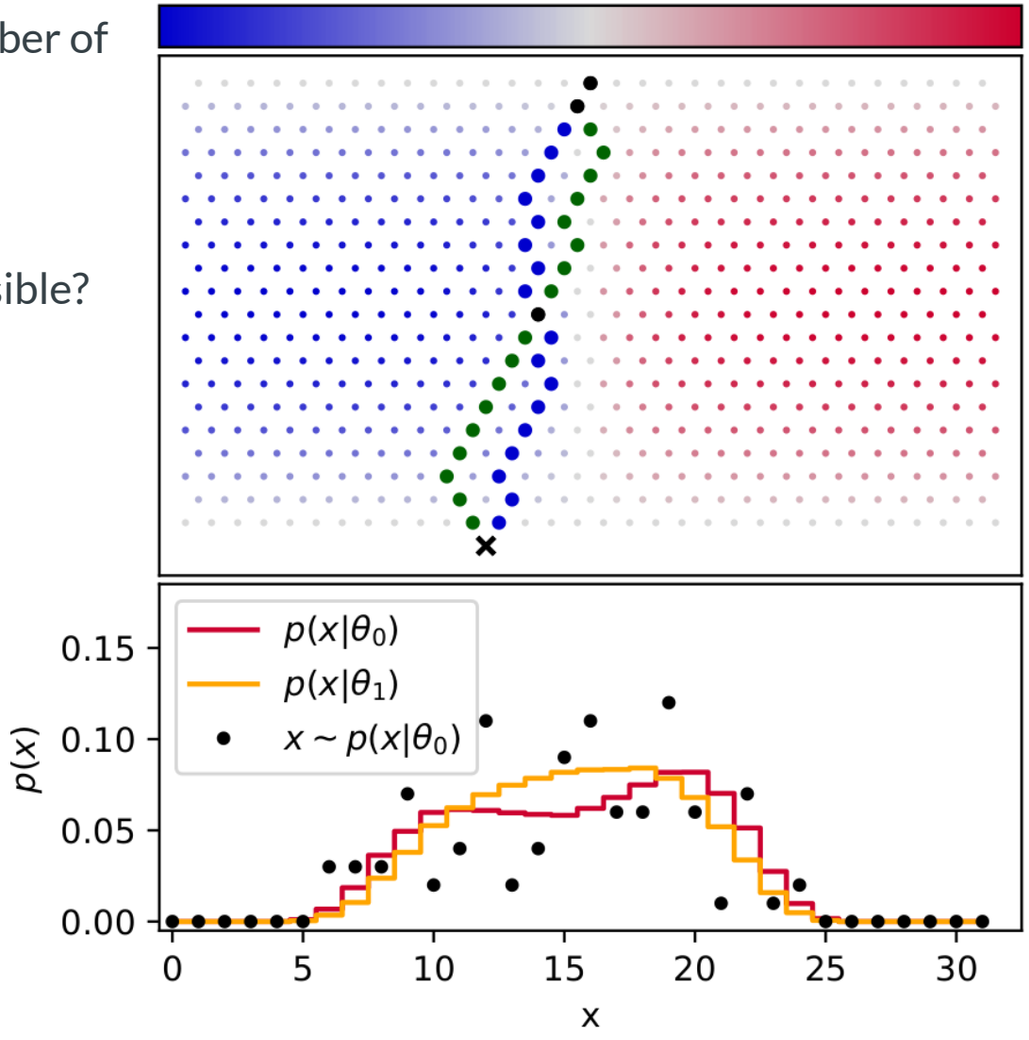
WHAT IF WE SHIFT THE PINS

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 no longer possible?



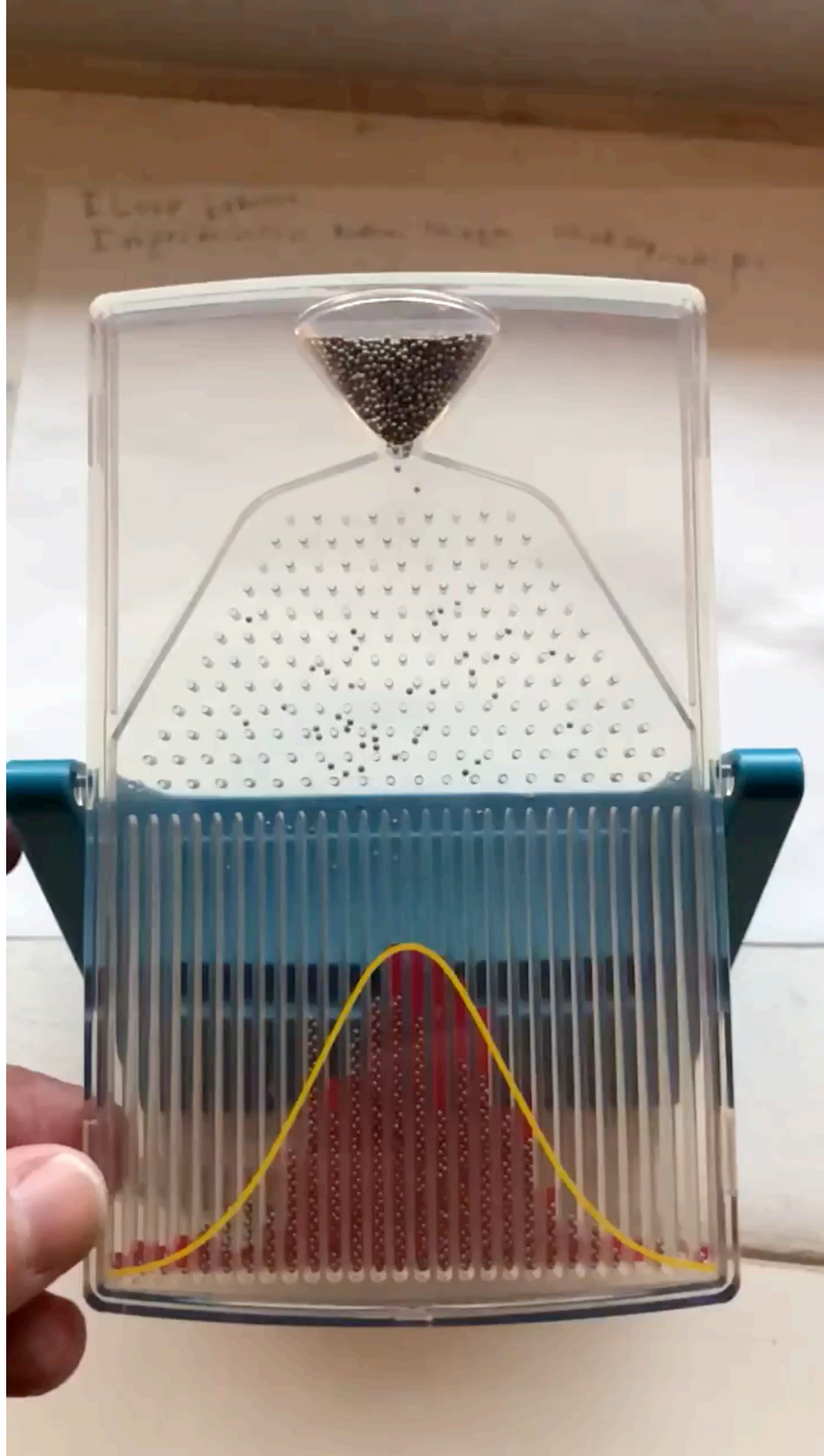
UH OH!

The actual situation is much more complicated.

It's not a Binomial distribution!

What is it?

I have no idea, but I could simulate it!



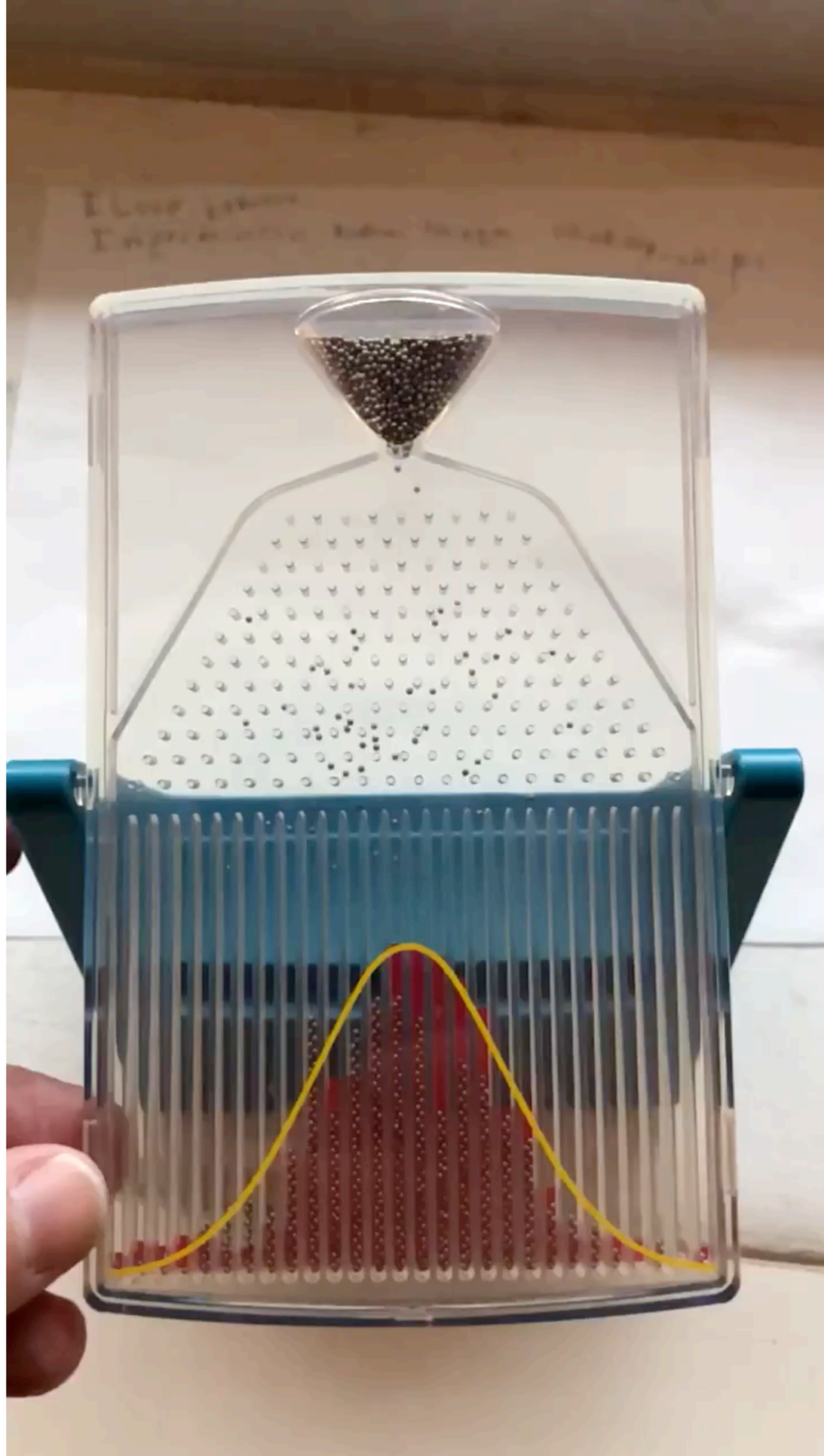
UH OH!

The actual situation is much more complicated.

It's not a Binomial distribution!

What is it?

I have no idea, but I could simulate it!



STATISTICAL FRAMING

forward modeling
generation
simulation

PREDICTION

θ
parameters of interest

p(x, z | θ, ν)

x
observed data
simulated data

z
latent variables
Monte Carlo truth

ν
nuisance parameters

INFERENCE

inverse problem
measurement
parameter estimation

A COMMON THEME, A COMMON LANGUAGE

ABC

resources on approximate Bayesian computational methods

Search

Home

Home

This website keeps track of developments in approximate Bayesian computation (ABC) (a.k.a. likelihood-free), a class of computational statistical methods for Bayesian inference under intractable likelihoods. The site is meant to be a resource both for biologists and statisticians who want to learn more about ABC and related methods. Recent publications are under Publications 2012. A comprehensive list of publications can be found under Literature. If you are unfamiliar with ABC methods see the Introduction. Navigate using the menu to learn more.

ABC in Montreal | ABC in Montreal (2014)

ABC in Montreal

Approximate Bayesian computation (ABC) or likelihood-free (LF) methods have developed mostly beyond the radar of the machine learning community, but are important tools for a large and diverse segment of the scientific community. This is particularly true for systems and population biology, computational neuroscience, computer vision, healthcare sciences, but also many others.

Interaction between the ABC and machine learning community has recently started and contributed to important advances. In general, however, there is still significant room for more intense interaction and collaboration. Our workshop aims at being a place for this to happen.

Markov chain Monte Carlo without likelihoods

Paul Marjoram*, John Molitor*, Vincent Plagnol†, and Simon Tavaré†‡

*Biostatistics Division, Department of Preventive Medicine, Keck School of Medicine, and †Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089

Communicated by Michael S. Waterman, University of Southern California, Los Angeles, CA, October 24, 2003 (received for review June 20, 2003)

Many stochastic simulation approaches for generating observations from a posterior distribution depend on knowing a likelihood function. However, for many complex probability models, such likelihoods are either impossible or computationally prohibitive to obtain. Here we present a Markov chain Monte Carlo method for generating observations from a posterior distribution without the use of likelihoods. It can also be used in frequentist applications, in particular for maximum-likelihood estimation. The approach is illustrated by an example of ancestral inference in population genetics. A number of open problems are highlighted in the discussion.

One of the basic problems in Bayesian statistics is the computation of posterior distributions. We imagine data $\mathcal{D}$ generated from a model $\mathcal{M}$ determined by parameters θ, the prior density of which is denoted by $\pi(\theta)$. We assume unless otherwise stated that the data are discrete. The posterior distribution of interest is $f(\theta|\mathcal{D})$, which is given by

$$f(\theta|\mathcal{D}) = \mathbb{P}(\mathcal{D}|\theta)\pi(\theta)/\mathbb{P}(\mathcal{D}), \qquad [1]$$

where $\mathbb{P}(\mathcal{D}) = \int \mathbb{P}(\mathcal{D}|\theta)\pi(\theta)d\theta$ is the normalizing constant.

In most scientific contexts, explicit formulae for such posterior densities are few and far between, and we usually resort to stochastic simulation to generate observations from f. Perhaps the simplest approach for this is the rejection method:

A1. Generate θ from $\pi(\cdot)$.
A2. Accept θ with probability $h = \mathbb{P}(\mathcal{D}|\theta)$; return to *A1*.

of ε therefore reflects a tension between computability and accuracy. The method is still honest in that, for a given ρ and ε, we are generating independent and identically distributed observations from $f(\theta|\rho(\mathcal{D}, \mathcal{D}') \leq \varepsilon)$.

When $\mathcal{D}$ is high-dimensional or continuous, this approach can be impractical as well, and then the comparison of $\mathcal{D}'$ with $\mathcal{D}$ can be made by using lower-dimensional summaries of the data. The motivation for this approach is that if the set of statistics $S = (S_1, \ldots, S_p)$ is sufficient for θ, in that $\mathbb{P}(\mathcal{D}|S, \theta)$ is independent of θ, then $f(\theta|\mathcal{D}) = f(\theta|S)$. The normalizing constant $\mathbb{P}(S)$ is typically larger than $\mathbb{P}(\mathcal{D})$, resulting in more acceptances. In practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach. Thus we seek to use knowledge of the particular problem at hand to suggest summary statistics that capture information about θ. With these statistics in hand, we have the following approximate Bayesian computation scheme for data $\mathcal{D}$ summarized by S:

D1. Generate θ from $\pi(\cdot)$.
D2. Simulate $\mathcal{D}'$ from stochastic model $\mathcal{M}$ with parameter θ, and compute the corresponding statistics S'.
D3. Calculate the distance $\rho(S, S')$ between S and S'.
D4. Accept θ if $\rho \leq \varepsilon$, and return to *D1*.

There are several advantages to these rejection methods, among them the fact that they are usually easy to code, they generate independent observations (and thus can use embarrassingly parallel computation), and they readily provide estimates of Baves factors that can be used for model com-

Markov chain Monte Carlo without likelihoods

Paul Marjoram*, John Molitor*, Vincent Plagnol†, and Simon Tavaré†‡

*Biostatistics Division, Department of Preventive Medicine, Keck School of Medicine, and †Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089

D1. Generate θ from $\pi(\cdot)$.
D2. Simulate $\mathcal{D}'$ from stochastic model $\mathcal{M}$ with parameter θ, and compute the corresponding statistics S'.
D3. Calculate the distance $\rho(S, S')$ between S and S'.
D4. Accept θ if $\rho \leq \varepsilon$, and return to *D1*.

One of the basic problems in Bayesian statistics is the computation of posterior distributions. We imagine data $\mathcal{D}$ generated from a model $\mathcal{M}$ determined by parameters θ, the prior density of which is denoted by $\pi(\theta)$. We assume unless otherwise stated that the data are discrete. The posterior distribution of interest is $f(\theta|\mathcal{D})$, which is given by

$$f(\theta|\mathcal{D}) = \mathbb{P}(\mathcal{D}|\theta)\pi(\theta)/\mathbb{P}(\mathcal{D}), \quad [1]$$

where $\mathbb{P}(\mathcal{D}) = \int \mathbb{P}(\mathcal{D}|\theta)\pi(\theta)d\theta$ is the normalizing constant.

In most scientific contexts, explicit formulae for such posterior densities are few and far between, and we usually resort to stochastic simulation to generate observations from f. Perhaps the simplest approach for this is the rejection method:

A1. Generate θ from $\pi(\cdot)$.
A2. Accept θ with probability $h = \mathbb{P}(\mathcal{D}|\theta)$; return to *A1*.

typically larger than $\mathbb{P}(\mathcal{D})$, resulting in more acceptances. In practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach. Thus we seek to use knowledge of the particular problem at hand to suggest summary statistics that capture information about θ. With these statistics in hand, we have the following approximate Bayesian computation scheme for data $\mathcal{D}$ summarized by S:

D1. Generate θ from $\pi(\cdot)$.
D2. Simulate $\mathcal{D}'$ from stochastic model $\mathcal{M}$ with parameter θ, and compute the corresponding statistics S'.
D3. Calculate the distance $\rho(S, S')$ between S and S'.
D4. Accept θ if $\rho \leq \varepsilon$, and return to *D1*.

There are several advantages to these rejection methods, among them the fact that they are usually easy to code, they generate independent observations (and thus can use embarrassingly parallel computation), and they readily provide estimates of Bayes factors that can be used for model com-

Markov chain Monte Carlo without likelihoods

Paul Marjoram*, John Molitor*, Vincent Plagnol†, and Simon Tavaré†‡

*Biostatistics Division, Department of Preventive Medicine, Keck School of Medicine, and †Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089

Communicated by Michael S. Waterman, University of Southern California, Los Angeles, CA, October 24, 2003 (received for review June 20, 2003)

Many stochastic simulation approaches for generating observations from a posterior distribution depend on knowing a likelihood function. However, for many complex probability models, such likelihoods are either impossible or computationally prohibitive to obtain. Here we present a Markov chain Monte Carlo method for generating observations from a posterior distribution without the use of likelihoods. It can also be used in frequentist applications, in particular for maximum-likelihood estimation. The approach is illustrated by an example of ancestral inference in population genetics. A number of open problems are highlighted in the discussion.

One of the basic problems in Bayesian statistics is the computation of posterior distributions. We imagine data $\mathcal{D}$ generated from a model $\mathcal{M}$ determined by parameters θ, the prior density of which is denoted by $\pi(\theta)$. We assume unless otherwise stated that the data are discrete. The posterior distribution of interest is $f(\theta|\mathcal{D})$, which is given by

A1. Generate θ from $\pi(\cdot)$.
A2. Accept θ with probability $h = \mathbb{P}(\mathcal{D}|\theta)$; return to *A1*.

of ε therefore reflects a tension between computability and accuracy. The method is still honest in that, for a given ρ and ε, we are generating independent and identically distributed observations from $f(\theta|\rho(\mathcal{D}, \mathcal{D}') \leq \varepsilon)$.

When $\mathcal{D}$ is high-dimensional or continuous, this approach can be impractical as well, and then the comparison of $\mathcal{D}'$ with $\mathcal{D}$ can be made by using lower-dimensional summaries of the data. The motivation for this approach is that if the set of statistics $S = (S_1, \ldots, S_p)$ is sufficient for θ, in that $\mathbb{P}(\mathcal{D}|S, \theta)$ is independent of θ, then $f(\theta|\mathcal{D}) = f(\theta|S)$. The normalizing constant $\mathbb{P}(S)$ is typically larger than $\mathbb{P}(\mathcal{D})$, resulting in more acceptances. In practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach. Thus we seek to use knowledge of the particular problem at hand to suggest summary statistics that capture information about θ. With these statistics in hand, we have the following approximate Bayesian computation scheme for data $\mathcal{D}$ summarized by S:

When $\mathcal{D}$ is high-dimensional or continuous, this approach can be impractical as well, and then the comparison of $\mathcal{D}'$ with $\mathcal{D}$ can be made by using lower-dimensional summaries of the data. The

θ, and

hods,

, they

generate independent observations (and thus can use embarrassingly parallel computation), and they readily provide estimates of Baves factors that can be used for model com-

Markov chain Monte Carlo without likelihoods

Paul Marjoram*, John Molitor*, Vincent Plagnol†, and Simon Tavaré†‡

*Biostatistics Division, Department of Preventive Medicine, Keck School of Medicine, and †Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089

Communicated by Michael S. Waterman, University of Southern California, Los Angeles, CA, October 24, 2003 (received for review June 20, 2003)

Many stochastic simulation approaches for generating observations from a posterior distribution depend on knowing a likelihood function. However, for many complex probability models, such likelihoods are either impossible or computationally prohibitive to obtain. Here we present a Markov chain Monte Carlo method for generating observations from a posterior distribution without the use of likelihoods. It can also be used in frequentist applications, in particular for maximum-likelihood estimation. The approach is illustrated by an example of ancestral inference in population genetics. A number of open problems are highlighted in the discussion.

One of the basic problems in Bayesian statistics is the computation of posterior distributions. We imagine data $\mathcal{D}$ generated from a model $\mathcal{M}$ determined by parameters θ, the prior density of which is denoted by $\pi(\theta)$. We assume unless otherwise stated that the data are discrete. The posterior distribution of interest is $f(\theta|\mathcal{D})$, which is given by

A1. Generate θ from $\pi(\cdot)$.
A2. Accept θ with probability $h = \mathbb{P}(\mathcal{D}|\theta)$; return to *A1*.

of ε therefore reflects a tension between computability and accuracy. The method is still honest in that, for a given ρ and ε, we are generating independent and identically distributed observations from $f(\theta|\rho(\mathcal{D}, \mathcal{D}') \leq \varepsilon)$.

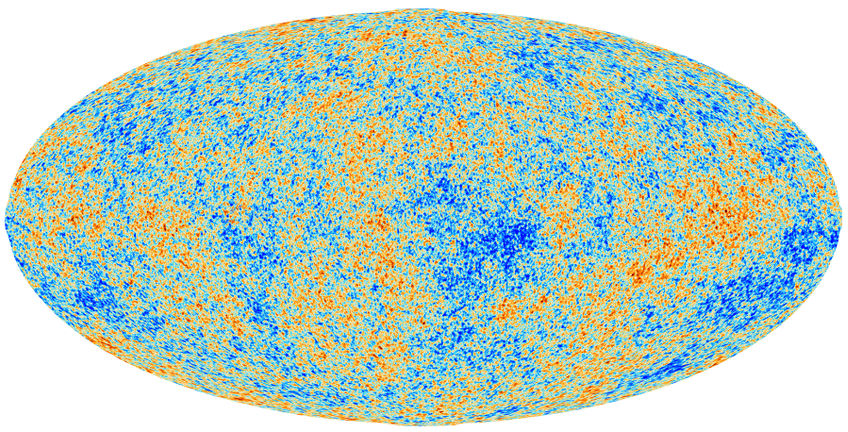
When $\mathcal{D}$ is high-dimensional or continuous, this approach can be impractical as well, and then the comparison of $\mathcal{D}'$ with $\mathcal{D}$ can be made by using lower-dimensional summaries of the data. The motivation for this approach is that if the set of statistics $S = (S_1, \ldots, S_p)$ is sufficient for θ, in that $\mathbb{P}(\mathcal{D}|S, \theta)$ is independent of θ, then $f(\theta|\mathcal{D}) = f(\theta|S)$. The normalizing constant $\mathbb{P}(S)$ is typically larger than $\mathbb{P}(\mathcal{D})$, resulting in more acceptances. In practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach. Thus we seek to use knowledge of the particular problem at hand to suggest summary statistics that capture information about θ. With these statistics in hand, we have the following approximate Bayesian computation scheme for data $\mathcal{D}$ summarized by S:

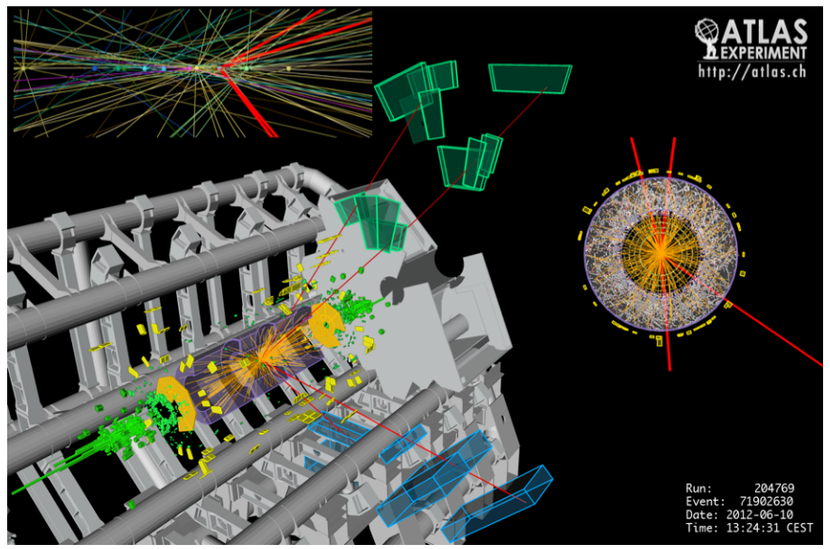
practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach.

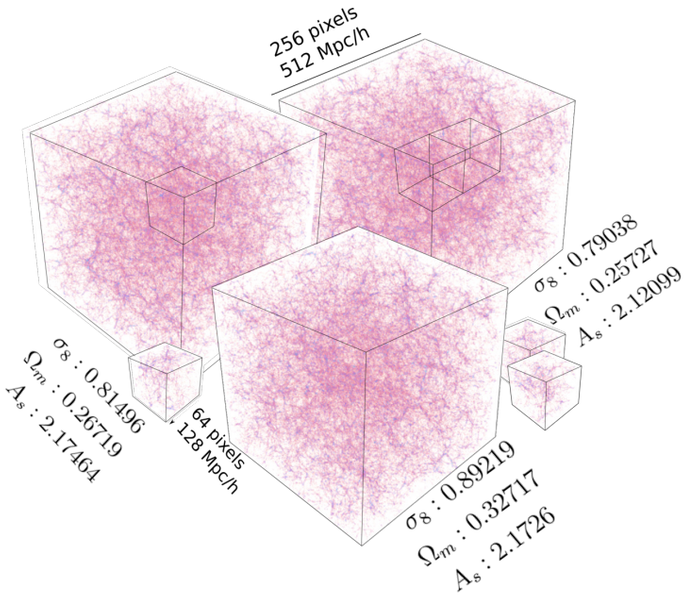
θ, and

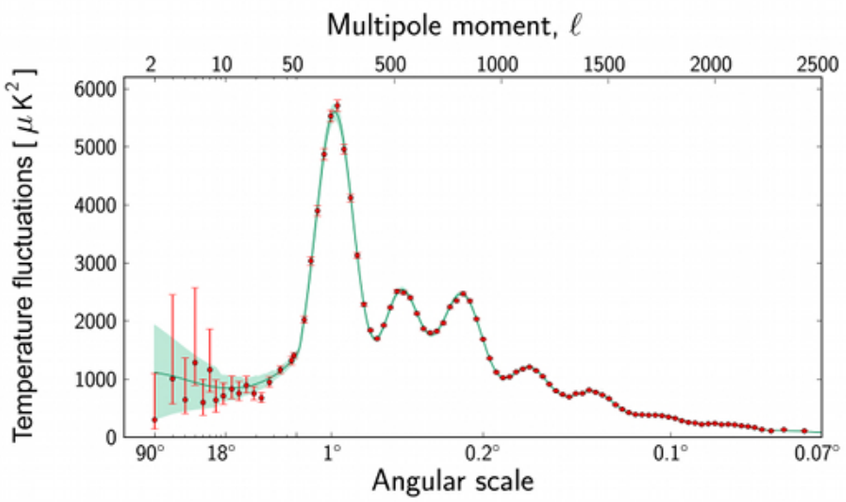
nods,
, they
generate independent observations (and thus can use embarrassingly parallel computation), and they readily provide estimates of Baves factors that can be used for model com-

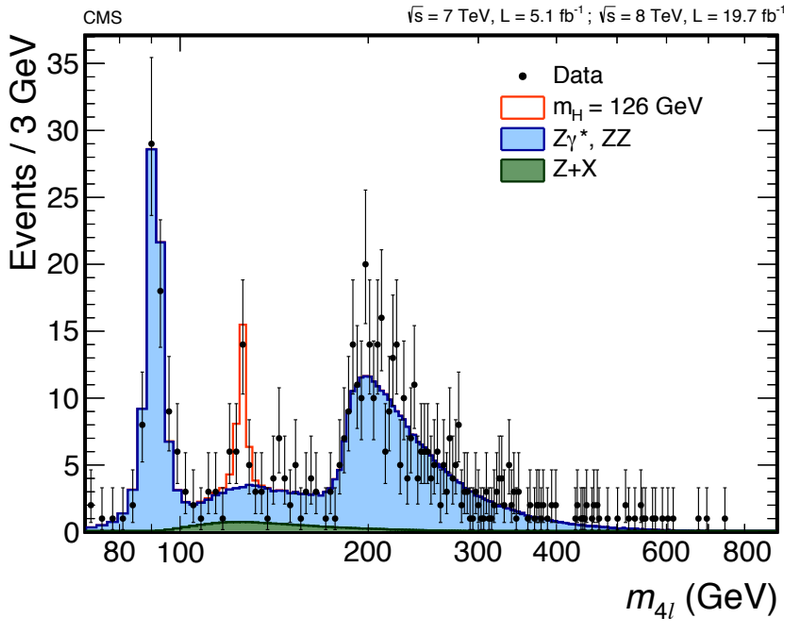
PREDICTION: THE FORWARD MODEL

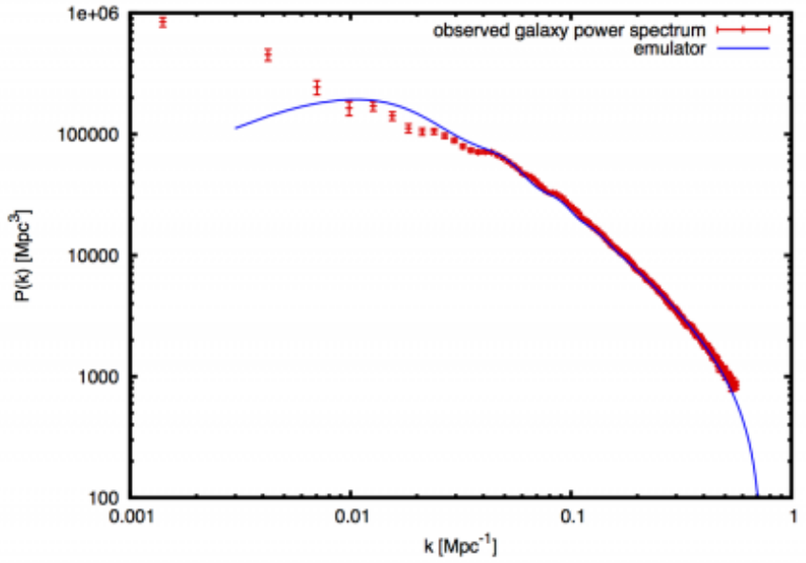




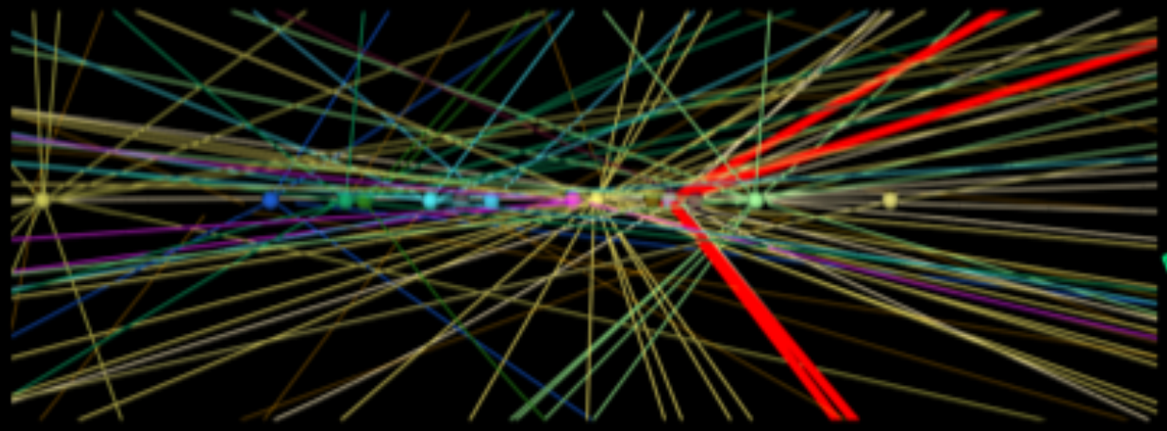




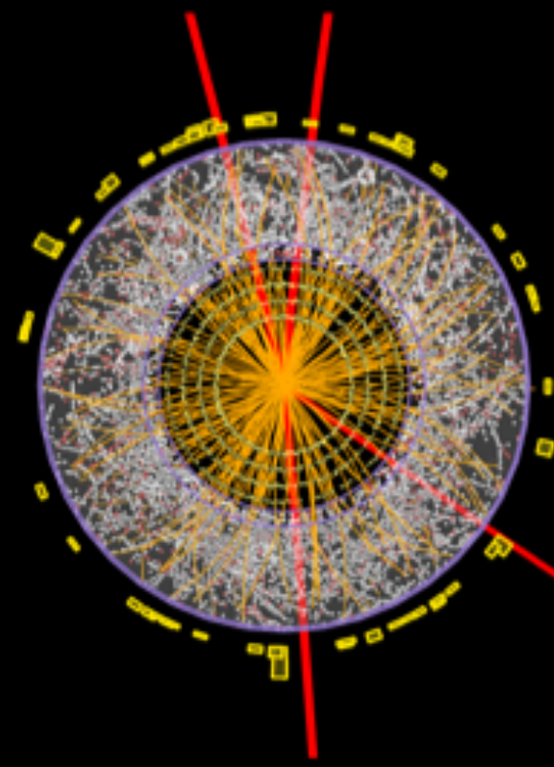




THE HIGGS BOSON



ATLAS EXPERIMENT
http://atlas.ch



Run: 204769
Event: 71902630
Date: 2012-06-10
Time: 13:24:31 CEST

PREDICTIONS IN PARTICLE PHYSICS

$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$

$$+ \underbrace{\bar{L}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})L + \bar{R}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu})R}_{\text{kinetic energies and electroweak interactions of fermions}}$$

$$+ \underbrace{\frac{1}{2}\left|(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})\,\phi\right|^{2} - V(\phi)}_{W^{\pm},Z,\gamma,\text{and Higgs masses and couplings}}$$

$$+ \underbrace{g''(\bar{q}\gamma^{\mu}T_{a}q)\,G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} + \underbrace{(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.)}_{\text{fermion masses and couplings to Higgs}}$$

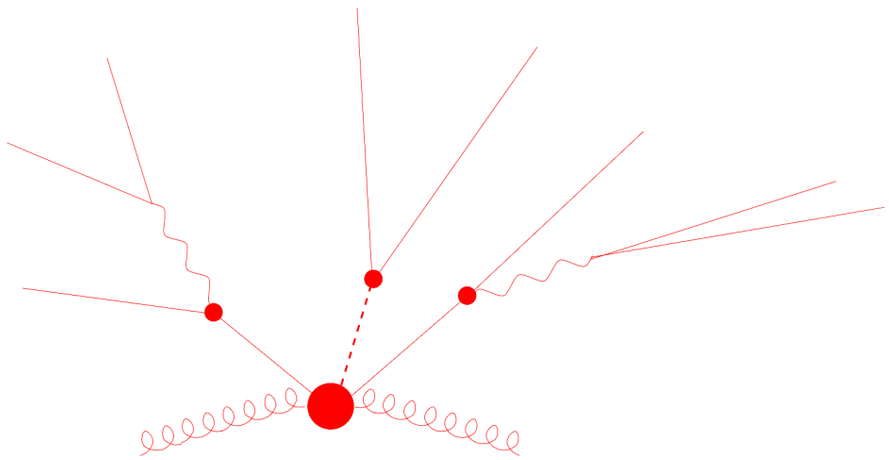
PREDICTIONS IN PARTICLE PHYSICS

$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$

$$+ \underbrace{\bar{L}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})L + \bar{R}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu})R}_{\text{kinetic energies and electroweak interactions of fermions}}$$

$$+ \underbrace{\frac{1}{2}\left|(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})\,\phi\right|^{2} - V(\phi)}_{W^{\pm},Z,\gamma,\text{and Higgs masses and couplings}}$$

$$+ \underbrace{g''(\bar{q}\gamma^{\mu}T_{a}q)\,G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} + \underbrace{(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.)}_{\text{fermion masses and couplings to Higgs}}$$



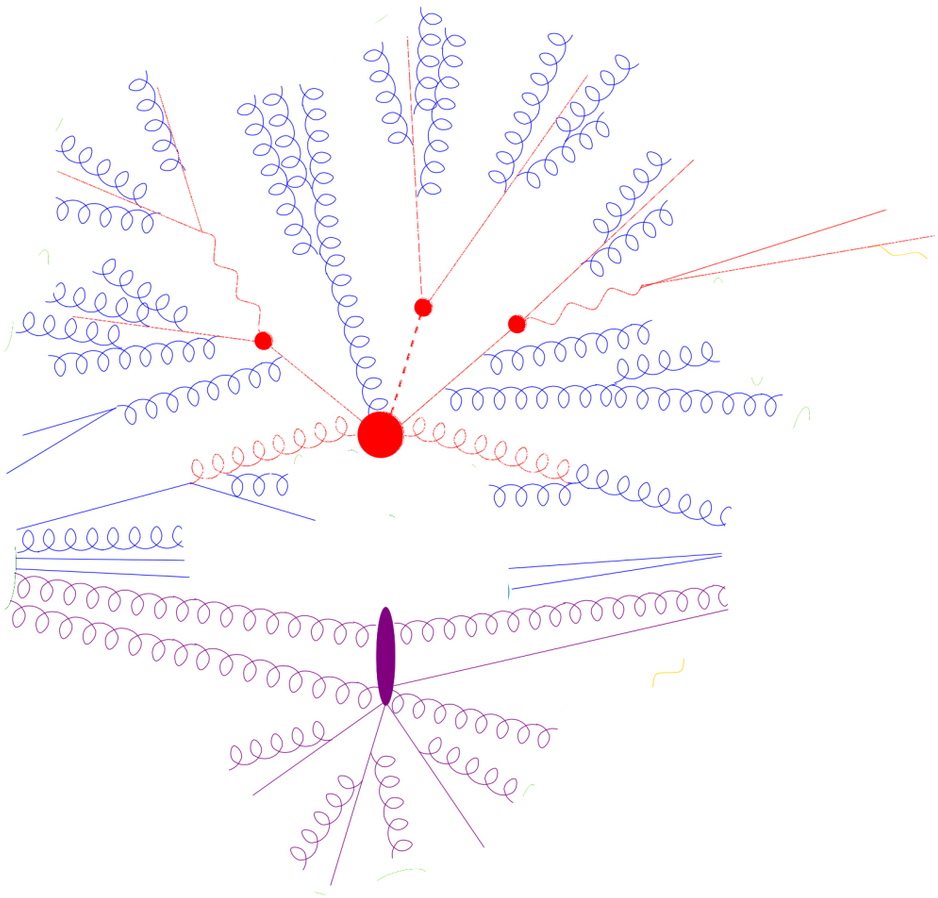
PREDICTIONS IN PARTICLE PHYSICS

$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$

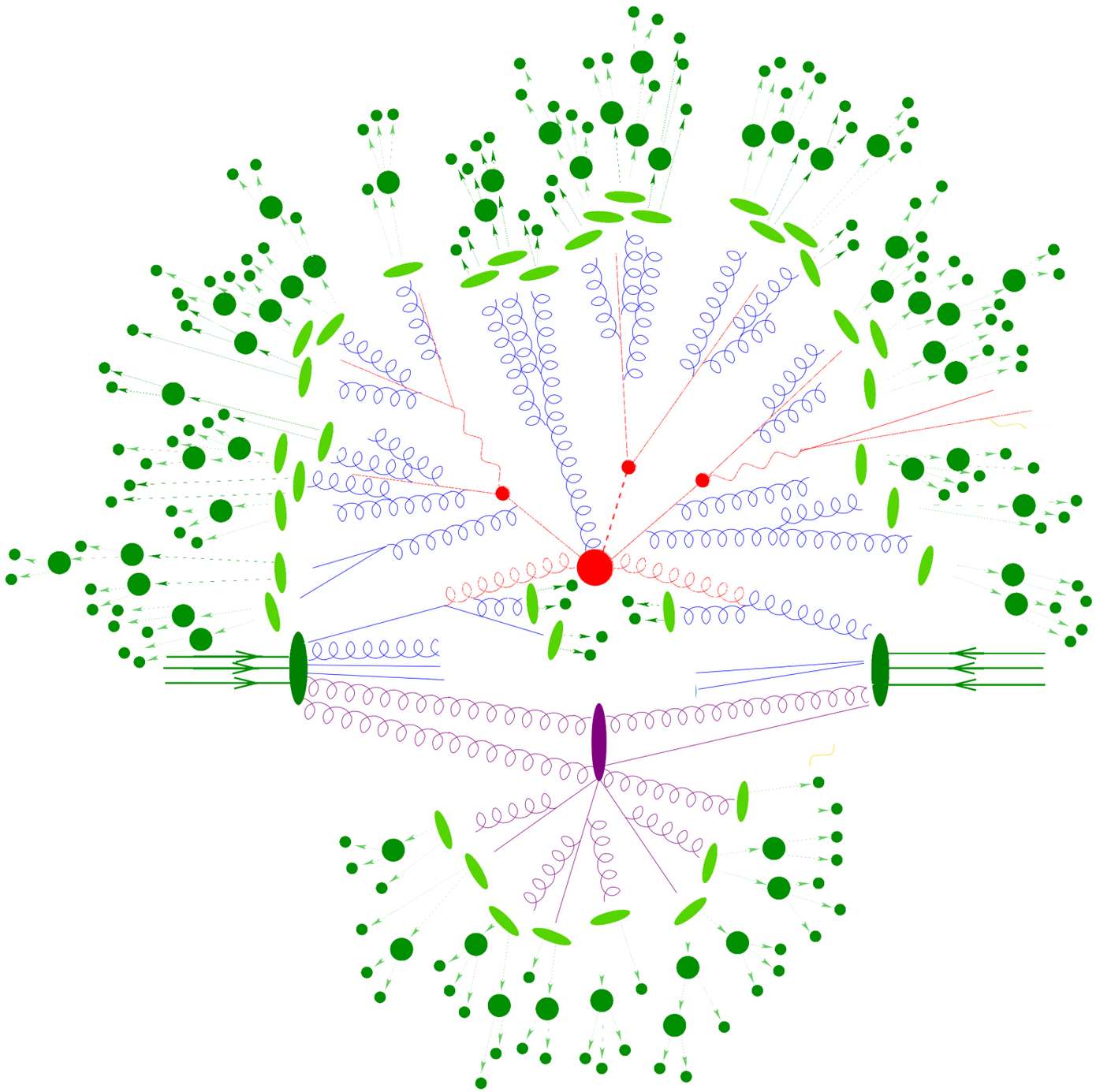
$$+ \underbrace{\bar{L}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})L + \bar{R}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu})R}_{\text{kinetic energies and electroweak interactions of fermions}}$$

$$+ \underbrace{\frac{1}{2}\left|(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})\,\phi\right|^{2} - V(\phi)}_{W^{\pm},Z,\gamma,\text{and Higgs masses and couplings}}$$

$$+ \underbrace{g''(\bar{q}\gamma^{\mu}T_{a}q)\,G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} + \underbrace{(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.)}_{\text{fermion masses and couplings to Higgs}}$$



PREDICTIONS IN PARTICLE PHYSICS



$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$

$$+ \underbrace{\bar{L}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})L + \bar{R}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu})R}_{\text{kinetic energies and electroweak interactions of fermions}}$$

$$+ \underbrace{\frac{1}{2}\left|(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})\,\phi\right|^{2} - V(\phi)}_{W^{\pm},Z,\gamma,\text{and Higgs masses and couplings}}$$

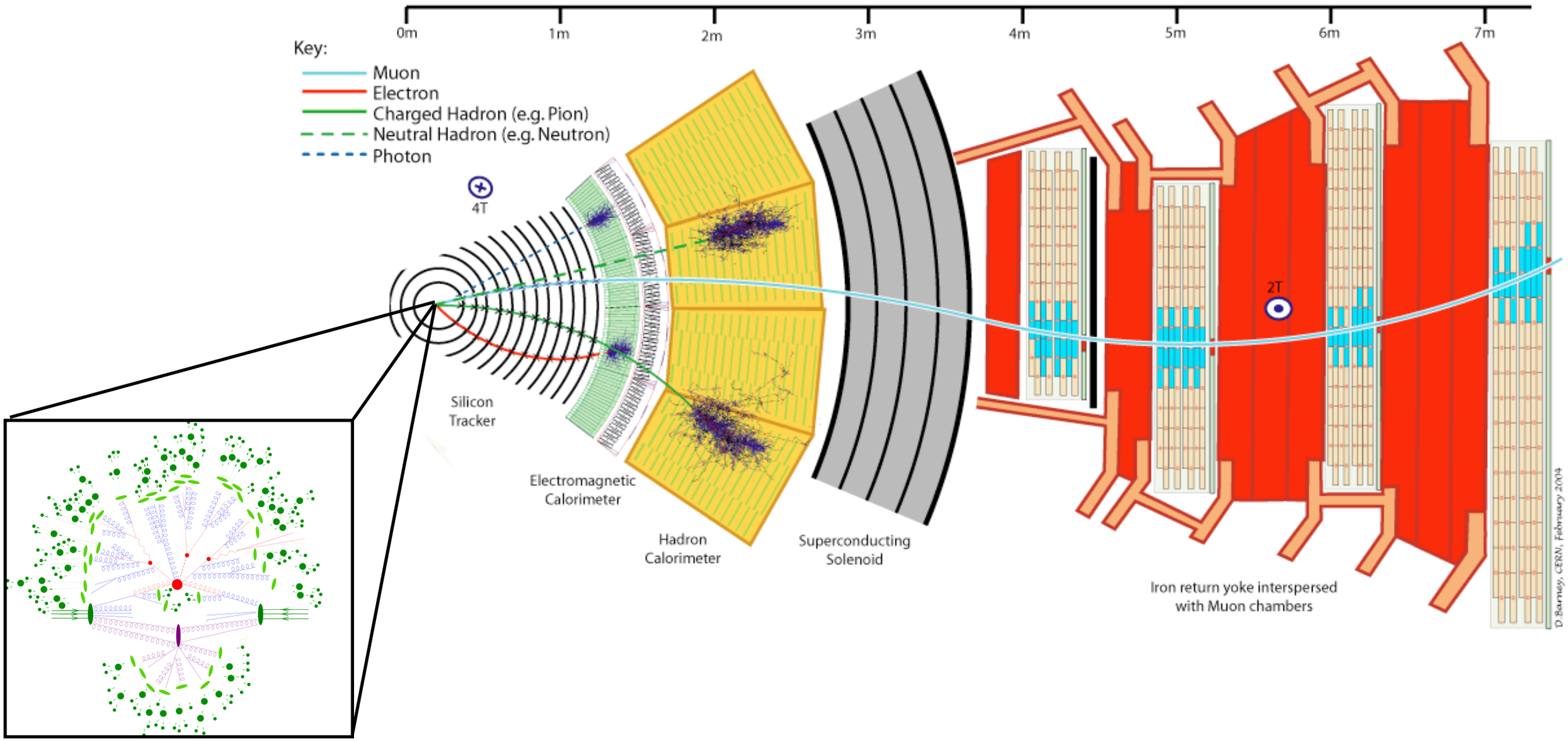
$$+ \underbrace{g''(\bar{q}\gamma^{\mu}T_{a}q)\,G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} + \underbrace{(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.)}_{\text{fermion masses and couplings to Higgs}}$$

DETECTOR SIMULATION

Conceptually: Prob(detector response | particles)

Implementation: Monte Carlo integration over micro-physics

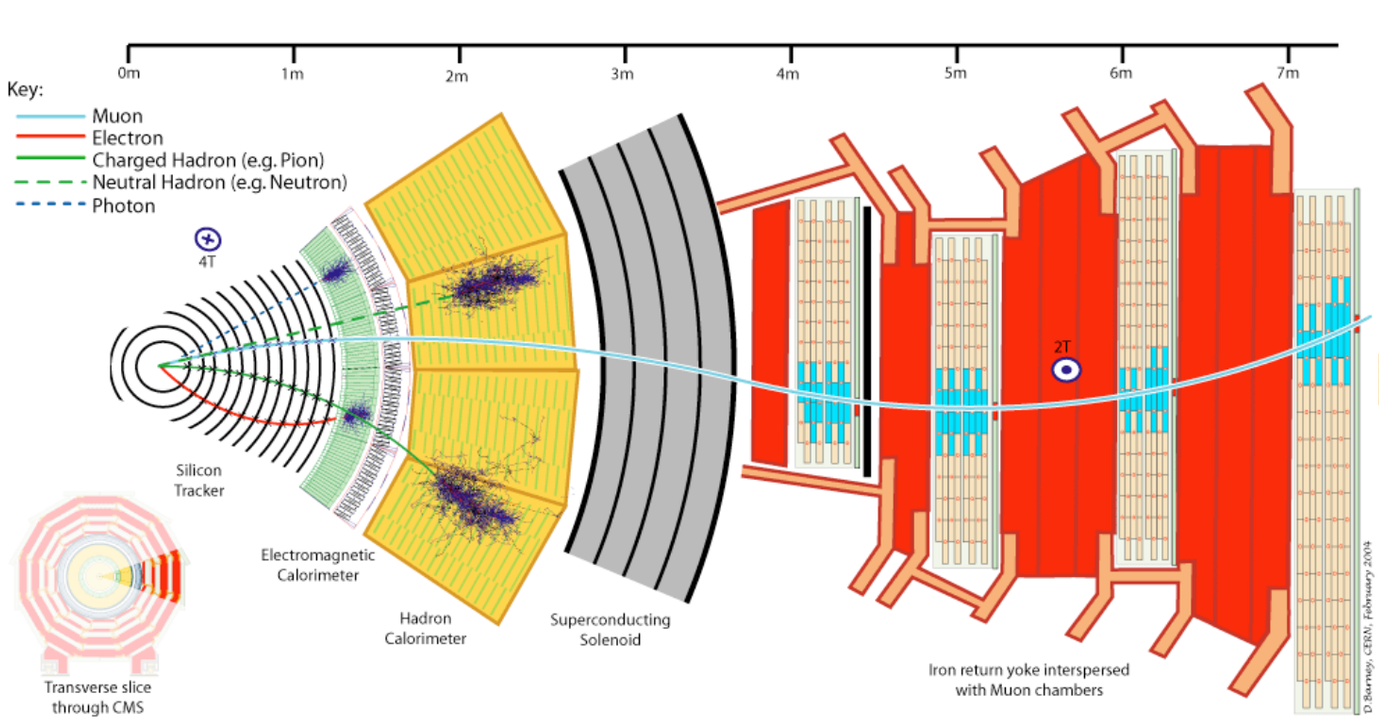
Consequence: evaluation of the likelihood is intractable

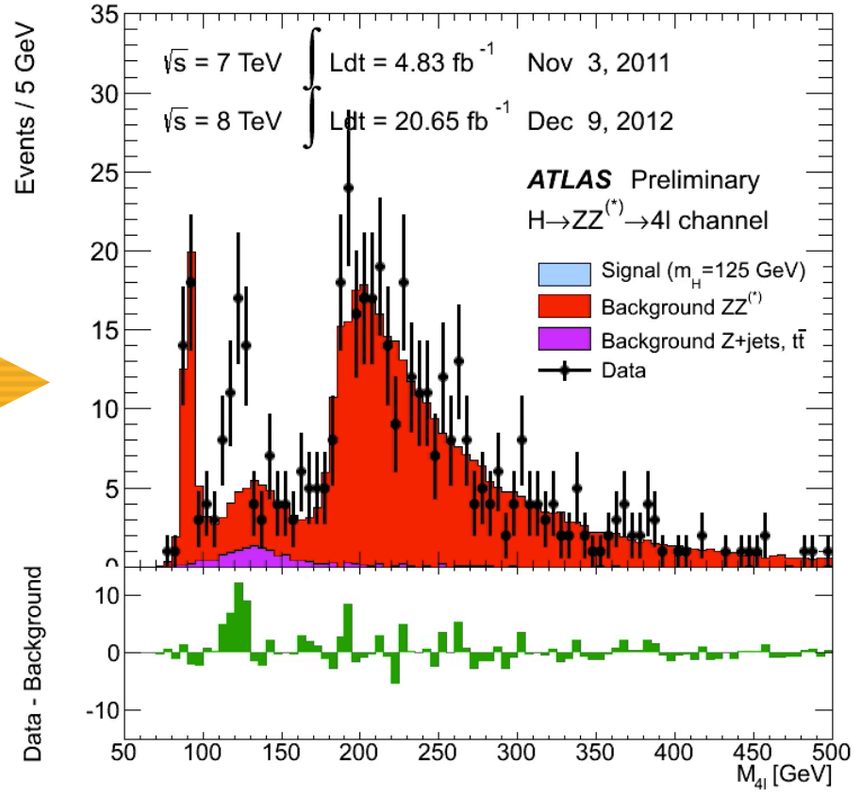


10^8 SENSORS → 1 REAL-VALUED QUANTITY

Most measurements and searches for new particles at the LHC are based on the distribution of a single summary statistic

- choosing a good summary statistic (feature engineering) is a task for a skilled physicist and tailored to the goal of measurement or new particle search
- likelihood $p(x|\theta)$ **approximated** using histograms (univariate density estimation)

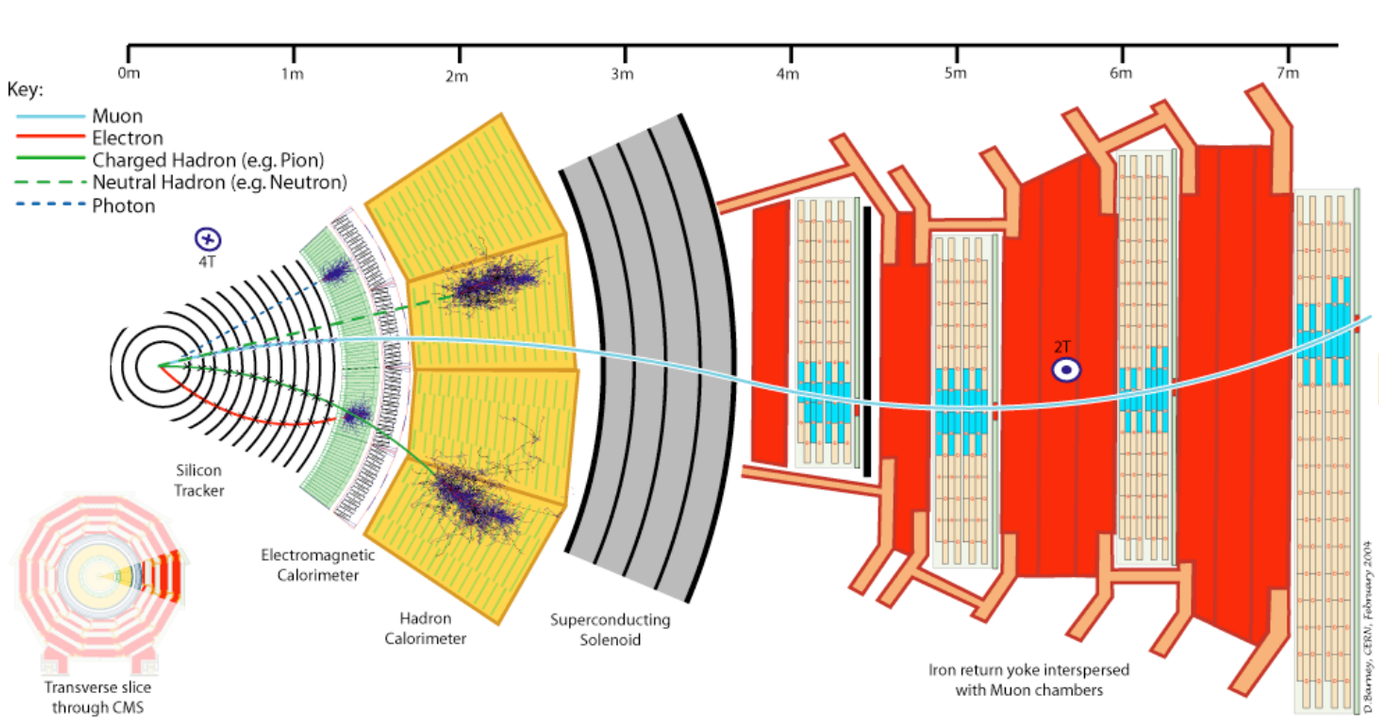


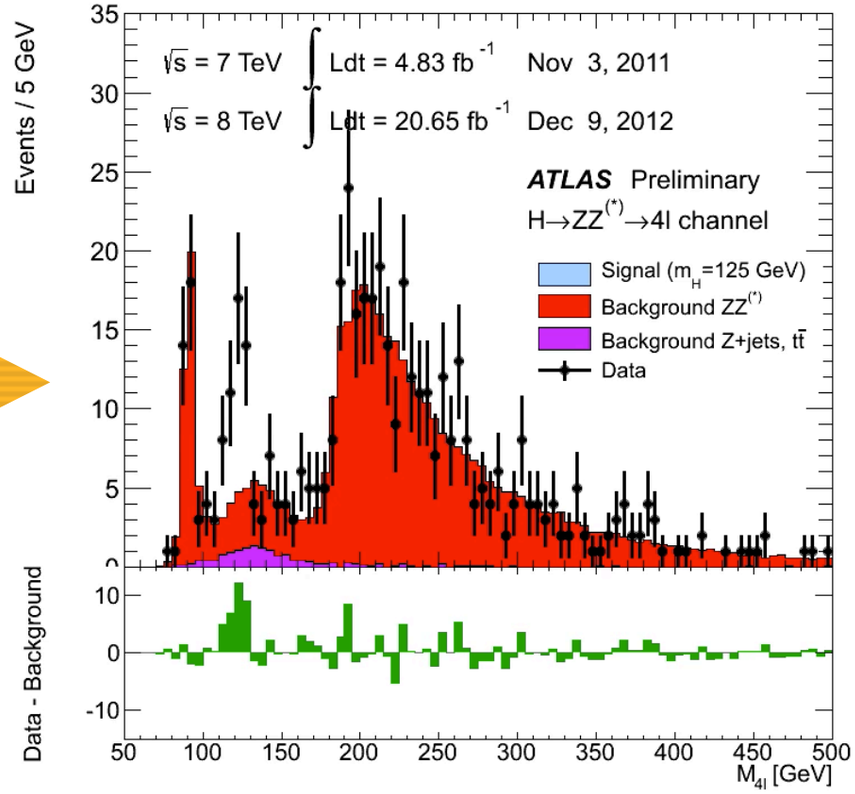


10^8 SENSORS → 1 REAL-VALUED QUANTITY

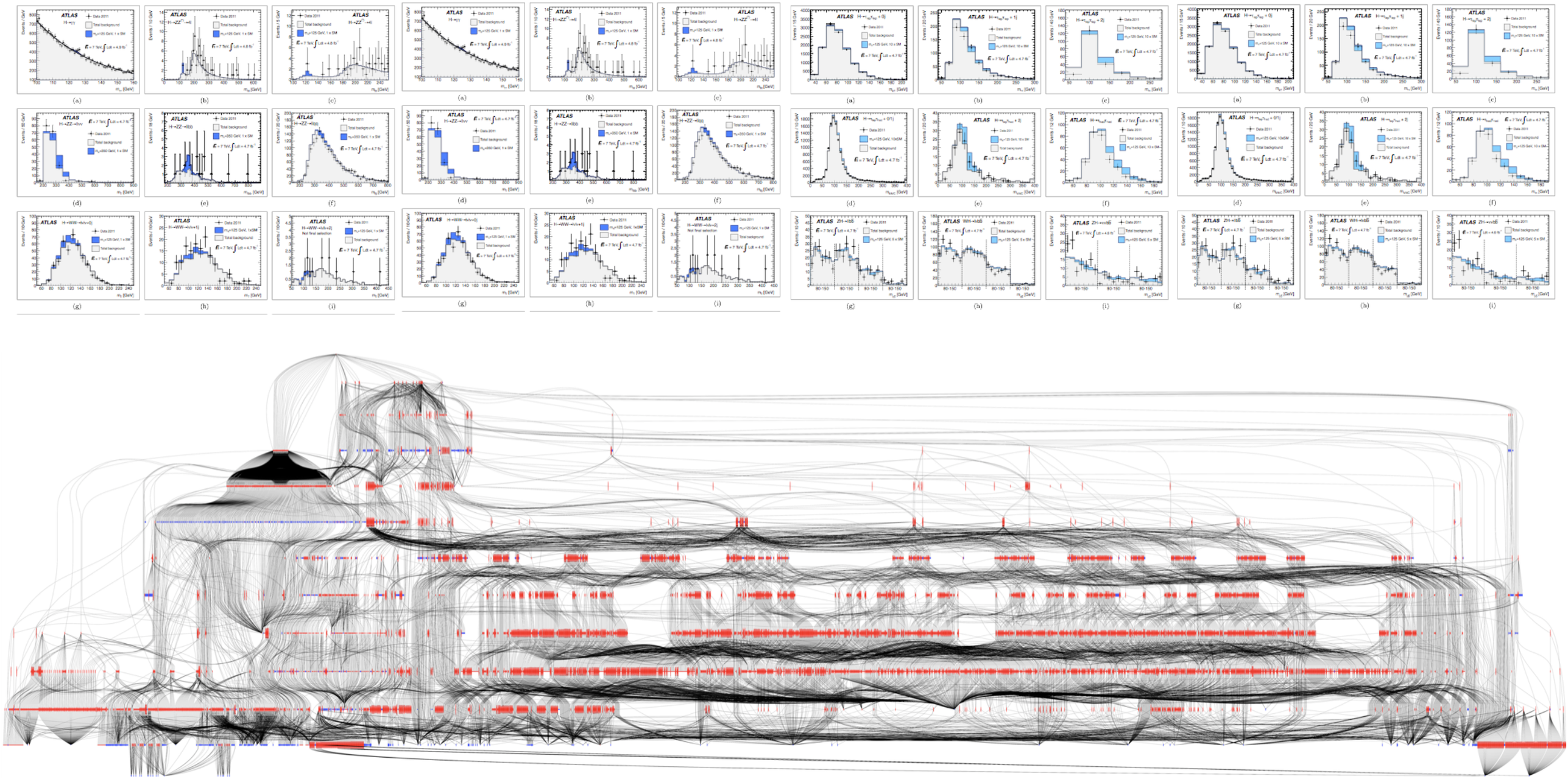
Most measurements and searches for new particles at the LHC are based on the distribution of a single summary statistic

- choosing a good summary statistic (feature engineering) is a task for a skilled physicist and tailored to the goal of measurement or new particle search
- likelihood $p(x|\theta)$ **approximated** using histograms (univariate density estimation)





THE DISCOVERY OF THE HIGGS BOSON



$$\mathbf{f}_{\text{tot}}(\mathcal{D}_{\text{sim}}, \mathcal{G}|\boldsymbol{\alpha}) = \prod_{c \in \text{channels}} \left[\text{Pois}(n_c|\nu_c(\boldsymbol{\alpha})) \prod_{e=1}^{n_c} f_c(x_{ce}|\boldsymbol{\alpha}) \right] \cdot \prod_{p \in \mathbb{S}} f_p(a_p|\alpha_p)$$

THE CRUX, AN INTRACTABLE INTEGRAL

observed

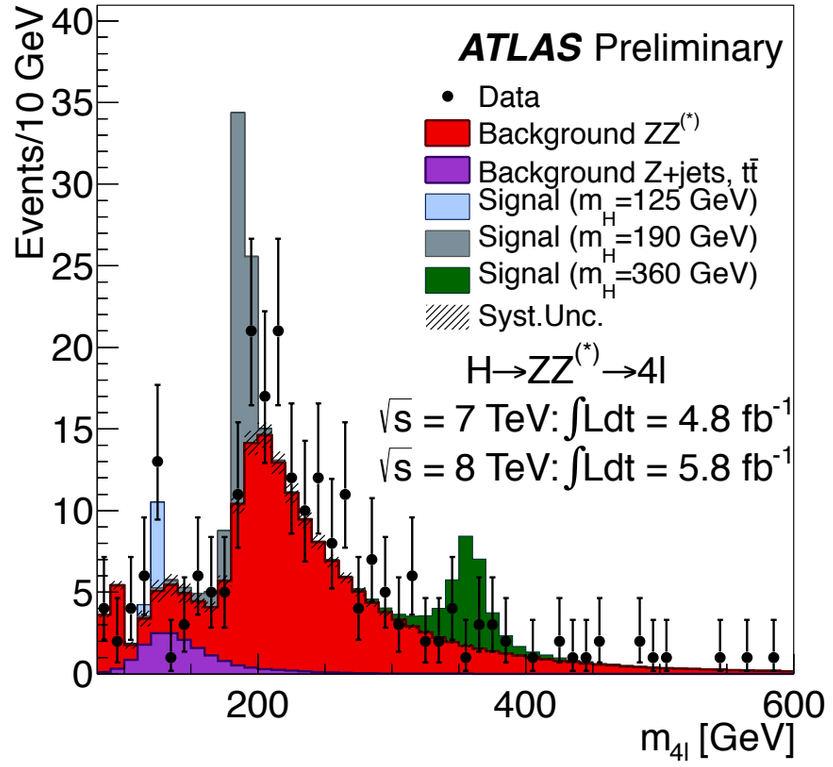
MC Sampling

simulation

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

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

histogram approximation



THE CRUX, AN INTRACTABLE INTEGRAL

observed

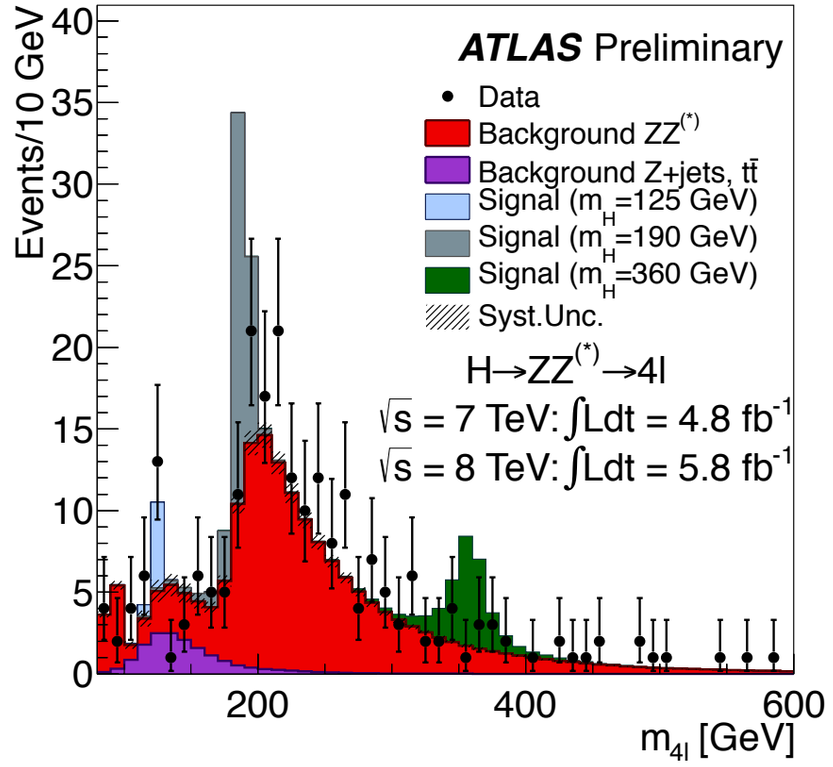
MC Sampling

simulation

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

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

histogram approximation

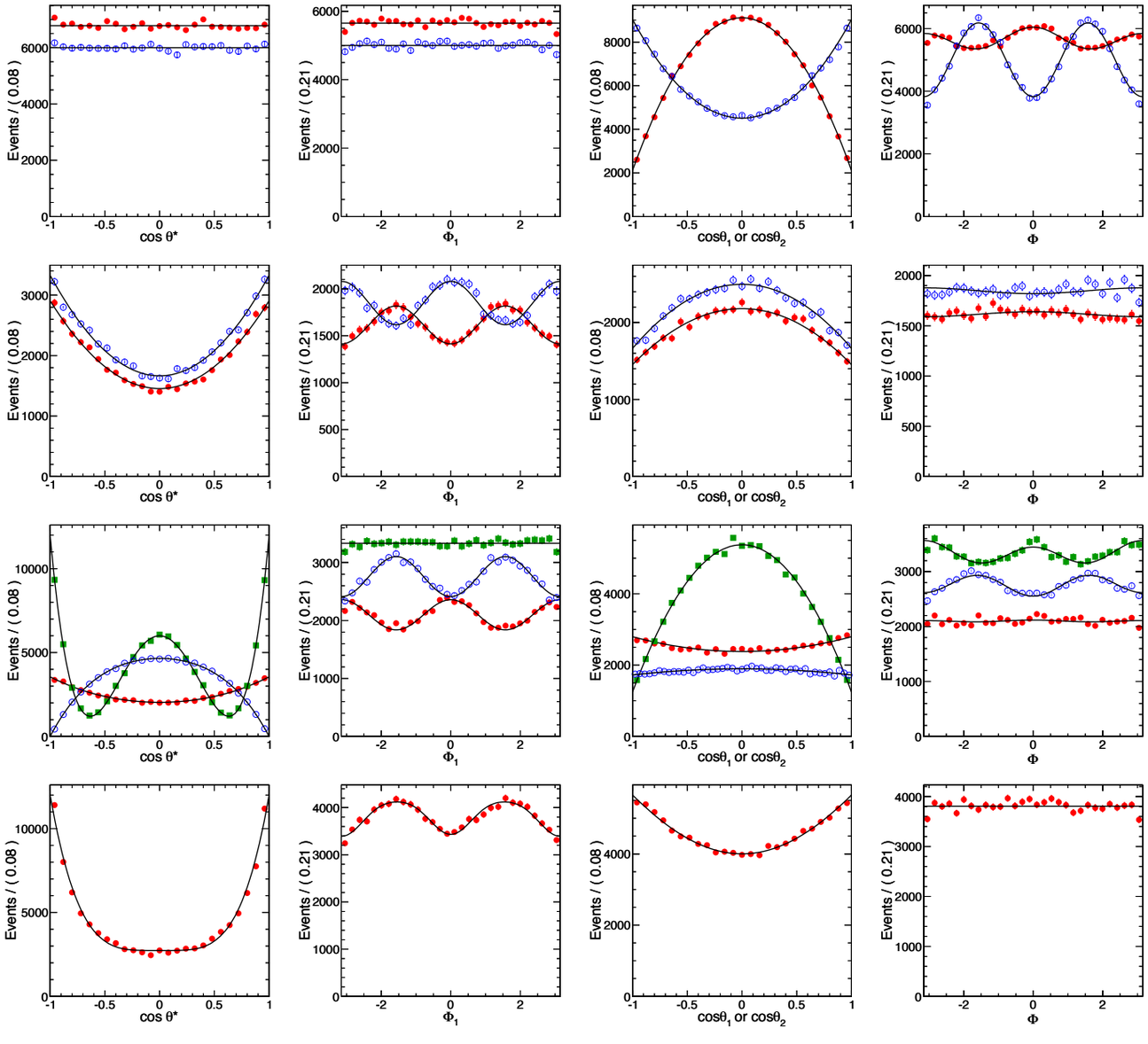


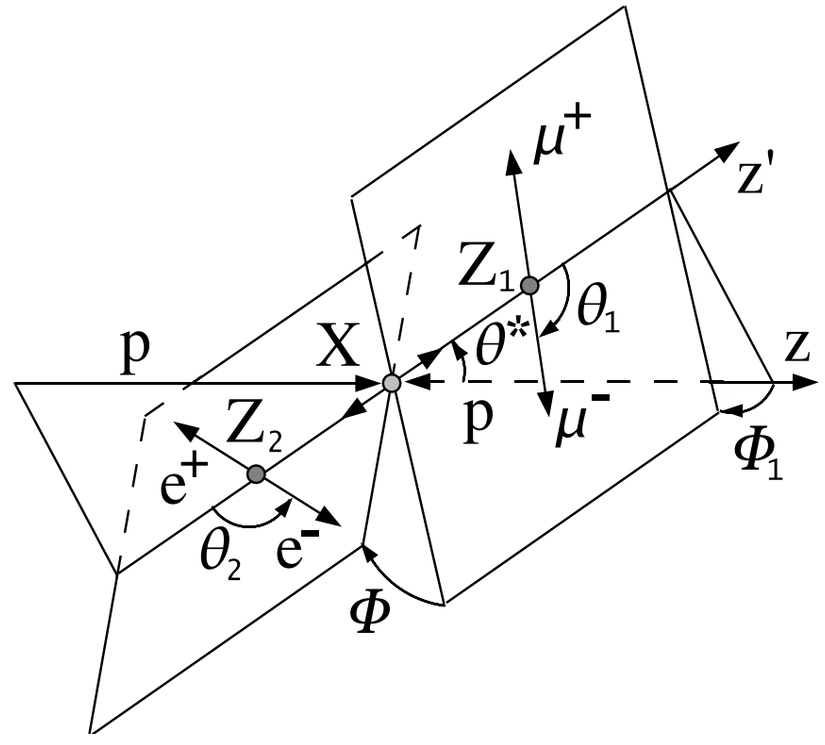
This doesn’t scale if x is high dimensional!

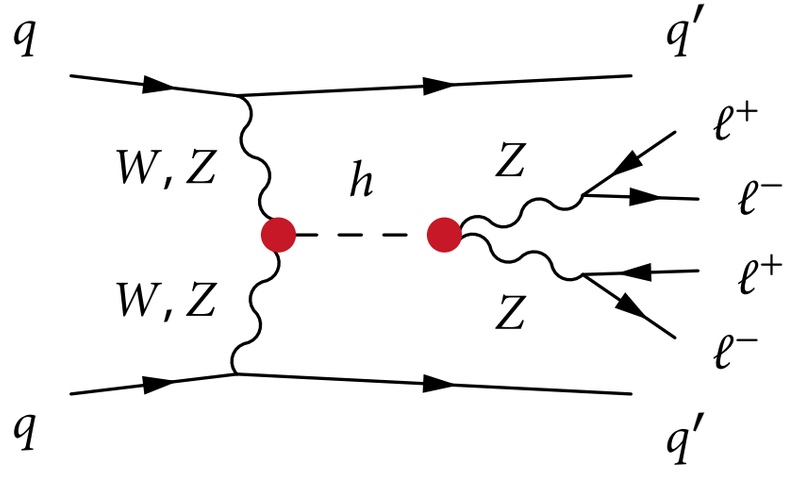
HIGH DIMENSIONAL EXAMPLE

When looking for deviations from the standard model Higgs, we would like to look at all sorts of kinematic correlations

- thus each observation **x** is high-dimensional







ICML 2017 Workshop on Implicit Models

Workshop Aims

Probabilistic models are an important tool in machine learning. They form the basis for models that generate realistic data, uncover hidden structure, and make predictions. Traditionally, probabilistic models in machine learning have focused on prescribed models. Prescribed models specify a joint density over observed and hidden variables that can be easily evaluated. The requirement of a tractable density simplifies their learning but limits their flexibility --- several real world phenomena are better described by simulators that do not admit a tractable density. Probabilistic models defined only via the simulations they produce are called implicit models.

Arguably starting with generative adversarial networks, research on implicit models in machine learning has exploded in recent years. This workshop's aim is to foster a discussion around the recent developments and future directions of implicit models.

Implicit models have many applications. They are used in ecology where models simulate animal populations over time; they are used in phylogeny, where simulations produce hypothetical ancestry trees; they are used in physics to generate particle simulations for high energy processes. Recently, implicit models have been used to improve the state-of-the-art in image and content generation. Part of the workshop's focus is to discuss the commonalities among applications of implicit models.

Of particular interest at this workshop is to unite fields that work on implicit models. For example:

- **Generative adversarial networks** (a NIPS 2016 workshop) are implicit models with an adversarial training scheme.
- Recent advances in **variational inference** (a NIPS 2015 and 2016 workshop) have leveraged implicit models for more accurate approximations.
- **Approximate Bayesian computation** (a NIPS 2015 workshop) focuses on posterior inference for models with implicit likelihoods.
- Learning implicit models is deeply connected to **two sample testing, density ratio and density difference** estimation.

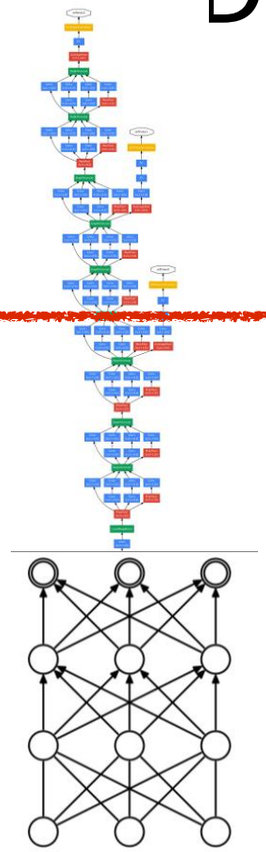
We hope to bring together these different views on implicit models, identifying their core challenges and combining their innovations.

PHYSICS AT THE INTERSECTION

Max Welling

We can leverage both the power of deep learning and inject our expert physics knowledge

Discriminative or Generative?



-Deep Learning

-Kernel Methods

-Random Forests

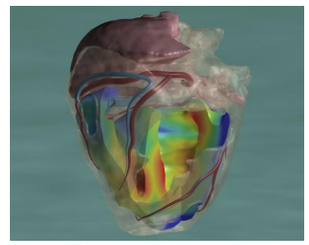
-Boosting

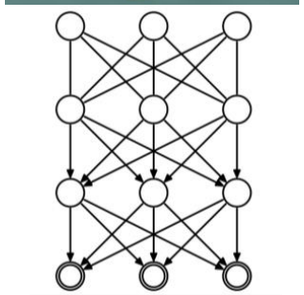
Exciting

-Bayesian Networks

-Probabilistic Programs

-Simulator Models





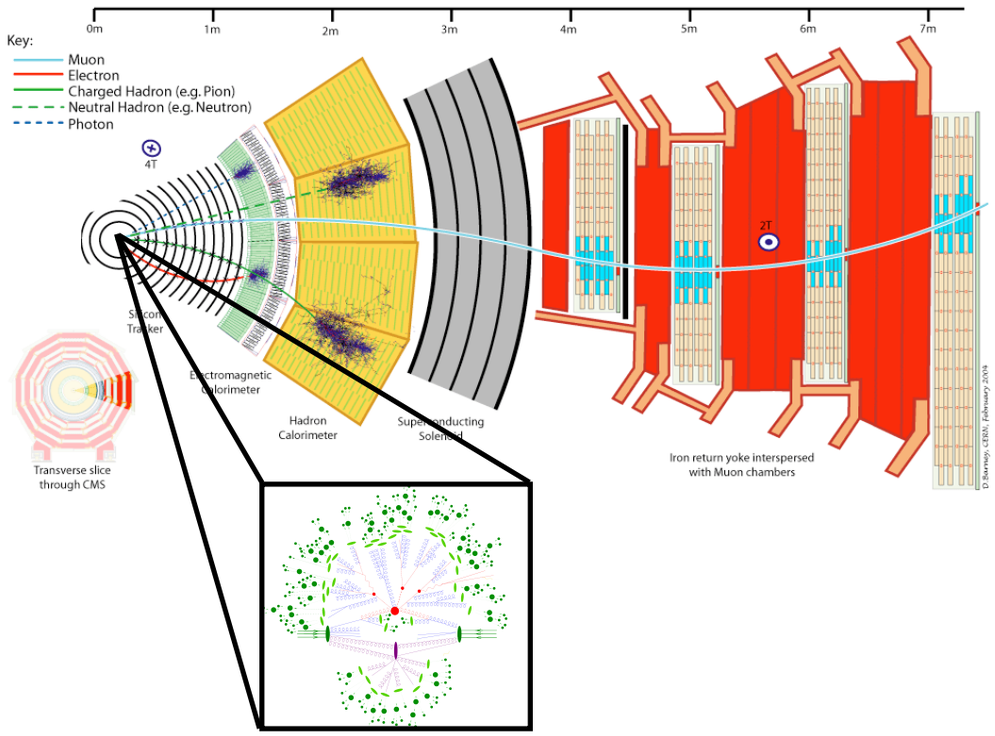
- Advantages discriminative models:
  - Flexible map from input to target (low bias)
  - Efficient training algorithms available
  - Solve the problem you are evaluating on.
  - Very successful and accurate!

- Advantages generative models:
  - Inject expert knowledge
  - Model causal relations
  - Interpretable
  - Data efficient
  - More robust to domain shift
  - Facilitate un/semi-supervised learning

TWO APPROACHES TO LIKELIHOOD FREE INFERENCE

Use simulator

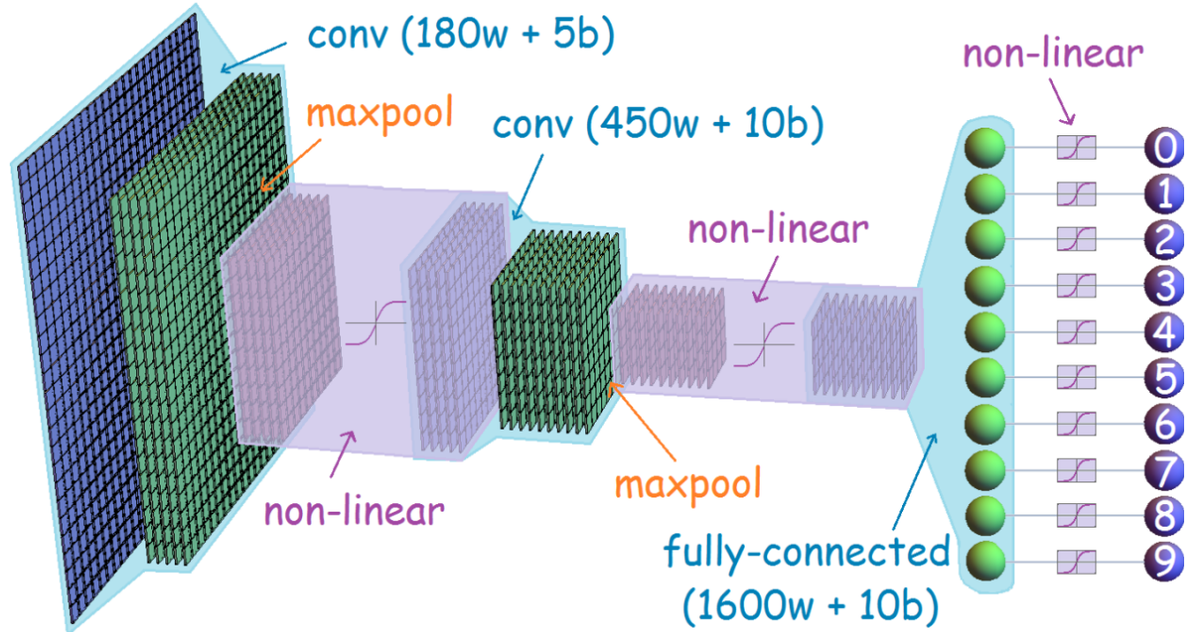
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator

(with deep learning)

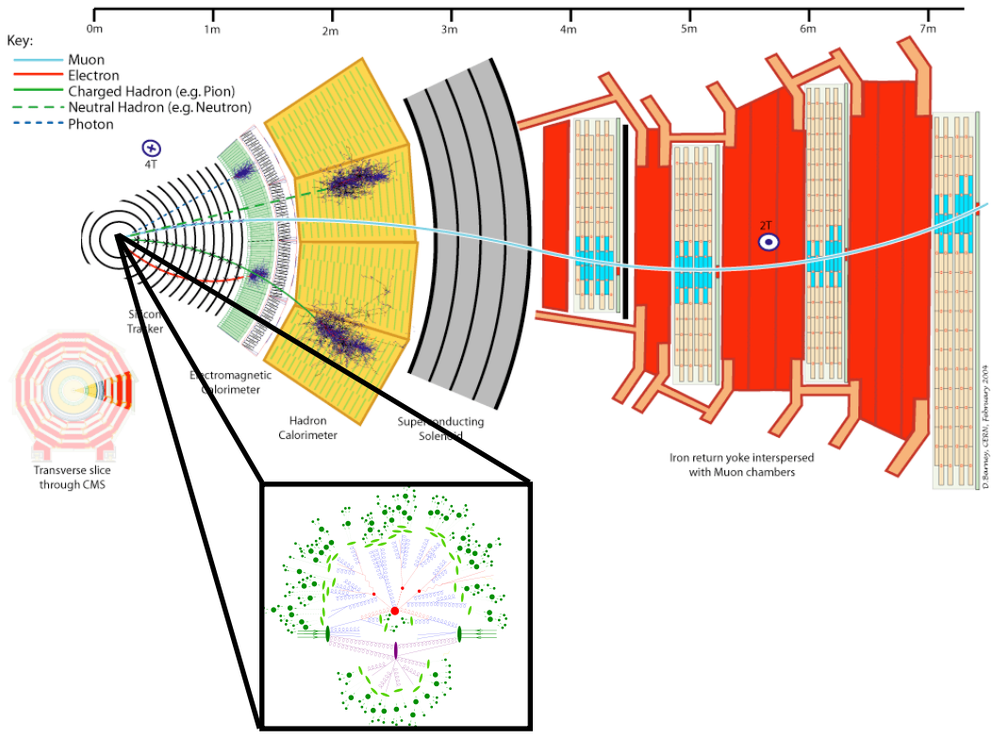


- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

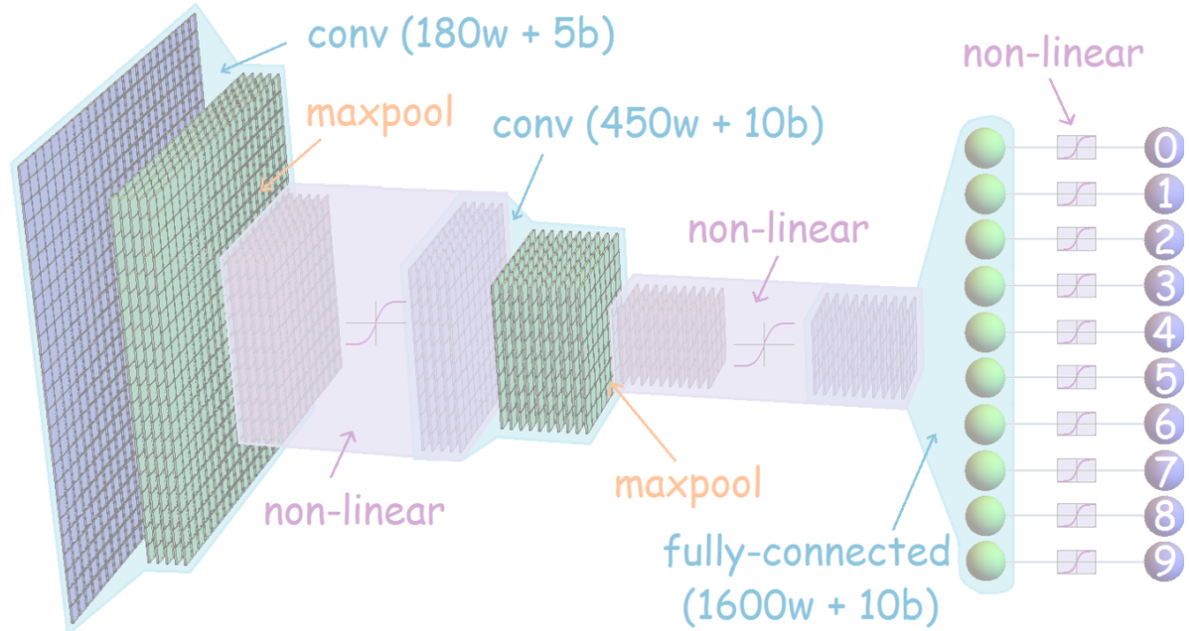
TWO APPROACHES TO LIKELIHOOD FREE INFERENCE

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

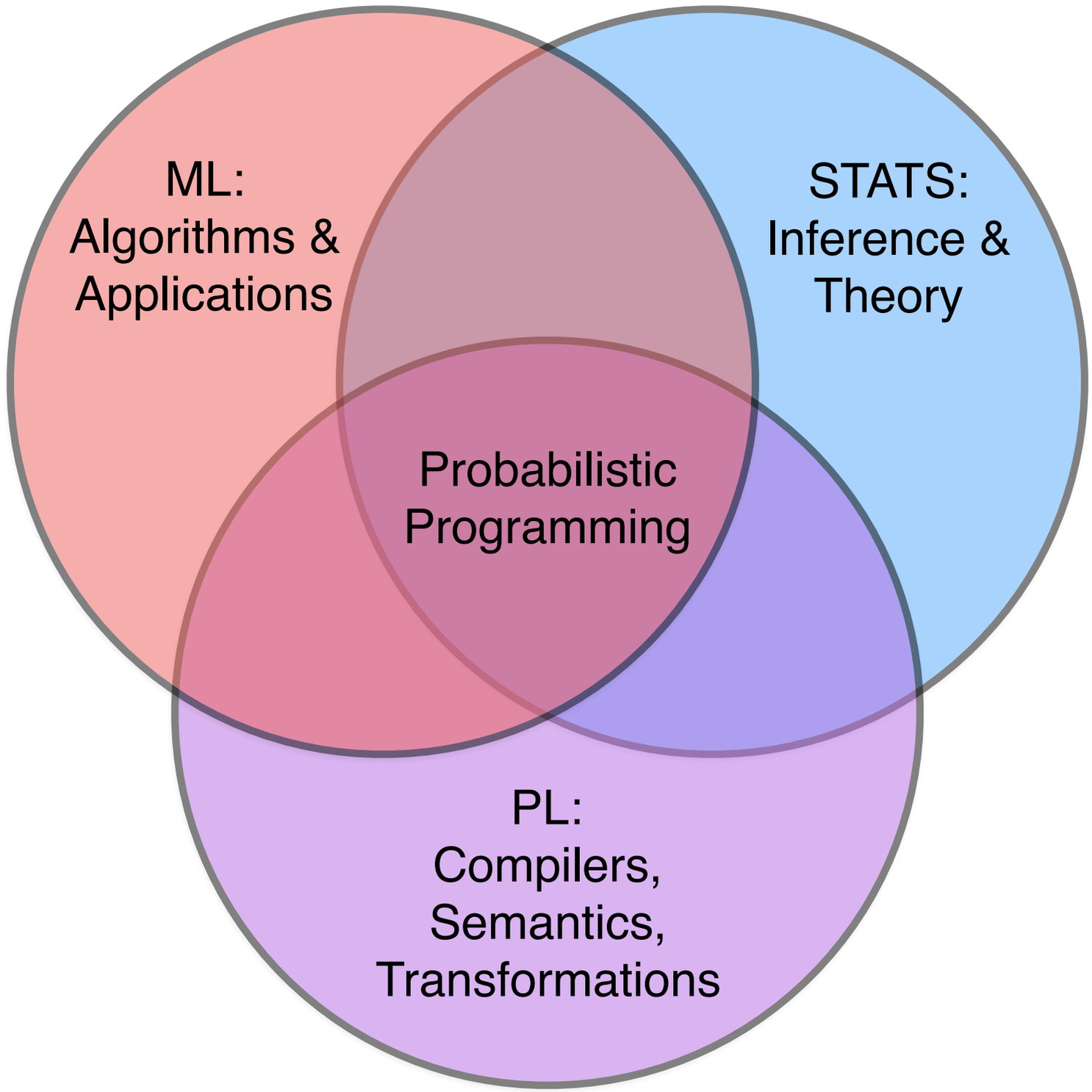
Learn simulator
(with deep learning)



- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

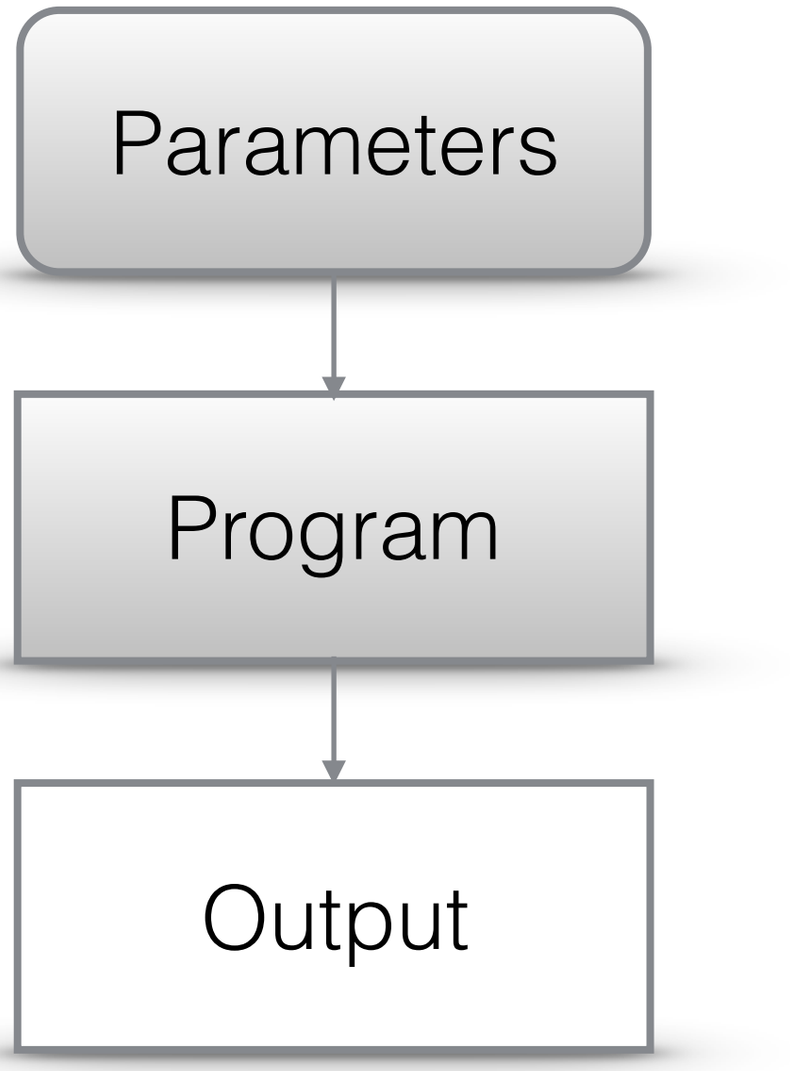
Probabilistic Programming



Intuition

[slides, Frank Wood]

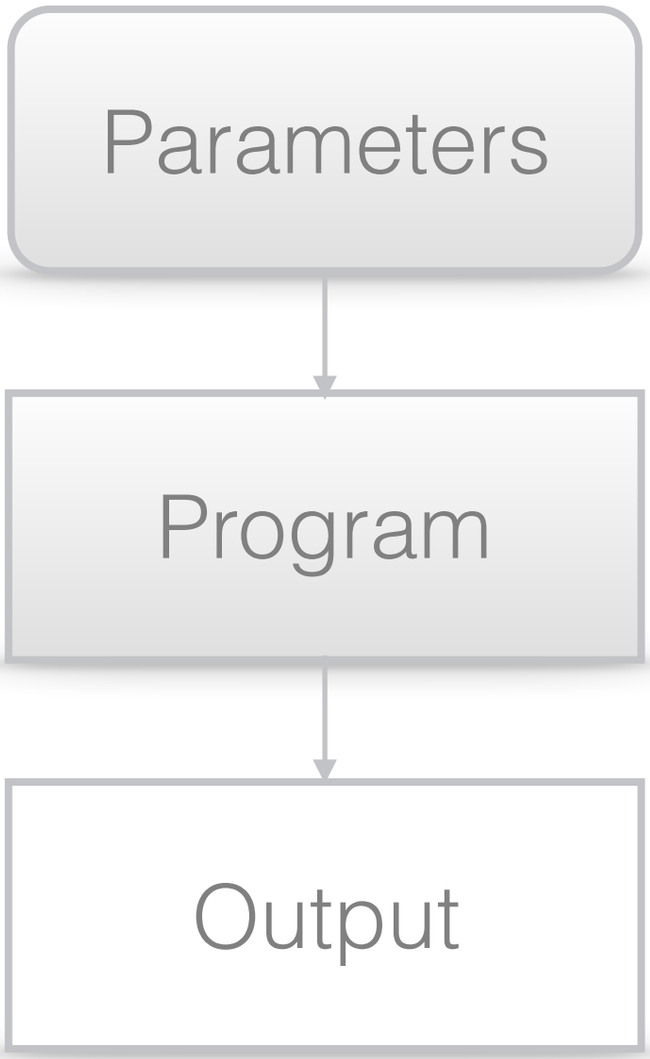
Intuition



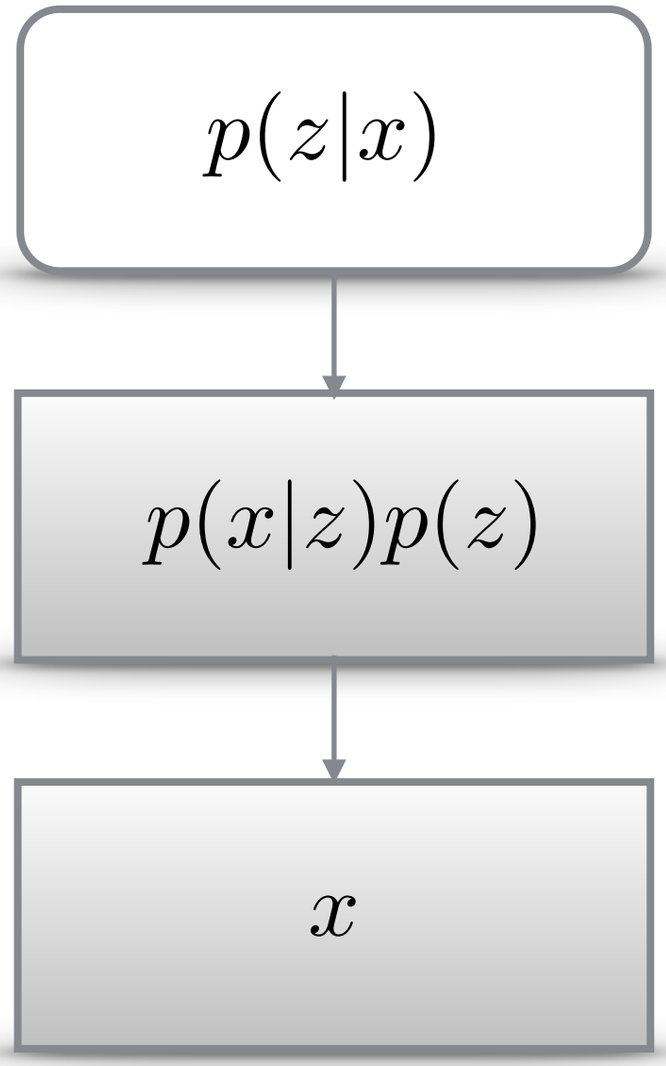
CS

[slides, Frank Wood]

Intuition



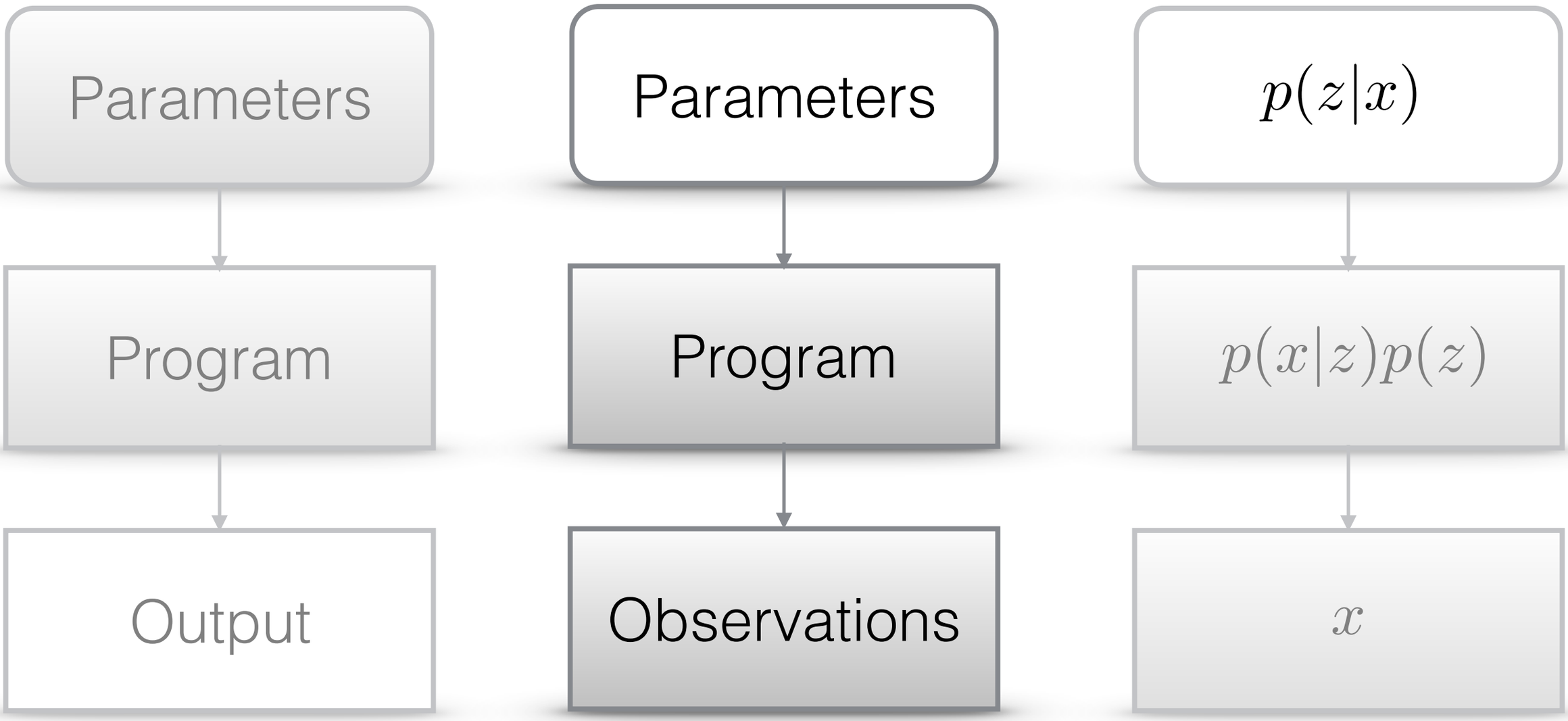
CS



Statistics

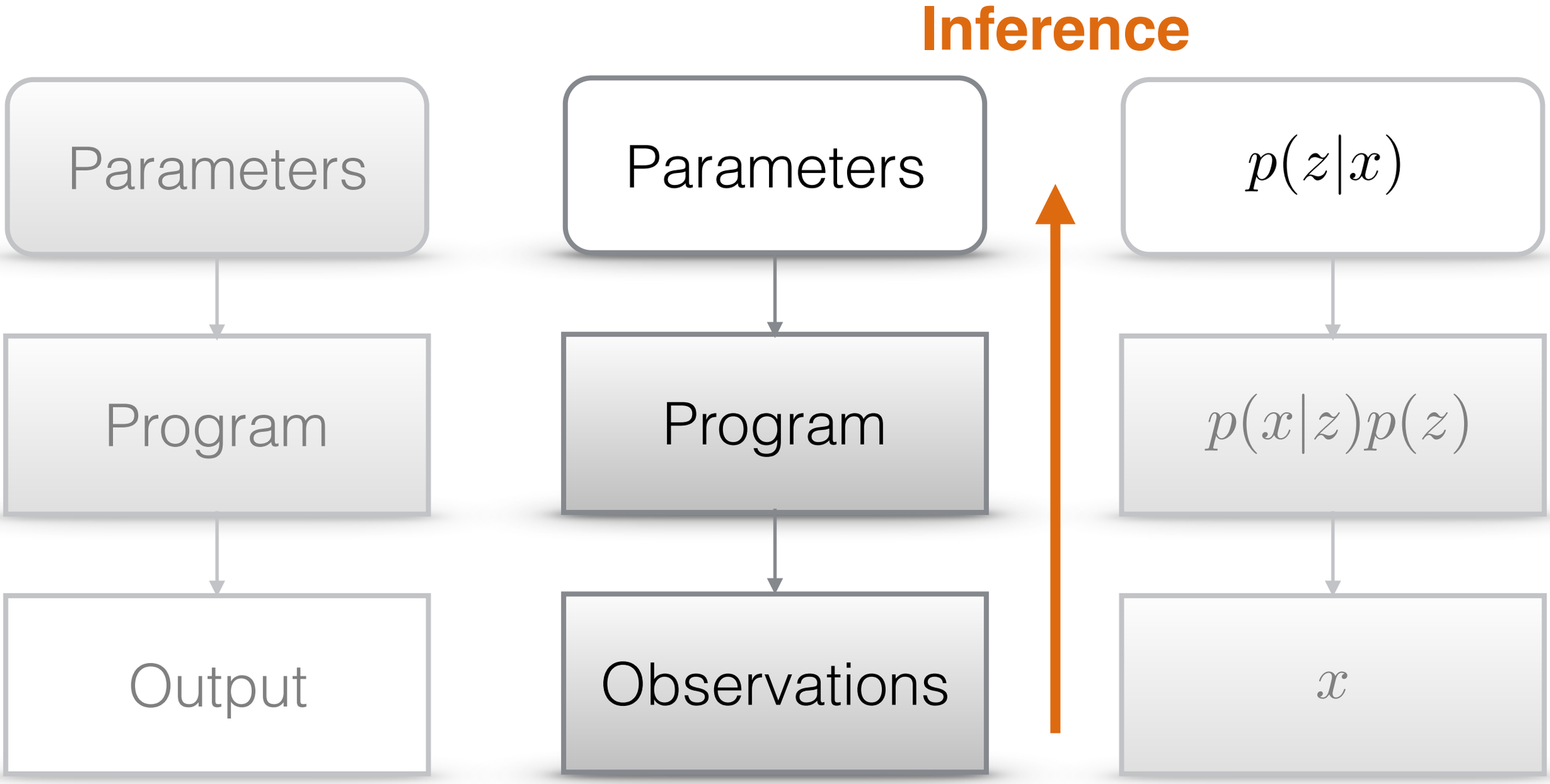
[slides, Frank Wood]

Intuition



[slides, Frank Wood]

Intuition



[slides, Frank Wood]

Probabilistic programming

Probabilistic models define a set of random variables and their relationships

- Observed variables
- ~~Unobserved (hidden, latent) variables~~ ***HEP: Monte Carlo truth***

Probabilistic programming extends this to *"ordinary programming with two added constructs"* (Gordon et al. 2014):

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

```
1: bool c1, c2;
2: c1 = Bernoulli(0.5);
3: c2 = Bernoulli(0.5);
4: return(c1, c2);
```

```
1: bool c1, c2;
2: c1 = Bernoulli(0.5);
3: c2 = Bernoulli(0.5);
4: observe(c1 || c2);
5: return(c1, c2);
```

7

[slide: Atılım Güneş Baydin]

UNIVERSAL PPLS

Quote from arXiv:1807.07706

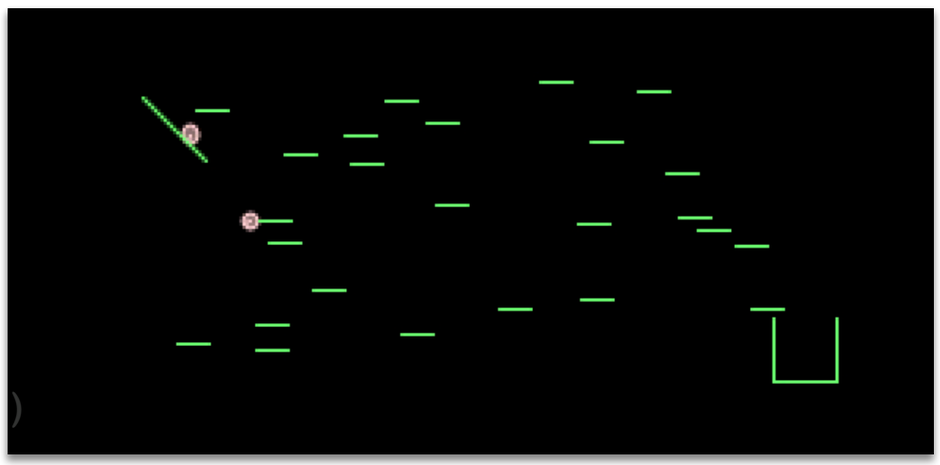
> Probabilistic programming languages (PPLs) attempt to decouple inference algorithms from model building, by creating a simple, yet expressive, syntax that allows one to take advantage of these powerful inference algorithms on any probabilistic generative model expressed as a regular computer program. *Universal* PPLs allow the expression of unrestricted probability models in a Turing-complete fashion [35–37], and there is a recent trend in combining these with variational inference and deep learning, leading to tools such as Pyro [38], ProbTorch [39], and Edward [40]. This is in contrast to languages such as Stan [41] that target the more restricted model class of probabilistic graphical models [26].

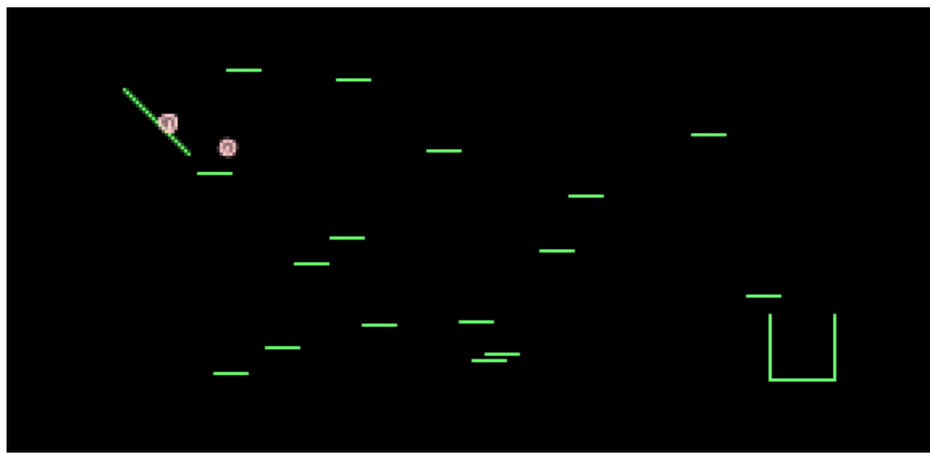
It is interesting to think of **non-standard interpretation** of the simulator code to **compile a non-standard output**

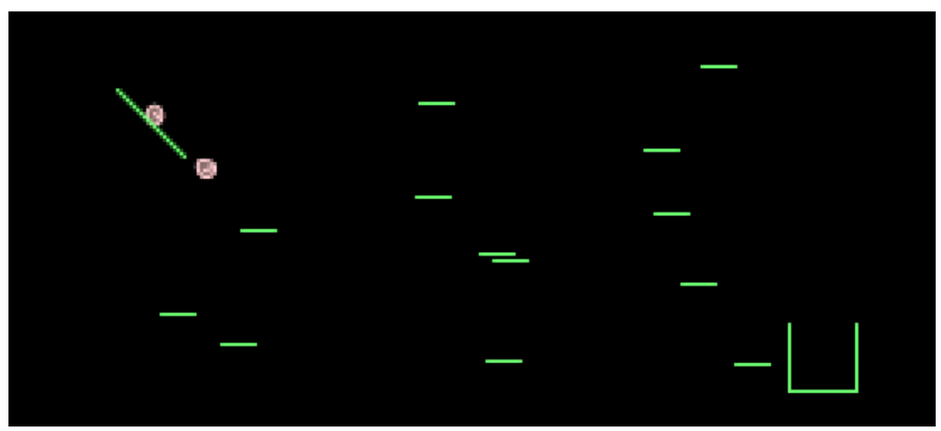
- eg. Output of compiler is a graphical model
- eg. Output of compiler is derivative of function
- eg. Output of compiler is joint score

PROBABILISTIC PROGRAMMING EXAMPLE

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)]

      {:balls balls
       :num-balls-in-box num-balls-in-box
       :bumper-positions bumper-positions}))
```





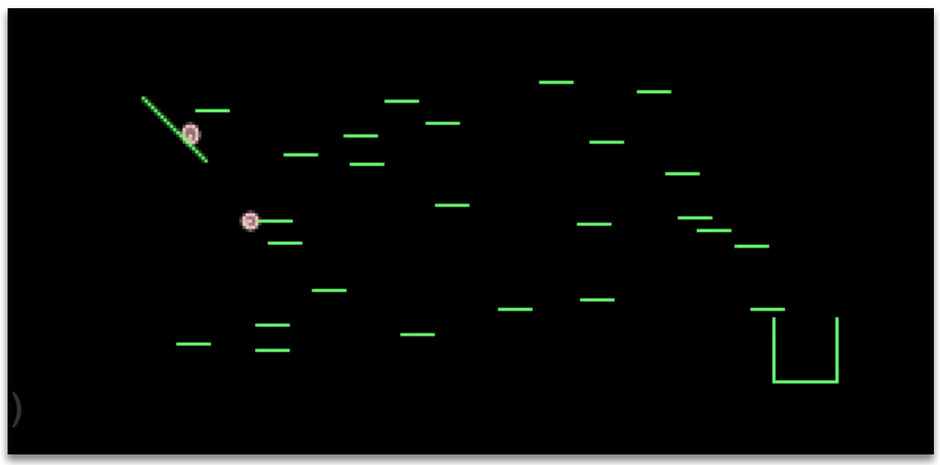


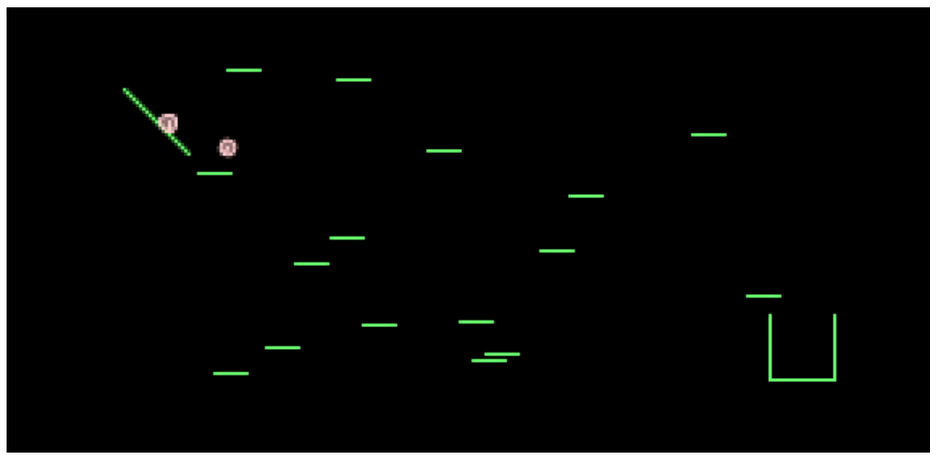
3 examples generated from simulator

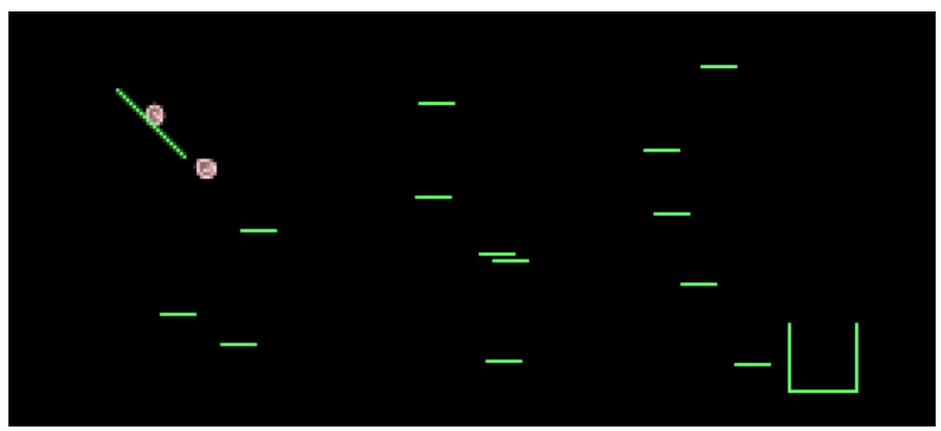
[slides, Frank Wood]

PROBABILISTIC PROGRAMMING EXAMPLE

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)]

      {:balls balls
       :num-balls-in-box num-balls-in-box
       :bumper-positions bumper-positions}))
```





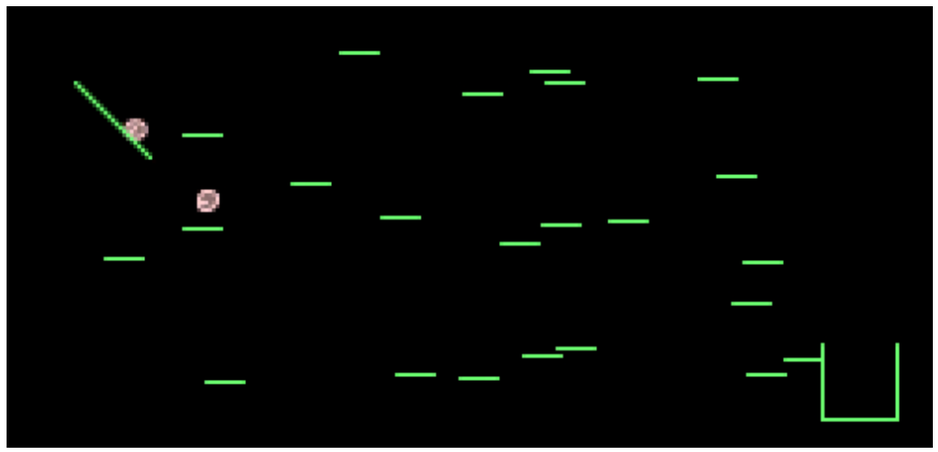


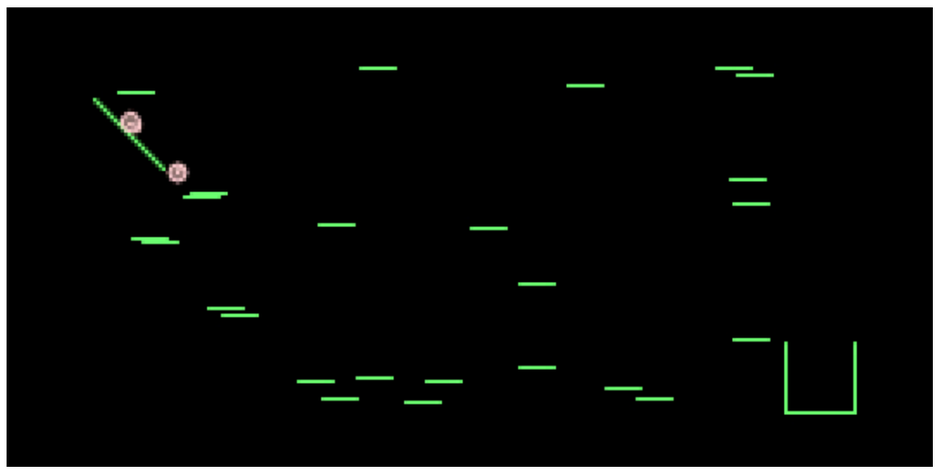
3 examples generated from simulator

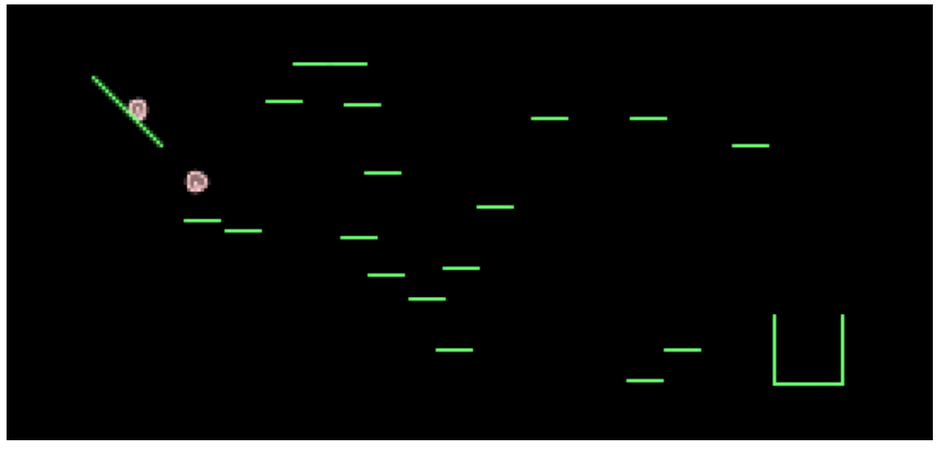
[slides, Frank Wood]

PROBABILISTIC PROGRAMMING EXAMPLE

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)

          obs-dist (normal 4 0.1)]

    (observe obs-dist num-balls-in-box)
```





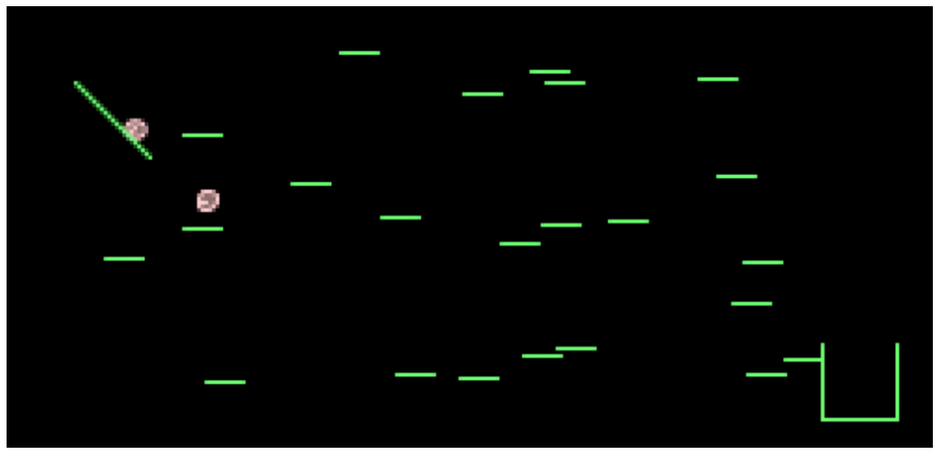


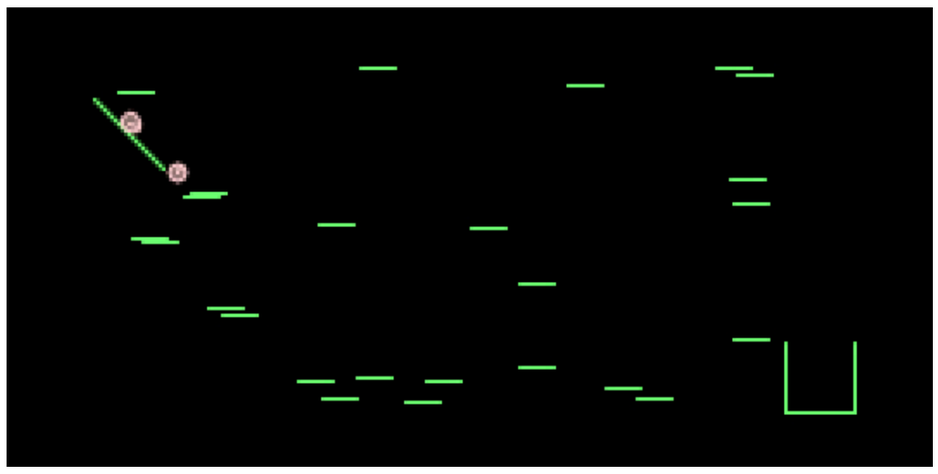
3 examples generated from simulator
conditioned on ~20% of balls land in box

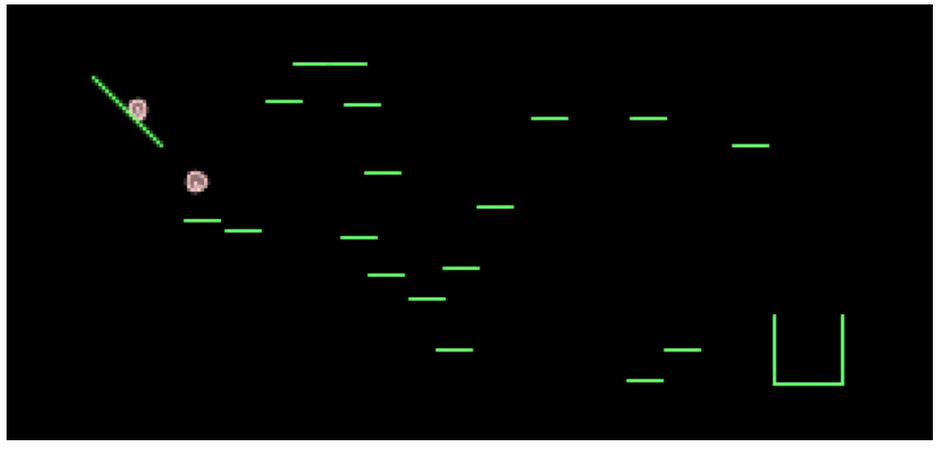
[slides, Frank Wood]

PROBABILISTIC PROGRAMMING EXAMPLE

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)

          obs-dist (normal 4 0.1)]

    (observe obs-dist num-balls-in-box)
```







3 examples generated from simulator
conditioned on ~20% of balls land in box

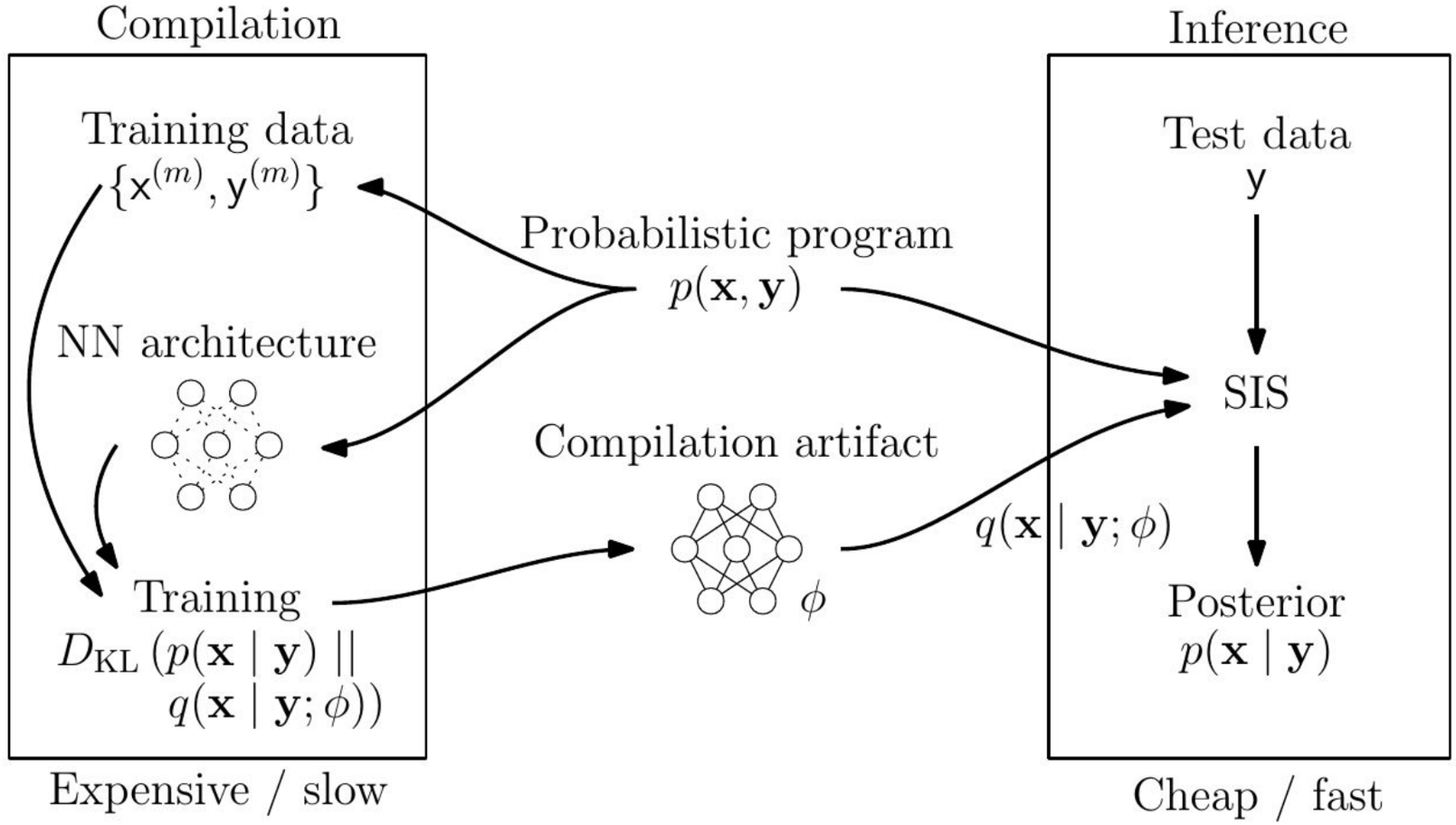
[slides, Frank Wood]

PROB PROG: HOW DOES IT WORK?

In short: hijack the random number generators and use NN's to perform a *very* smart type of importance sampling

Input: an inference problem denoted in a universal PPL (Anglican, CPProb)

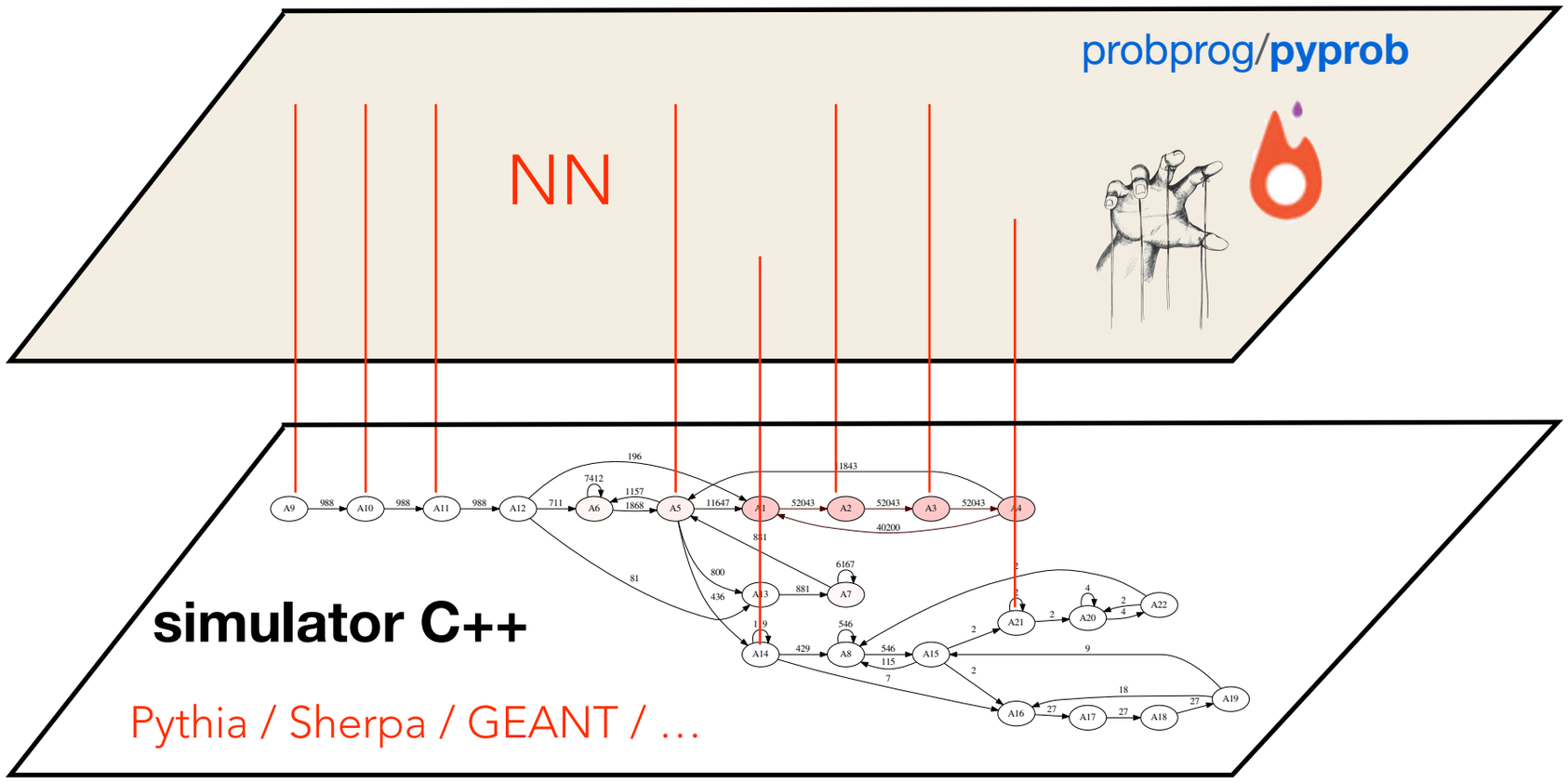
Output: a trained inference network, or "compilation artifact" (Torch, PyTorch)



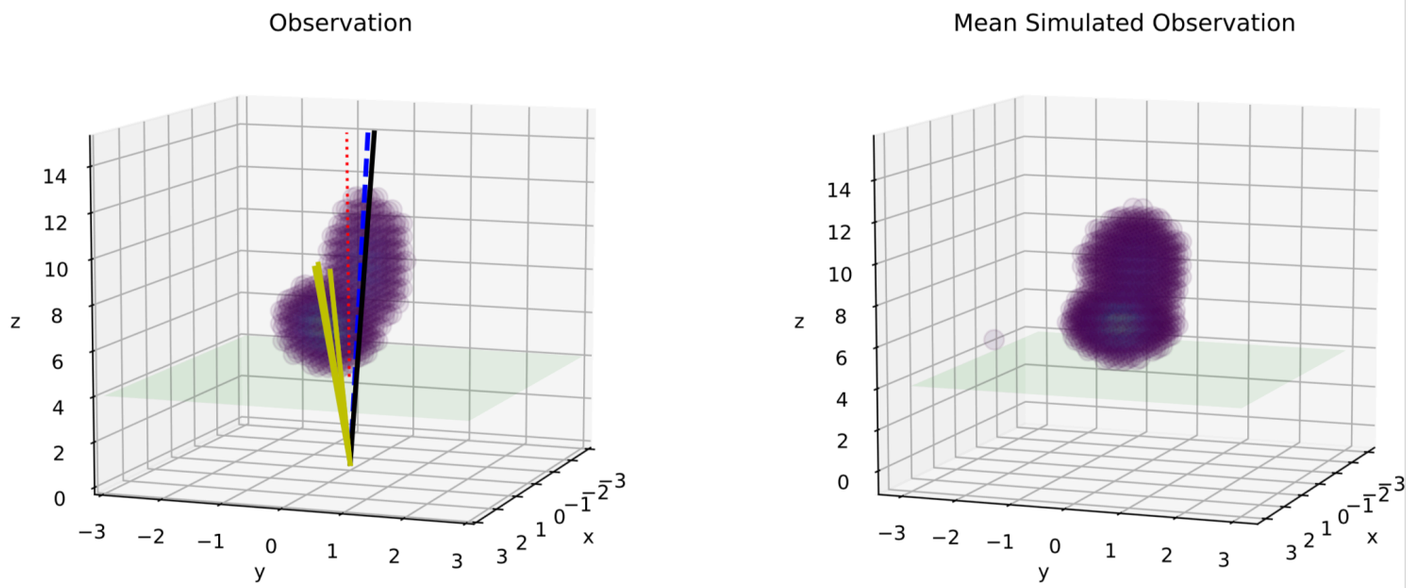
Le, Baydin and Wood. Inference Compilation and Universal Probabilistic Programming. AISTATS 2017. *arXiv:1610.09900*

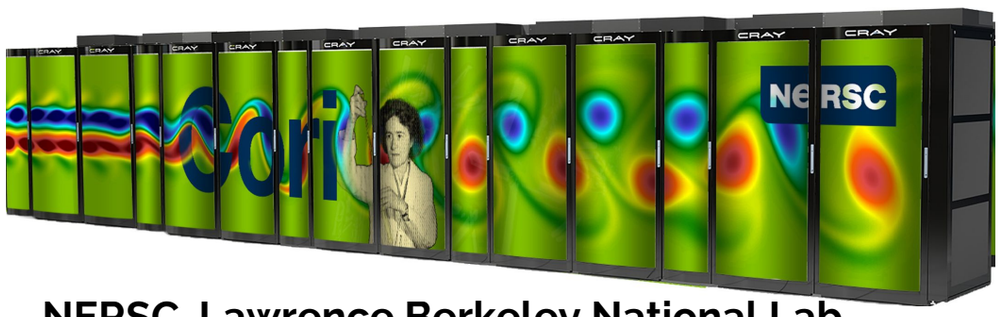
PROBABILISTIC PROGRAMING

Idea: hijack the random number generators and use Neural Network to perform a *very* fancy type of importance sampling



- Neural Network powered inference engine (python)
- real-world scientific simulator (C++)





NERSC, Lawrence Berkeley National Lab

arXiv:1807.07706

CODE

pyprob

https://github.com/probprog/pyprob

A PyTorch-based PPL

Inference engines:

- Markov chain Monte Carlo
  - Lightweight Metropolis Hastings (LMH)
  - Random-walk Metropolis Hastings (RMH)
- Importance Sampling
  - Regular (proposals from prior)
  - **Inference compilation (IC)**

Le, Baydin and Wood. Inference Compilation and Universal Probabilistic Programming. AISTATS 2017 *arXiv:1610.09900*.

PPX

https://github.com/probprog/ppx

Probabilistic **P**rogramming e**X**ecution protocol

- Cross-platform, via flatbuffers: http://google.github.io/flatbuffers/
- Supported languages: C++, C#, Go, Java, JavaScript, PHP, Python, TypeScript, Rust, Lua
- Similar to Open Neural Network Exchange (ONNX) for deep learning

Enables inference engines and simulators to be

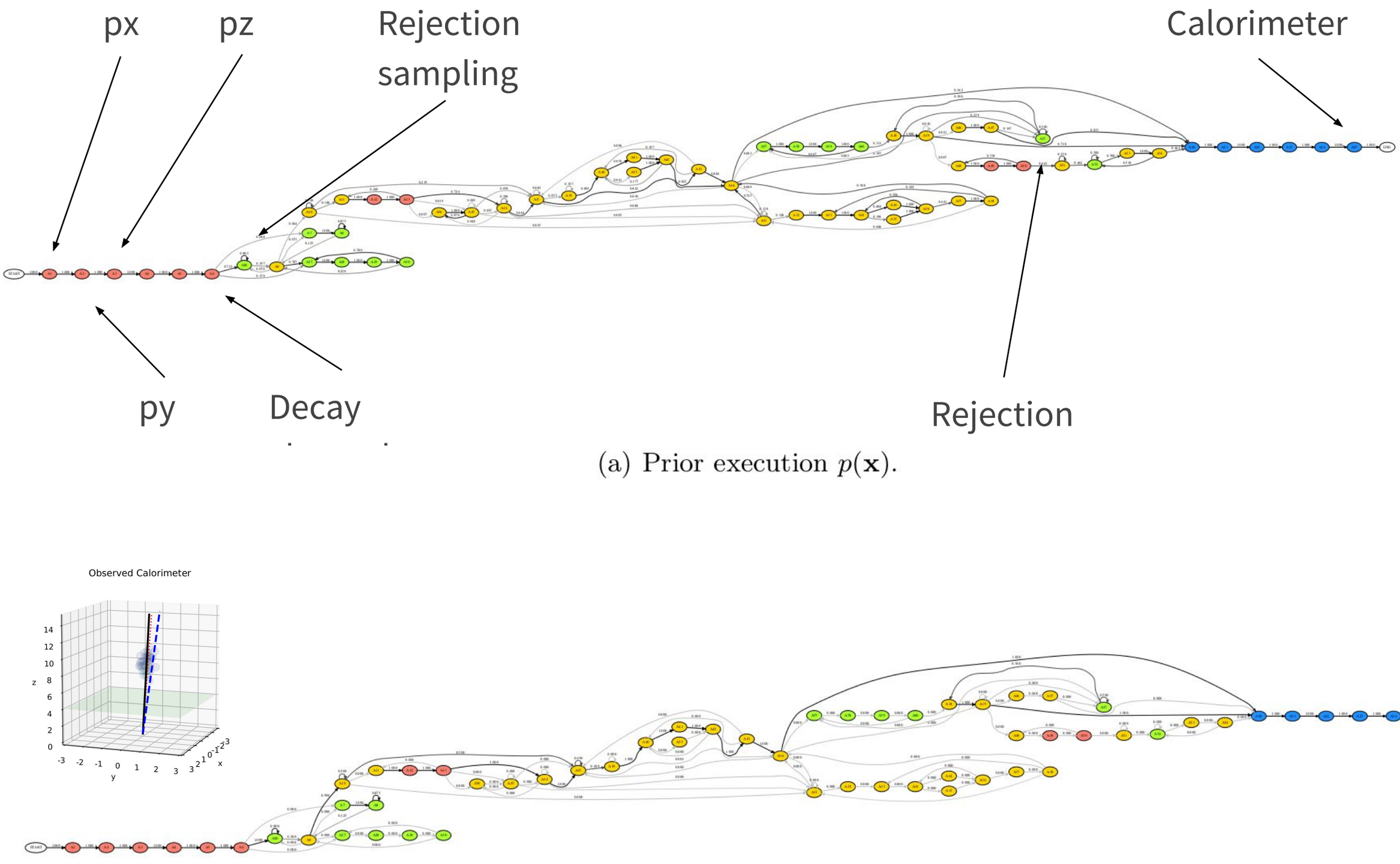
- implemented in different programming languages
- executed in separate processes, separate machines across networks

[slide: Atılım Güneş Baydin]

Interpretability

[slide: Atılım Güneş Baydin]

Latent probabilistic structure of 250 most frequent trace types

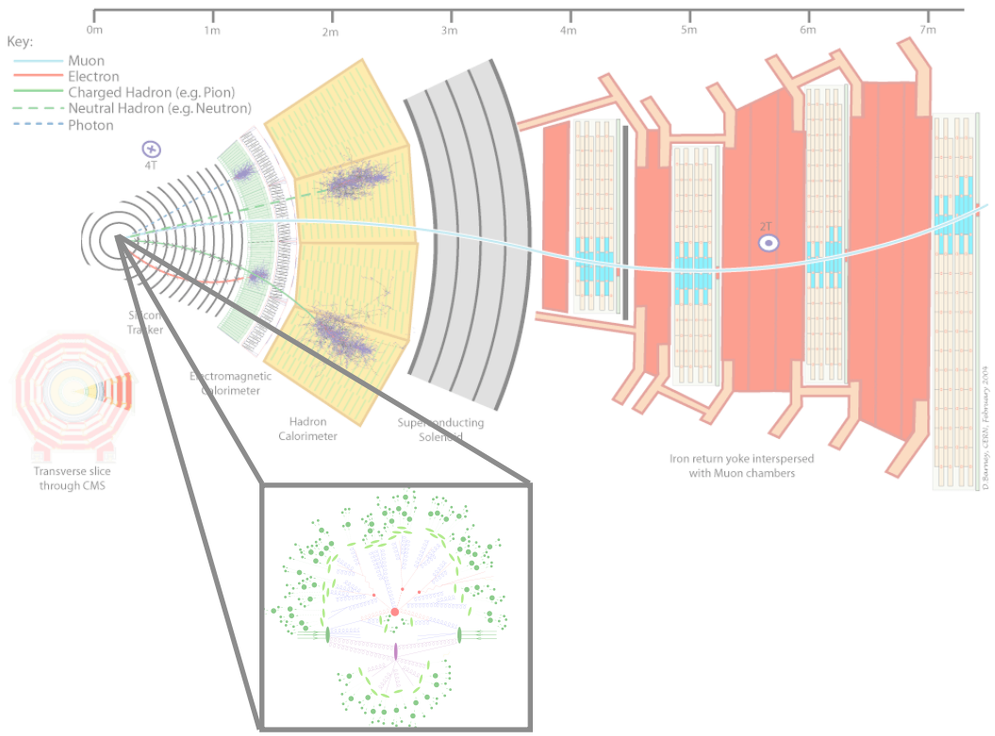


(a) Prior execution $p(\mathbf{x})$.

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

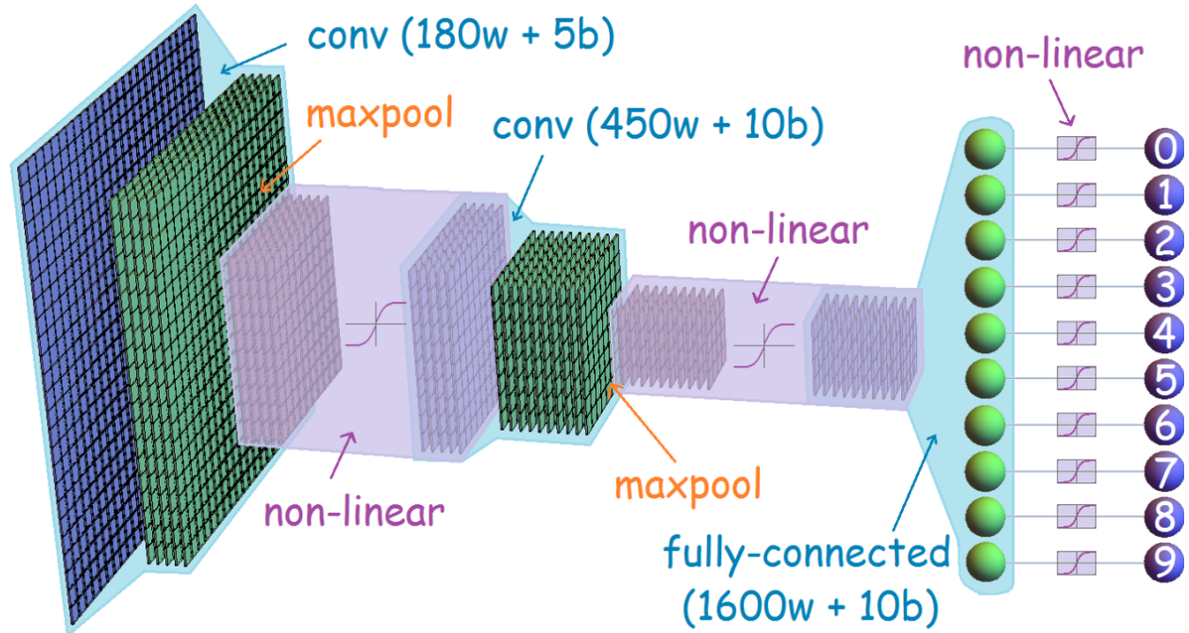
TWO APPROACHES TO LIKELIHOOD FREE INFERENCE

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator
(with deep learning)



- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

GENERATIVE MODEL FOR IMAGES

Note, same NN can model birds, ants, and volcanos! Is that good or bad?

How an A.I. 'Cat-and-Mouse Game' Generates Believable Fake Photos

By CADE METZ and KEITH COLLINS JAN. 2, 2018

This one is computer-generated

This one is also computer-generated

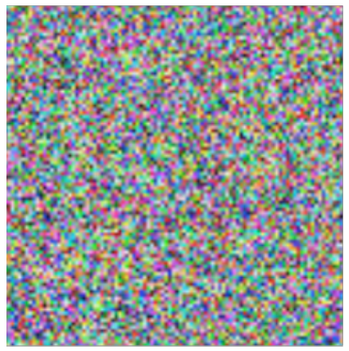
LEARNING THE SIMULATOR

note, same NN can model birds, ants, and volcanos! (lack of implicit bias)

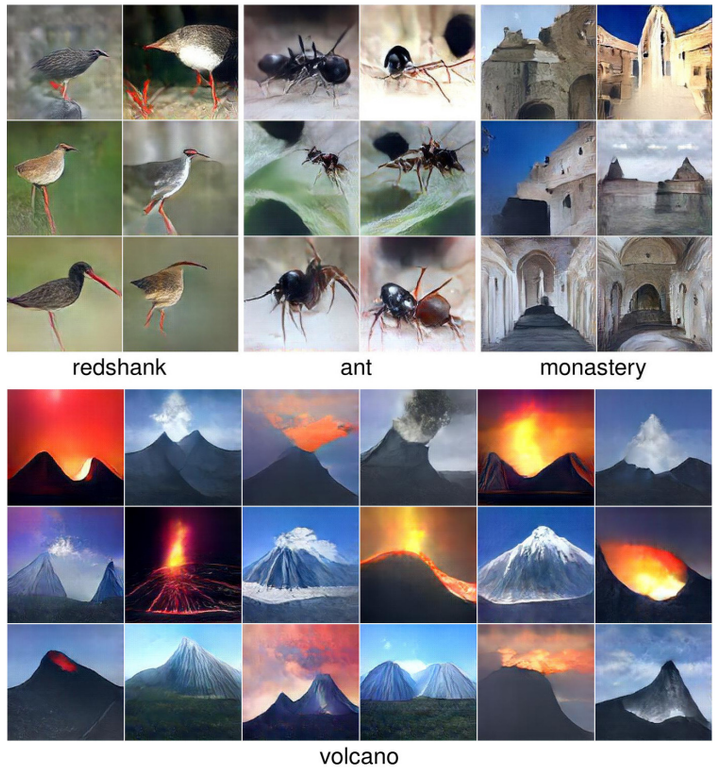
Z

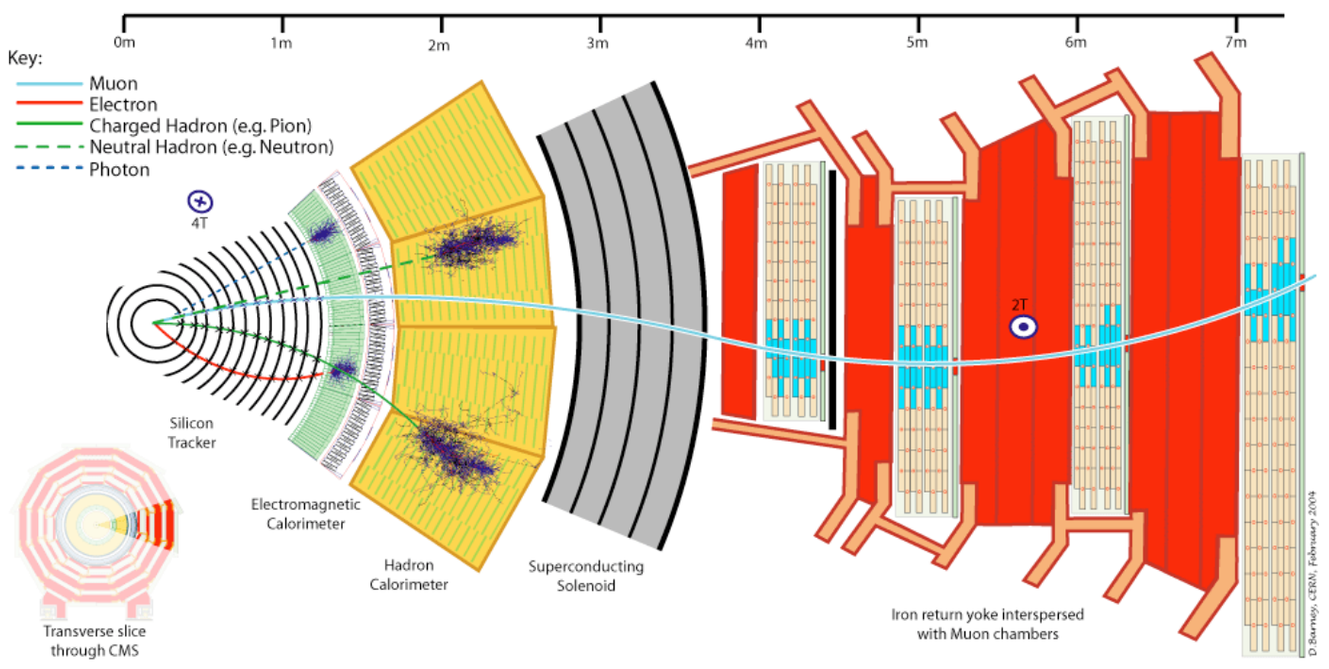
X

Noise ~ N(0,1)



Generative Model





http://torch.ch/blog/2015/11/13/gan.html

Paganini, et. al. Phys.Rev.Lett. 120 (2018) no.4, 042003 ; & ATL-SOFT-PUB-2018-001

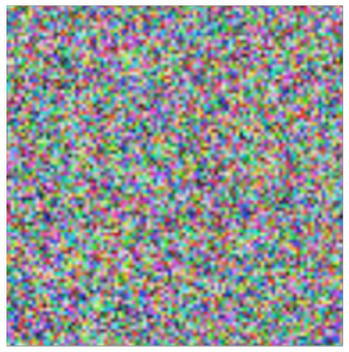
LEARNING THE SIMULATOR

Z

X

note, same NN can model birds, ants, and volcanos! (lack of implicit bias)

Noise ~ N(0,1)



Generative Model

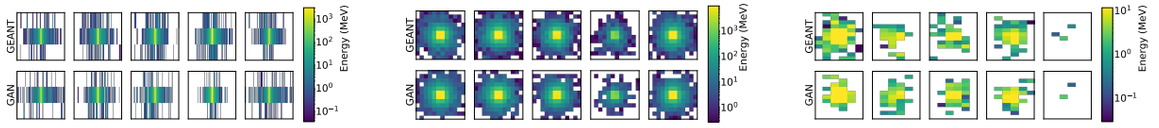


Figure 9: Five randomly selected e^+ showers per calorimeter layer from the training set (top) and the five nearest neighbors (by euclidean distance) from a set of CaloGAN candidates.

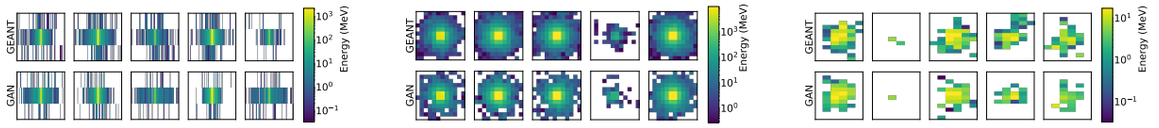


Figure 10: Five randomly selected γ showers per calorimeter layer from the training set (top) and the five nearest neighbors (by euclidean distance) from a set of CaloGAN candidates.

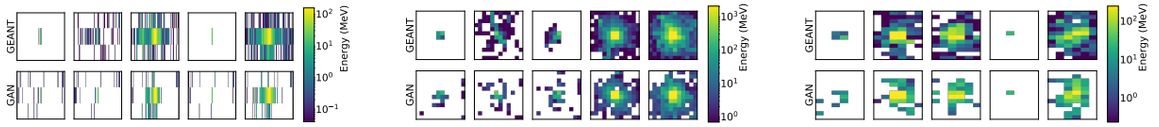
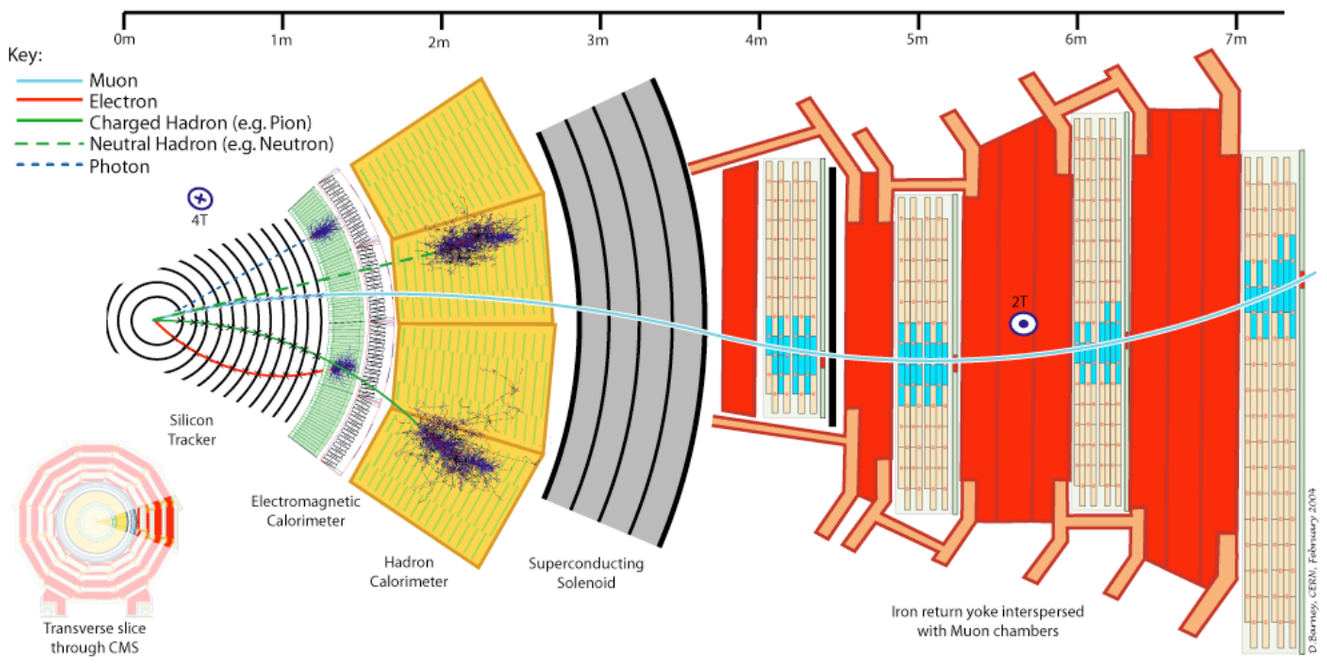


Figure 11: Five randomly selected π^+ showers per calorimeter layer from the training set (top) and the five nearest neighbors (by euclidean distance) from a set of CaloGAN candidates.



http://torch.ch/blog/2015/11/13/gan.html

Paganini, et. al. Phys.Rev.Lett. 120 (2018) no.4, 042003 ; & ATL-SOFT-PUB-2018-001

NEW! AVO

G. Louppe, Joeri Hermans, & K.C. arXiv:1707.07113
AISTATS 2019

Adversarial Variational Optimization of Non-Differentiable Simulators

Gilles Louppe University of Liège
Joeri Hermans University of Liège
Kyle Cranmer New York University

Complex computer simulators are increasingly used across fields of science as generative models tying parameters of an underlying theory to experimental observations. Inference in this setup is often difficult, as simulators rarely admit a tractable density or likelihood function. We introduce Adversarial Variational Optimization (AVO), a likelihood-free inference algorithm for fitting a non-differentiable generative model incorporating ideas from empirical Bayes and variational inference. We adapt the training procedure of generative adversarial networks by replacing the differentiable generative network with a domain-specific simulator. We solve the resulting non-differentiable minimax problem by minimizing variational upper bounds of the two adversarial objectives. Effectively, the procedure results in learning a proposal distribution over simulator parameters, such that the corresponding marginal distribution of the generated data matches the observations. We present results of the method with simulators producing both discrete and continuous data.

Leo is *G* Tom is *D*

Similar to GAN setup, but instead of using a neural network as the generator, use the actual simulation (eg. Pythia, GEANT)

Continue to use a neural network discriminator / critic.

Difficulty: the simulator isn't differentiable, but there's a **trick**!

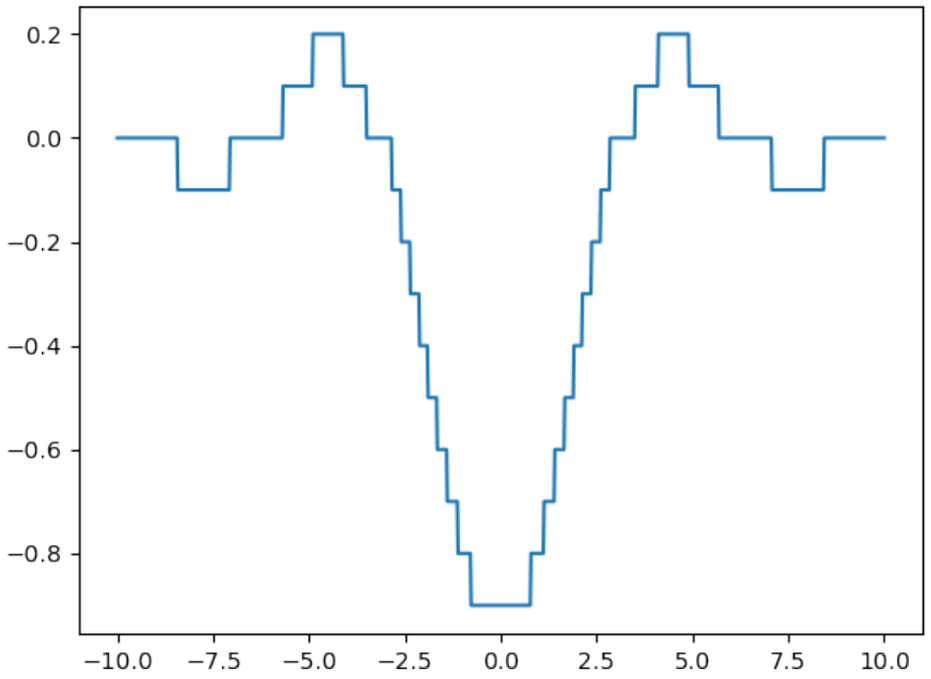
Allows us to efficiently fit / **tune simulation** with stochastic gradient techniques!

VARIATIONAL OPTIMIZATION

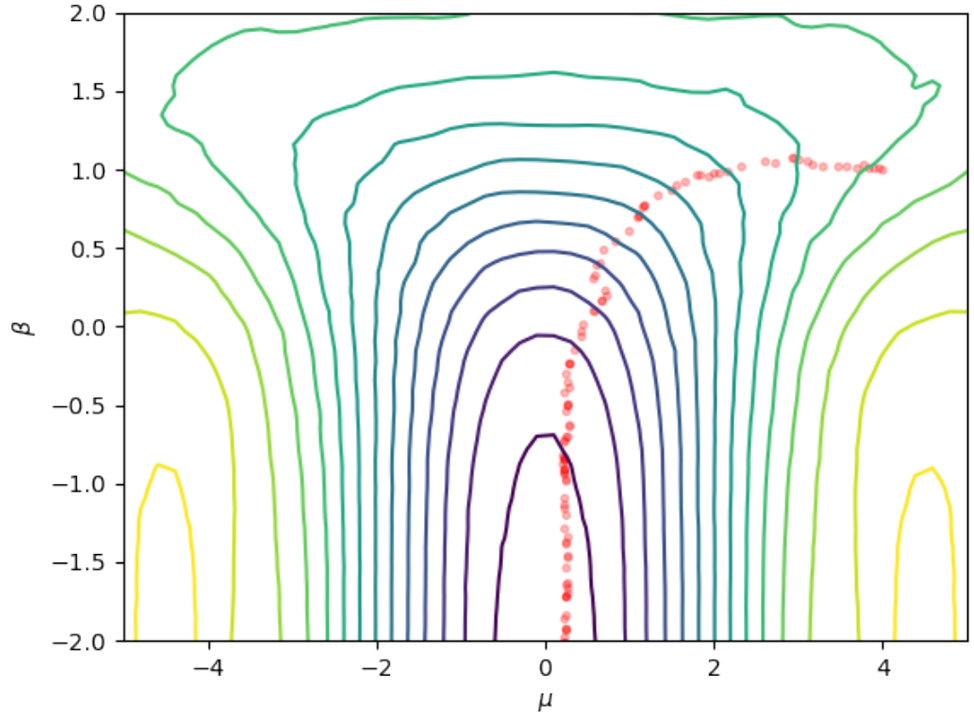
G. Louppe, Joeri Hermans, & K.C. arXiv:1707.07113
AISTATS 2019

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \leq \mathbb{E}_{\boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\boldsymbol{\psi})}[f(\boldsymbol{\theta})] = U(\boldsymbol{\psi})$$

$$\nabla_{\boldsymbol{\psi}} U(\boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\boldsymbol{\psi})}[f(\boldsymbol{\theta}) \nabla_{\boldsymbol{\psi}} \log q(\boldsymbol{\theta}|\boldsymbol{\psi})]$$



Piecewise constant $-\frac{\sin(\mathbf{x})}{\mathbf{x}}$



$$q(\boldsymbol{\theta}|\boldsymbol{\psi} = (\mu, \beta)) = \mathcal{N}(\mu, e^{\beta})$$

Recurrent Inference Machines

RECURRENT INFERENCE MACHINES

arXiv:1706.04008

Recurrent Inference Machines for Solving Inverse Problems

Patrick Putzky & Max Welling
Informatics Institute
University of Amsterdam
{pputzky,m.welling}@uva.nl

The Usual Approach

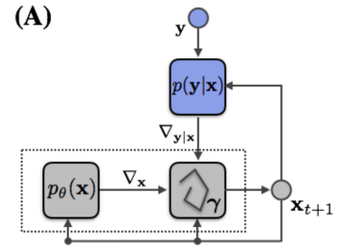
$$L(X) = \log P_A(Y|X) + \log P_\theta(X)$$
$$Y = A \cdot X + \eta$$

observations — generative model (known) — prior (learn)

$$X_{t+1} = X_t + \alpha_t(\nabla_X \log P_A(Y|X_t) + \nabla_X \log P_\theta(X_t))$$

advantage: model P(X) and optimization are separated.

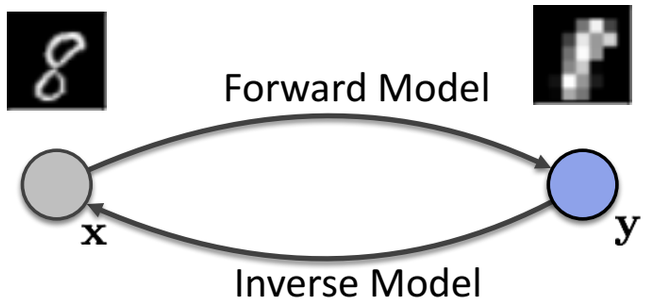
disadvantage: accuracy suffers because model and optimization interact…



Inverse Problems

w/ Patrick Putzky

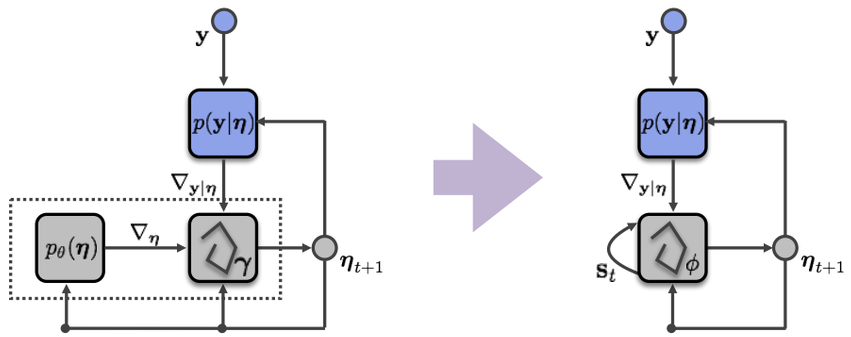
Quantity of interest — Measurement



Forward Model $\mathbf{y} = g(\mathbf{x}) + n$

Inverse Model $\hat{\mathbf{x}} = h(\mathbf{y})$

Learning Inference: Recurrent Inference Machine



$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \gamma_t(\nabla_{\mathbf{y}|\boldsymbol{\eta}} + \nabla_{\boldsymbol{\eta}})$$

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + h_\phi(\nabla_{\mathbf{y}|\boldsymbol{\eta}}, \boldsymbol{\eta}_t, \mathbf{s}_t)$$

- Abstract and parameterize computation graph into RNN
- Integrate prior P(X) in RNN
- Add memory state s
- Meta learn the parameters of the RNN

arXiv:1901.01359

DATA-DRIVEN RECONSTRUCTION OF GRAVITATIONALLY LENSED GALAXIES USING RECURRENT INFERENCE MACHINES

WARREN R. MORNINGSTAR[1], LAURENCE PERREAULT LEVASSEUR[3], YASHAR D. HEZAVEH[3], ROGER BLANDFORD[1], PHIL MARSHALL[1], PATRICK PUTZKY[4], THOMAS D. RUETER[2], RISA WECHSLER[1], AND MAX WELLING[4]

Draft version January 8, 2019

ABSTRACT

We present a machine learning method for the reconstruction of the undistorted images of background sources in strongly lensed systems. This method treats the source as a pixelated image and utilizes the Recurrent Inference Machine (RIM) to iteratively reconstruct the background source given a lens model. Our architecture learns to minimize the likelihood of the model parameters (source pixels) given the data using the physical forward model (ray tracing simulations) while implicitly learning the prior of the source structure from the training data. This results in better performance compared to linear inversion methods, where the prior information is limited to the 2-point covariance of the source pixels approximated with a Gaussian form, and often specified in a relatively arbitrary manner. We combine our source reconstruction network with a convolutional neural network that predicts the parameters of the mass distribution in the lensing galaxies directly from telescope images, allowing a fully automated reconstruction of the background source images and the foreground mass distribution.

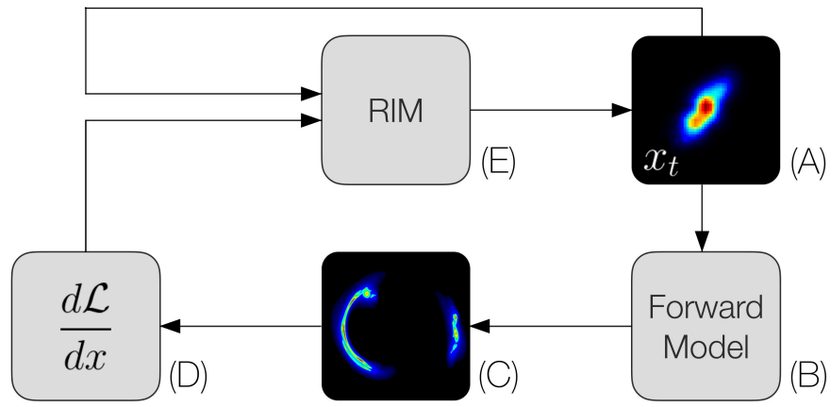
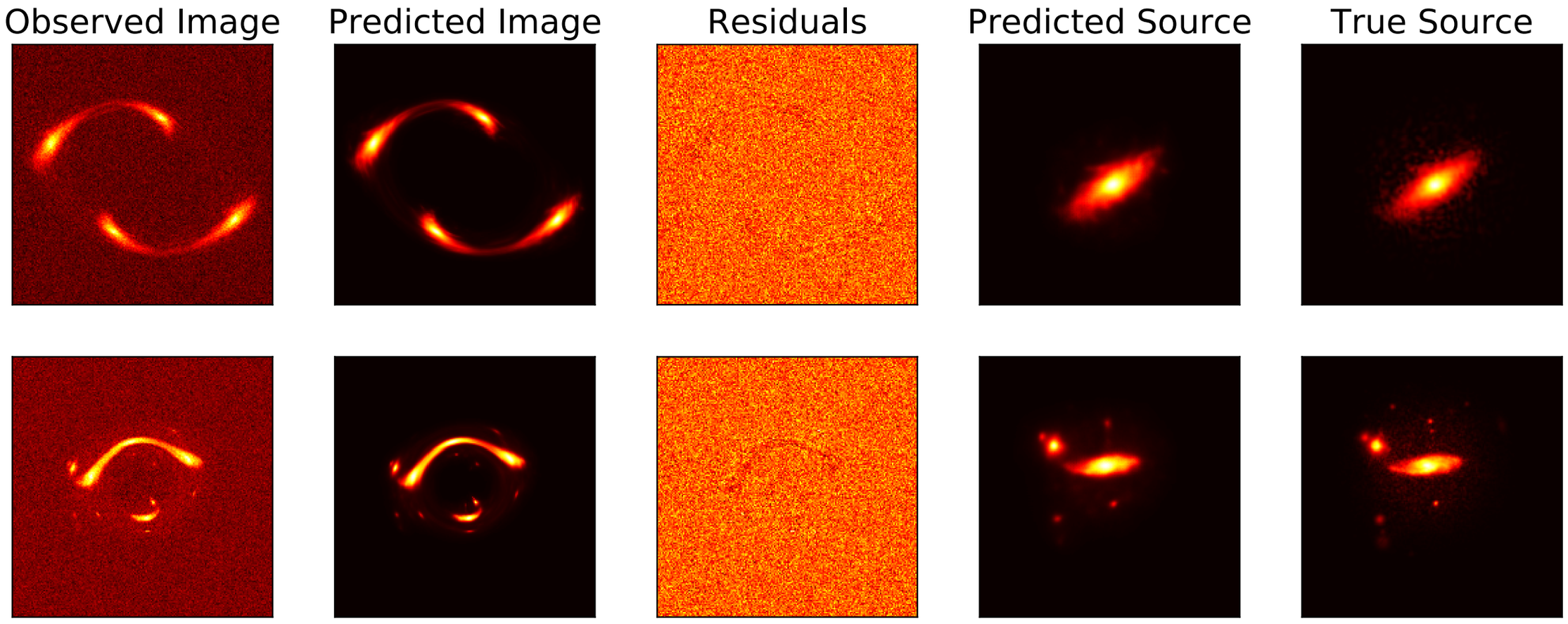


Figure 1. A diagram of the structure of the reconstruction. A proposed image of the source (A) is lensed using the forward model (B, a ray-tracing simulation) to produce the lensed arcs (C, model data). The likelihood is computed by comparing the model with data. The derivative of the likelihood with respect to the values of the source pixels is calculated (D). The derivative of the likelihood and the current estimate (x_t) are given to the RIM (E) to produce a new estimate for the source (x_{t+1}).



Recurrent machines for likelihood-free inference

Arthur Pesah[*]
KTH Royal Institute of Technology
Stockholm, Sweden

Antoine Wehenkel[*]
University of Liège
Liège, Belgium

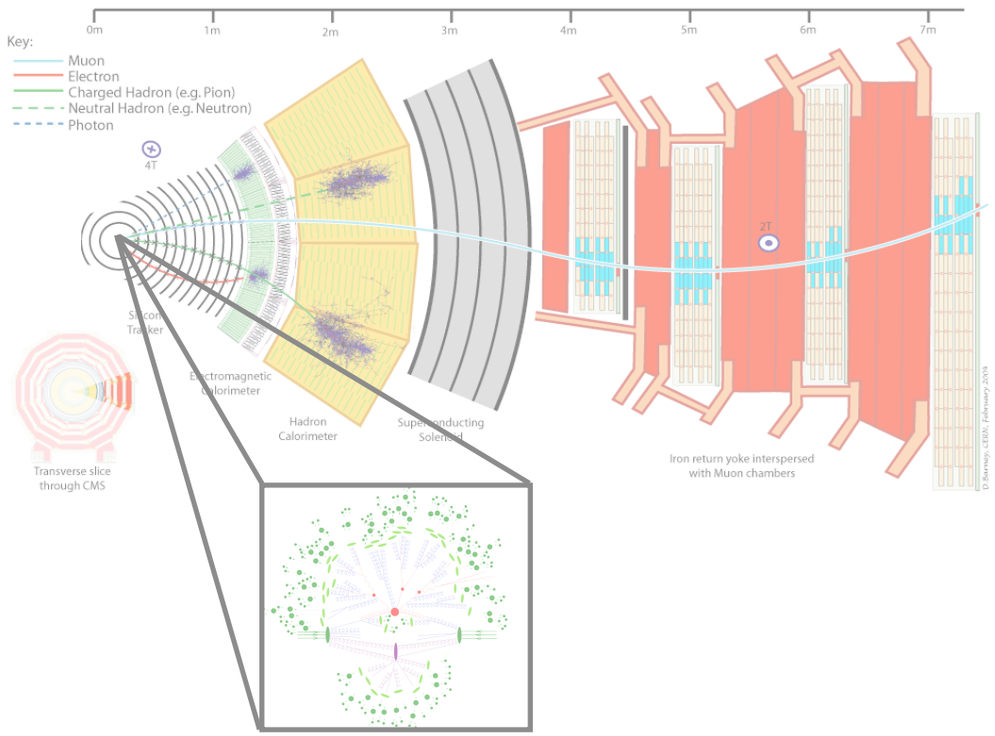
Gilles Louppe
University of Liège
Liège, Belgium

Abstract

Likelihood-free inference is concerned with the estimation of the parameters of a non-differentiable stochastic simulator that best reproduce real observations. In the absence of a likelihood function, most of the existing inference methods optimize the simulator parameters through a handcrafted iterative procedure that tries to make the simulated data more similar to the observations. In this work, we explore whether meta-learning can be used in the likelihood-free context, for learning automatically from data an iterative optimization procedure that would solve likelihood-free inference problems. We design a recurrent inference machine that learns a sequence of parameter updates leading to good parameter estimates, without ever specifying some explicit notion of divergence between the simulated data and the real data distributions. We demonstrate our approach on toy simulators, showing promising results both in terms of performance and robustness.

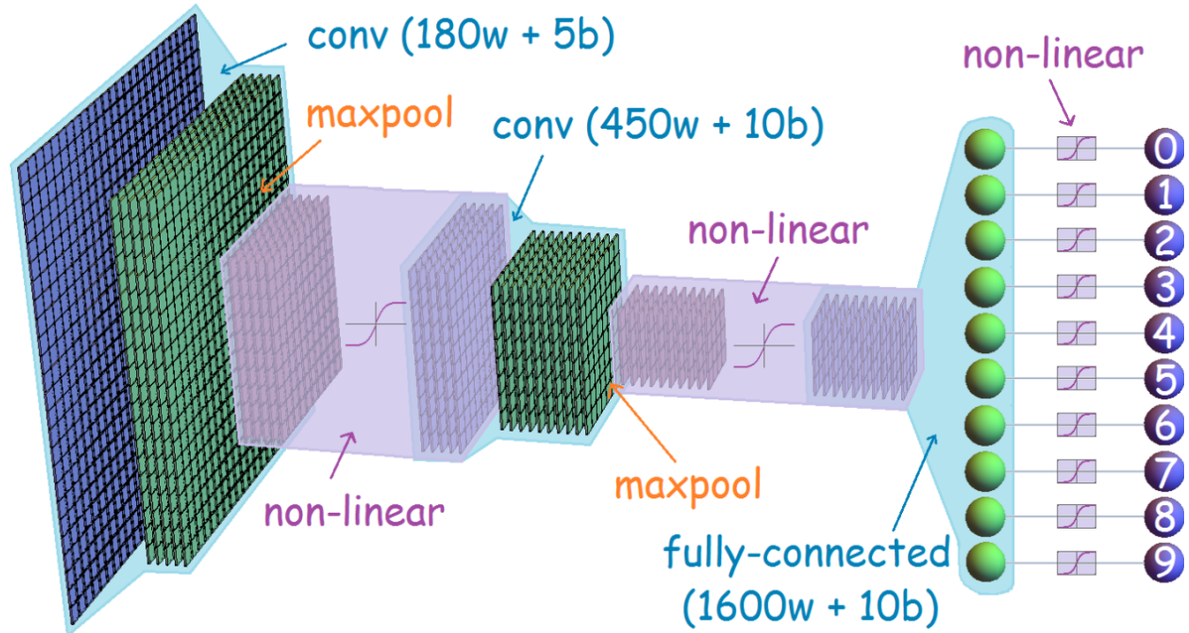
TWO APPROACHES TO LIKELIHOOD FREE INFERENCE

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator
(with deep learning)

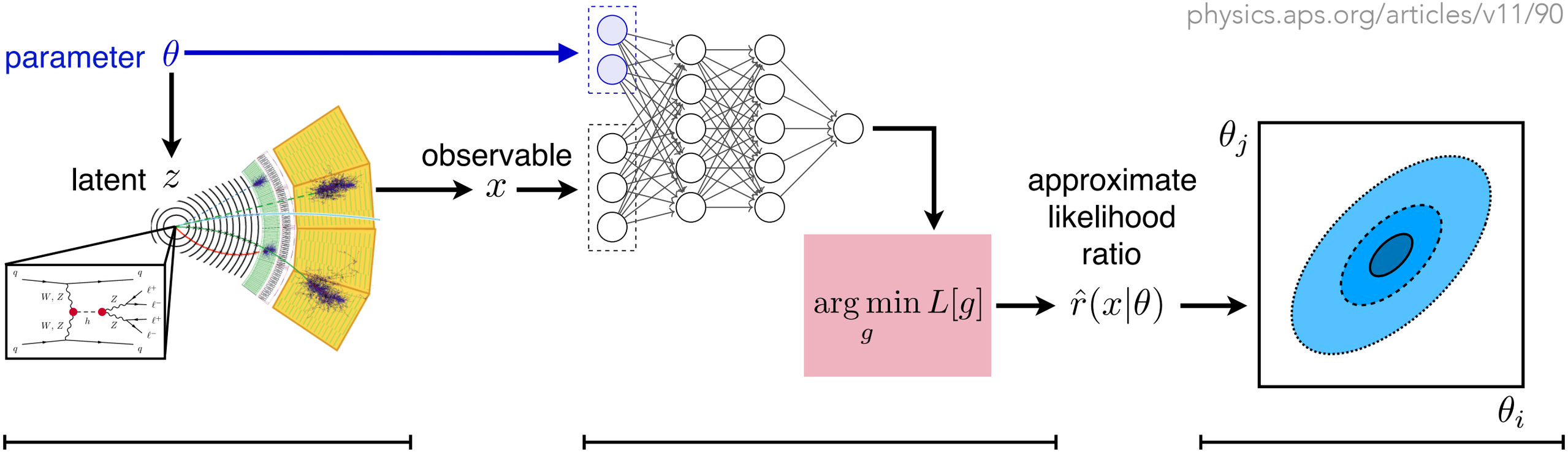


- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

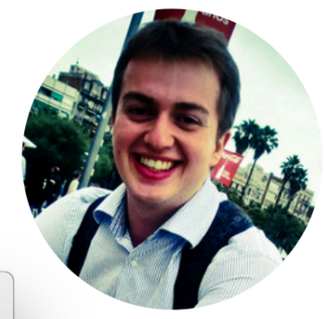
LEARNING THE LIKELIHOOD RATIO

arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
physics.aps.org/articles/v11/90



CARL SOFTWARE

http://diana-hep.org/carl/



Gilles Louppe

diana-hep.org

DiscoveryLinks | Higgs | RooStats | ALEPH | Apple | News | Life Stuff | ATLAS | Wikipedia, | inSpire | Theory&Practice | nyu espace | JCSS | HCG

Meet | Jupyter Note... | Weekend rea... | early-career-... | 2016 Electio... | 12-day Event... | Joint meetin... | carl API doc...

Fork me on GitHub

Index

Sub-modules

- `carl.data`
- `carl.distributions`
- `carl.learning`
- `carl.ratios`

Notebooks

- Composing and fitting distributions
- Diagnostics for approximate likelihood ratios
- Likelihood ratios of mixtures of normals
- Parameterized inference from multidimensional data
- Parameterized inference with nuisance parameters

Display a menu

`carl` module

`carl` is a toolbox for likelihood-free inference in Python.

The likelihood function is the central object that summarizes the information from an experiment needed for inference of model parameters. It is key to many areas of science that report the results of classical hypothesis tests or confidence intervals using the (generalized or profile) likelihood ratio as a test statistic. At the same time, with the advance of computing technology, it has become increasingly common that a simulator (or generative model) is used to describe complex processes that tie parameters of an underlying theory and measurement apparatus to high-dimensional observations. However, directly evaluating the likelihood function in these cases is often impossible or is computationally impractical.

In this context, the goal of this package is to provide tools for the likelihood-free setup, including likelihood (or density) ratio estimation algorithms, along with helpers to carry out inference on top of these.

This project is still in its early stage of development. Join us on GitHub if you feel like contributing!

build passing | coverage 91% | DOI 10.5281/zenodo.47798 | JOSS 10.21105/joss.00011

Likelihood-free inference with calibrated classifiers

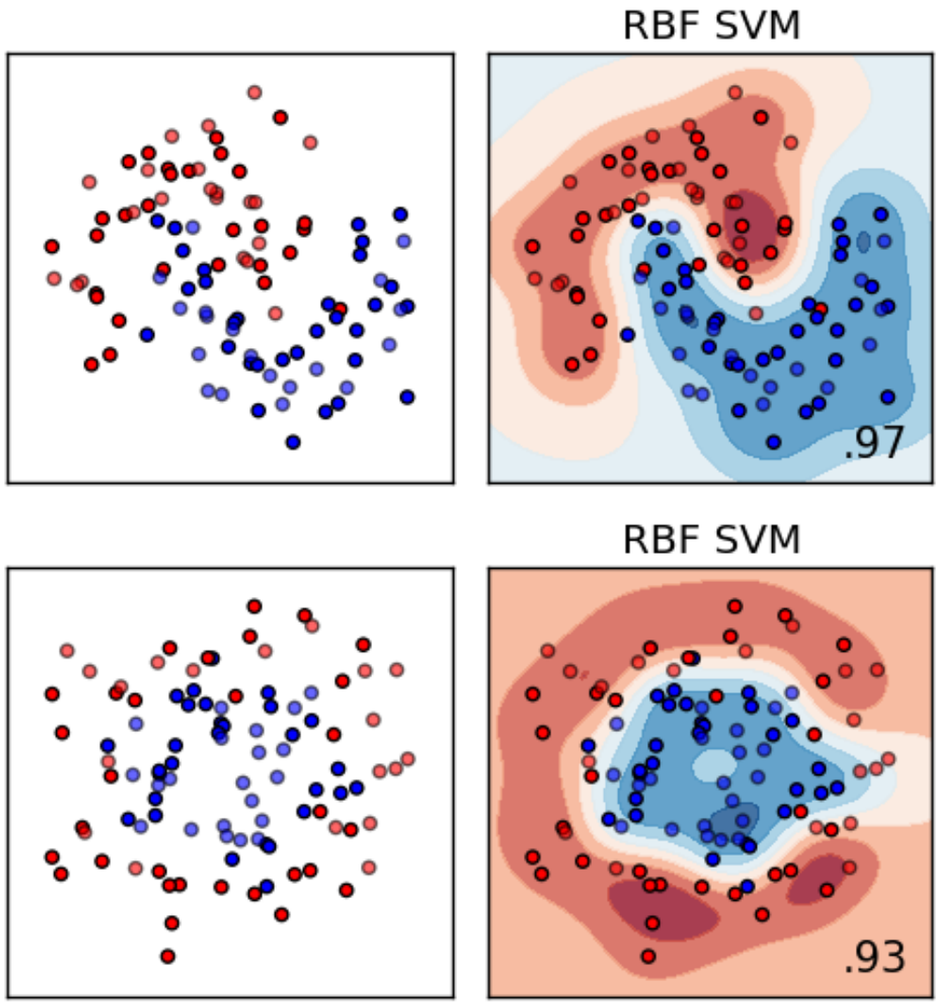
Extensive details regarding likelihood-free inference with calibrated classifiers can be found in the companion paper *"Approximating Likelihood Ratios with Calibrated Discriminative Classifiers", Kyle Cranmer, Juan Pavez, Gilles Louppe.* http://arxiv.org/abs/1506.02169

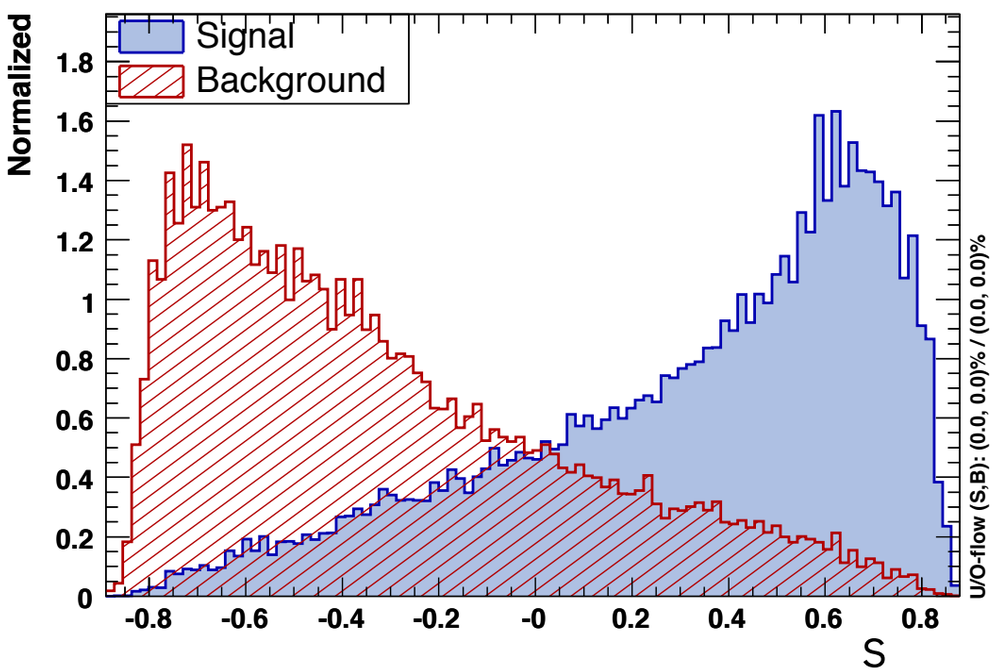
Installation

The following dependencies are required:

- Numpy >= 1.11

LIKELIHOOD RATIO TRICK





- **binary classifier**: find function s(x) that minimizes ***loss***:

$$L[s] = \int p(x|H_0)\,(0 - s(x))^2\,dx + \int p(x|H_1)\,(1 - s(x))^2 dx$$

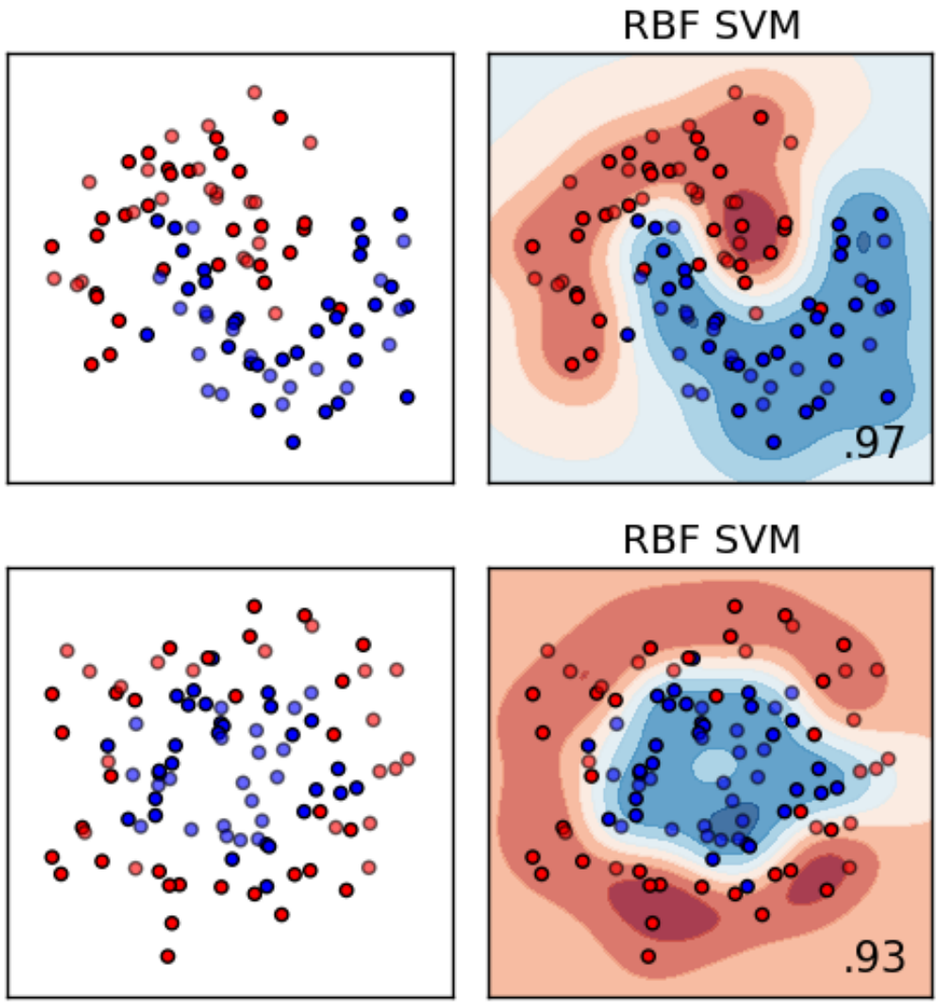
- i.e. approximate the optimal classifier

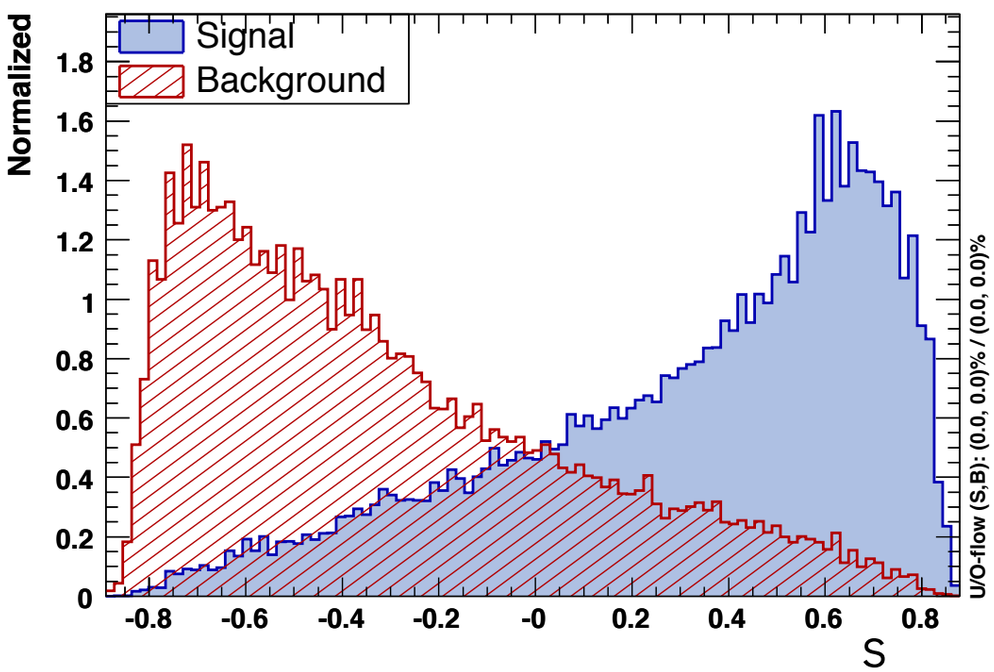
$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

- which is 1-to-1 with the likelihood ratio

$$\frac{p(x|H_1)}{p(x|H_0)}$$

LIKELIHOOD RATIO TRICK





- **binary classifier**: find function s(x) that minimizes ***loss***:

$$L[s] = \int p(x|H_0)\,(0 - s(x))^2\,dx + \int p(x|H_1)\,(1 - s(x))^2 dx \approx \frac{1}{N}\sum_{i=1}^{N}(y_i - s(x_i))^2$$

- i.e. approximate the optimal classifier

$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

- which is 1-to-1 with the likelihood ratio

$$\frac{p(x|H_1)}{p(x|H_0)}$$

LIKELIHOOD RATIO TRICK

- **Note:**
  the cross-entropy loss

$$L[\hat{s}(x)] = -\int \mathrm{d}x \left[p(x|y=1)\ \log(\hat{s}(x)) + p(x|y=0)\ \log(1-\hat{s}(x))\right]$$

$$-\frac{1}{N}\sum_{(x_i,y_i)}\left[y_i \log(\hat{s}(x_i)) + (1-y_i)\log(1-\hat{s}(x_i))\right].$$

  is also minimized by the same s(x)

  less variance / better results

- **binary classifier**: find function s(x) that minimizes ***loss***:

$$L[s] = \int p(x|H_0)\,(0-s(x))^2\,dx + \int p(x|H_1)\,(1-s(x))^2 dx$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}(y_i - s(x_i))^2$$

- i.e. approximate the optimal classifier

$$s(x) = \frac{p(x|H_1)}{p(x|H_0)+p(x|H_1)}$$

- which is 1-to-1 with the likelihood ratio

$$\frac{p(x|H_1)}{p(x|H_0)}$$

s

EXTENDING THE LIKELIHOOD RATIO TRICK

A binary classifier approximates

$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

Which is one-to-one with the likelihood ratio

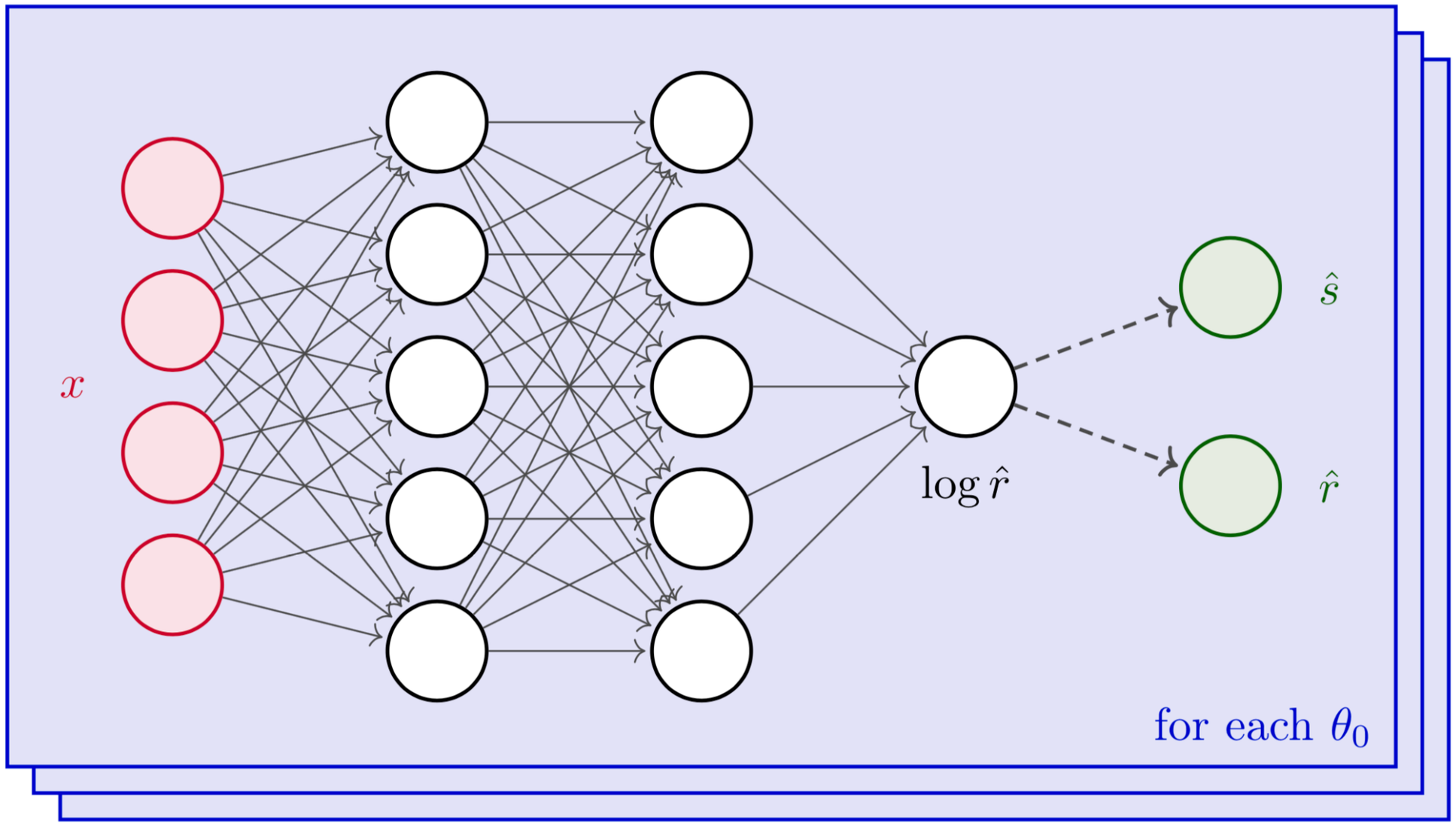
$$\frac{p(x|H_1)}{p(x|H_0)} = 1 - \frac{1}{s(x)}$$

Can do the same thing for any two points θ_0 & θ_1 in parameter space Θ. I call this a **parametrized classifier**

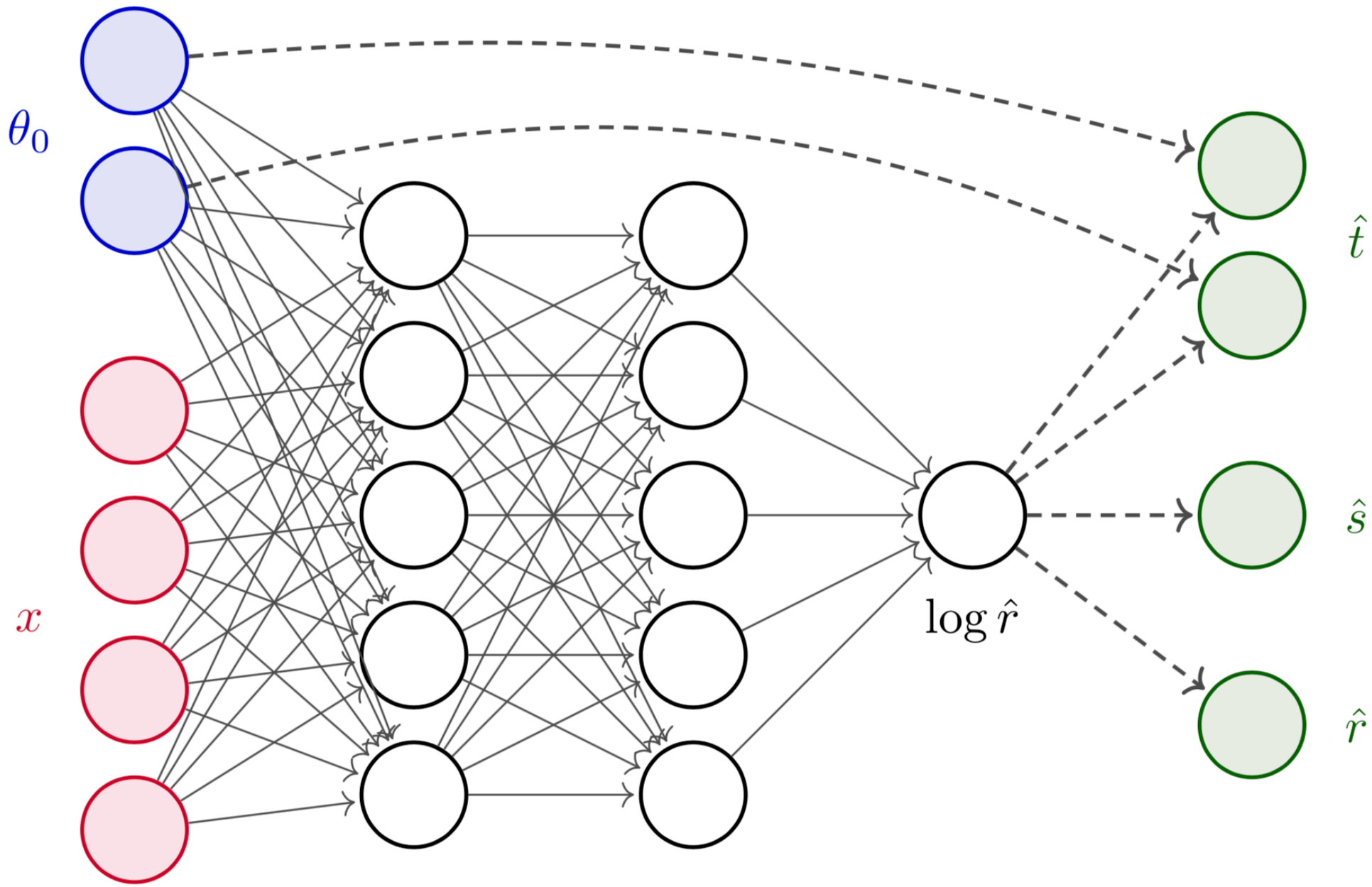
$$s(x;\theta_0,\theta_1) = \frac{p(x|\theta_1)}{p(x|\theta_0) + p(x|\theta_1)}$$

K.C., G. Louppe, J. Pavez: Approximating Likelihood Ratios with Calibrated Discriminative Classifiers [arXiv:1506.02169]

POINT BY POINT



(AGNOSTIC) PARAMETERIZED ESTIMATORS



Calibration

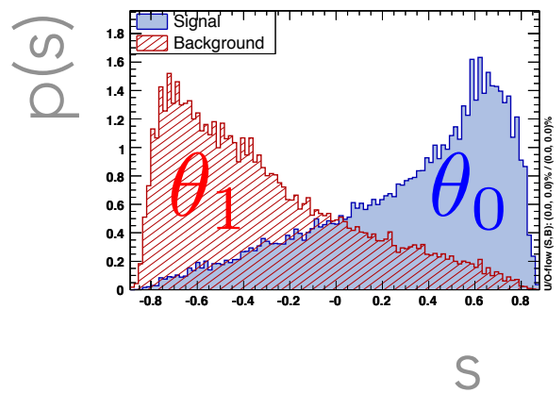
CALIBRATING THE LIKELIHOOD-RATIO TRICK

The intractable likelihood ratio based on high-dimensional features x is:

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

We can show that an **equivalent test** can be made from 1-D projection

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



if the scalar map s: X → $\mathbb{R}$ has the same level sets as the likelihood ratio

$$s(x;\theta_0;\theta_1) = \text{monotonic}[\ p(x|\theta_0)/p(x|\theta_1)\ ]$$

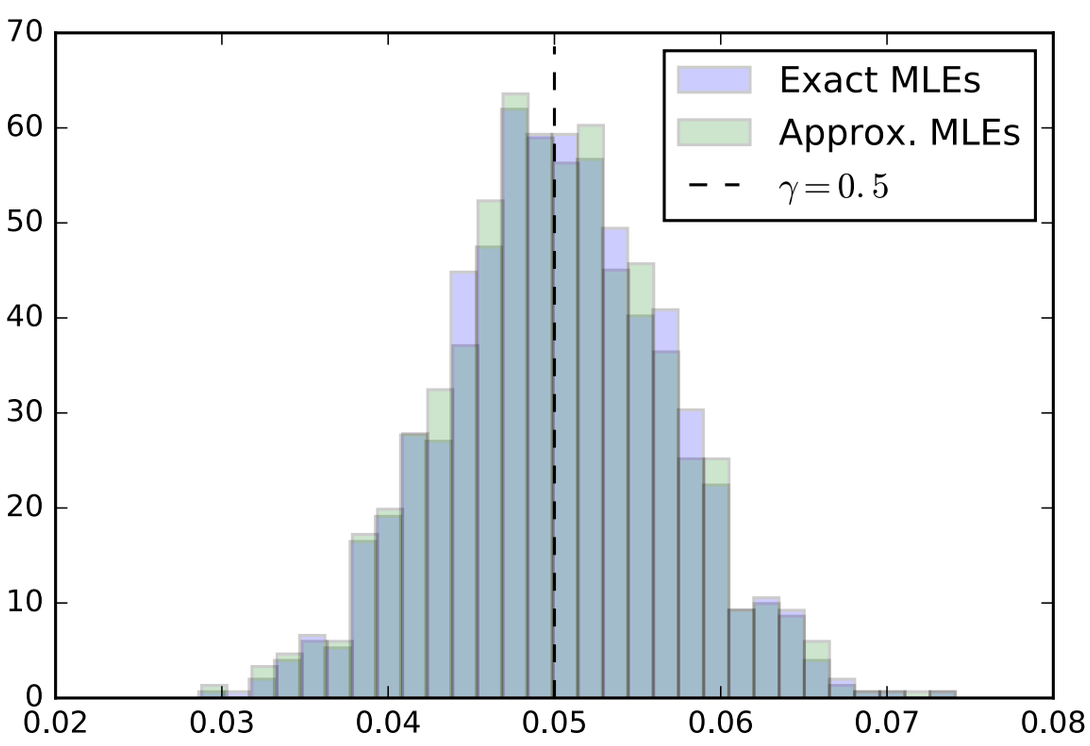
Estimating the density of $s(x;\theta_0,\theta_1)$ via the simulator calibrates the ratio.

K.C., G. Louppe, J. Pavez: Approximating Likelihood Ratios with Calibrated Discriminative Classifiers [arXiv:1506.02169]

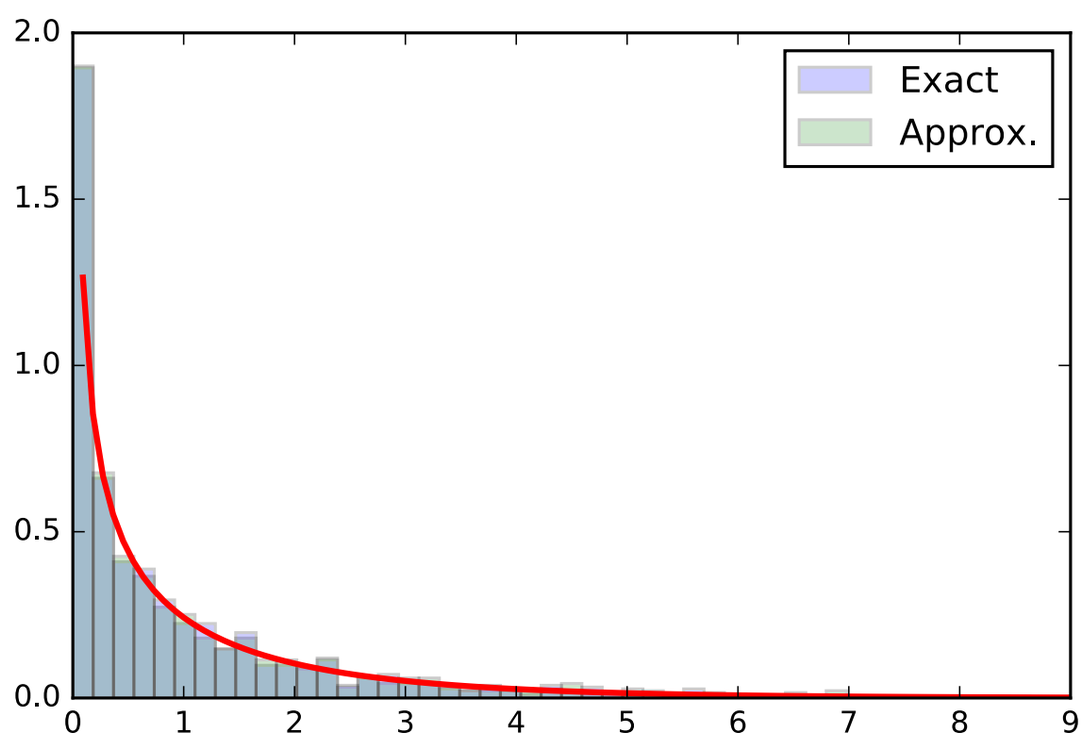
AMORTIZED LIKELIHOOD-FREE INFERENCE

Once we've learned the function s(x; θ) to approximate the likelihood, we can apply it to any data x.

- unlike MCMC, we pay biggest computational costs up front
- Here we repeat inference thousands of times & check asymptotic statistical theory



(a) Exact vs. approximated MLEs.



(b) $p(-2\log\Lambda(\gamma = 0.05)\,|\,\gamma = 0.05)$

K.C., G. Louppe, J. Pavez: http://arxiv.org/abs/1506.02169

Diagnostics

In practice $\hat{r}(\hat{s}(\mathbf{x};\theta_0,\theta_1))$ will not be exact. Diagnostic procedures are needed to assess the quality of this approximation.

1. For inference, the value of the MLE $\hat{\theta}$ should be independent of the value of θ_1 used in the denominator of the ratio.

The denominator in the likelihood ratio is just a shift

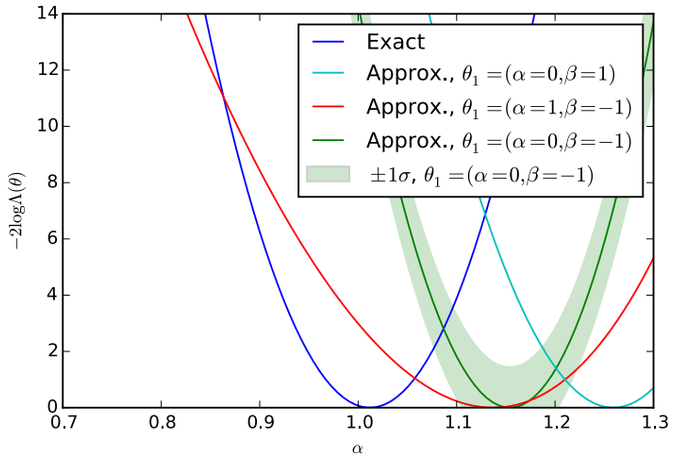
$$\hat{\theta} = \arg\max_{\theta} \sum \ln \frac{p(x_e|\theta)}{p(x_e|\theta_1)} = \arg\max_{\theta} \sum \ln \frac{p(s(x_e;\theta,\theta_1)|\theta)}{p(s(x_e;\theta,\theta_1)|\theta_1)} \,. \tag{4.4}$$

It is important that we include the denominator $p(s(x_e;\theta,\theta_1)|\theta_1)$ because this cancels Jacobian factors that vary with θ.

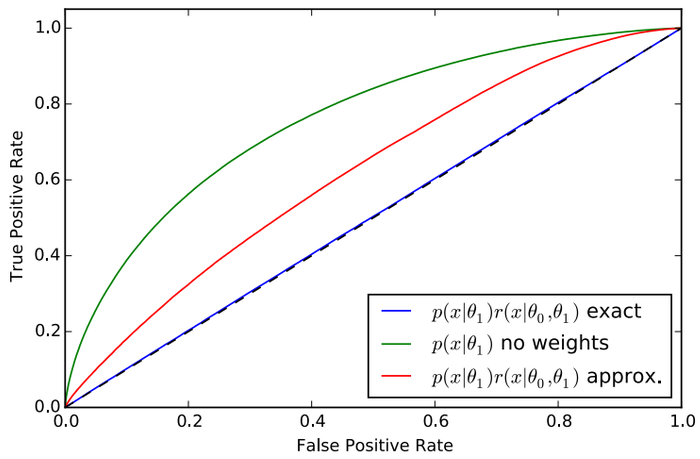
Provides a non-trivial diagnostic:

$$\frac{p_1(s^*)}{p_0(s^*)} = \frac{p_1(x)}{p_0(x)} \boxed{\frac{\int d\Omega_{s^*} p_0(x)/|\hat{n}\cdot\nabla s|}{\int d\Omega_{s^*} p_0(x)/|\hat{n}\cdot\nabla s|}} = \frac{p_1(x)}{p_0(x)}$$

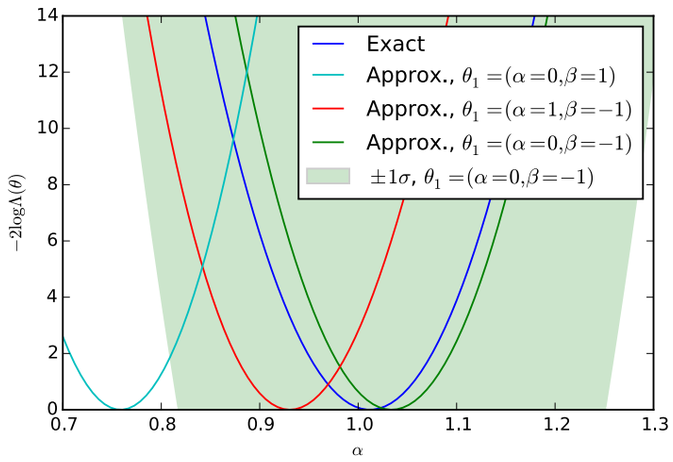
DIAGNOSTICS



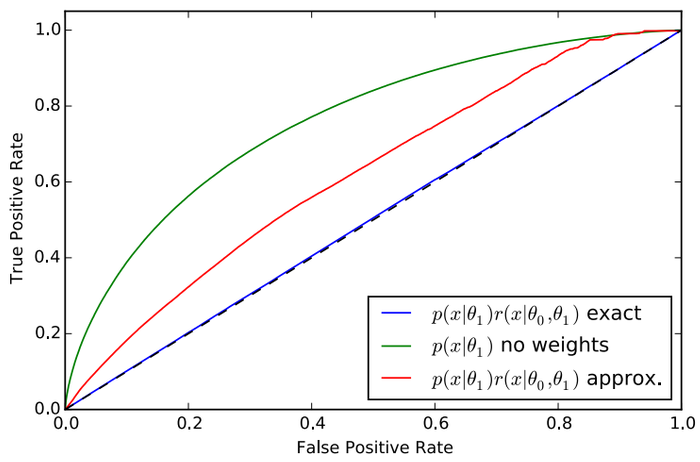
(a) Poorly trained, well calibrated.



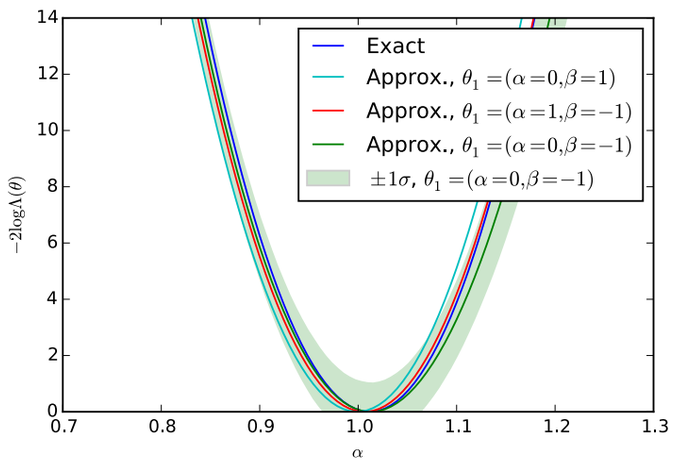
(b) Poorly trained, well calibrated.



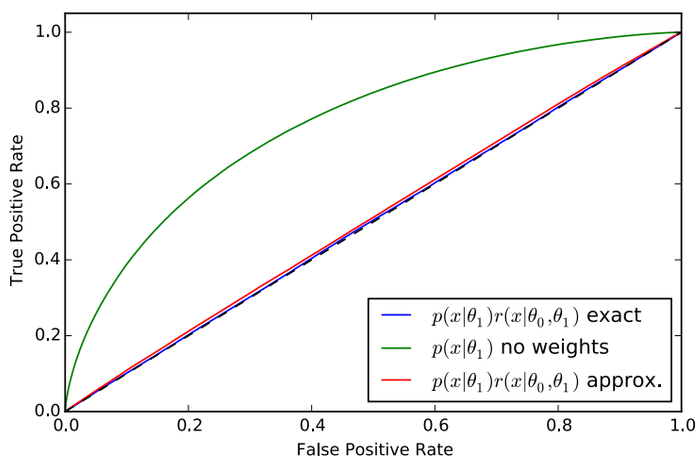
(c) Poorly calibrated, well trained.



(d) Poorly calibrated, well trained.



(e) Well trained, well calibrated.



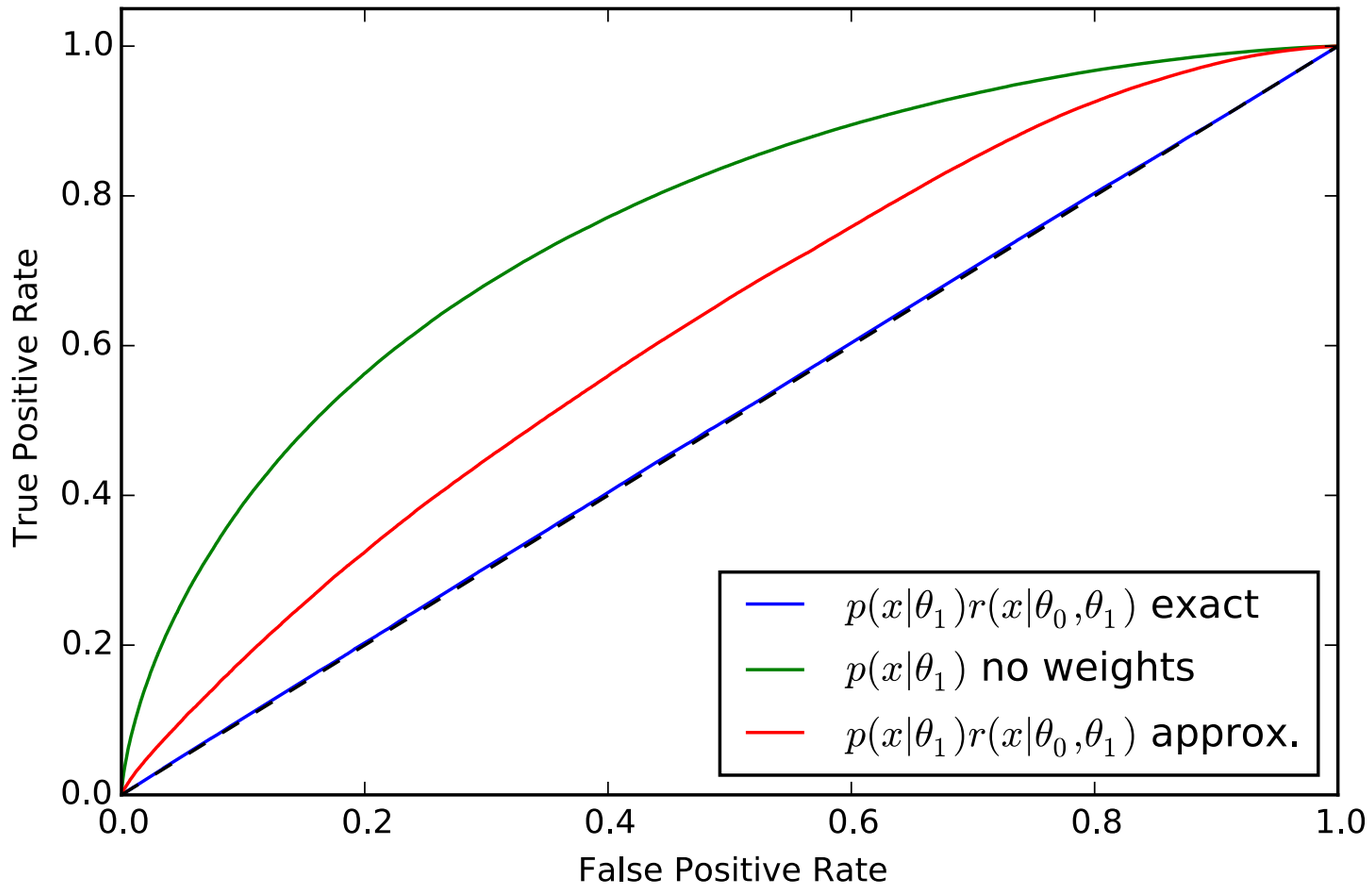
(f) Well trained, well calibrated.

DIAGNOSTICS WITH AN ADVERSARY

Train a new classifier to **discriminate** between events from target $p(x|\theta_0)$ and events resampled from original distribution $p(x|\theta_1)$ with probabilities given by the predicted weights $\hat{r}(x|\theta_0, \theta_1) \approx p(x|\theta_0)/p(x|\theta_1)$

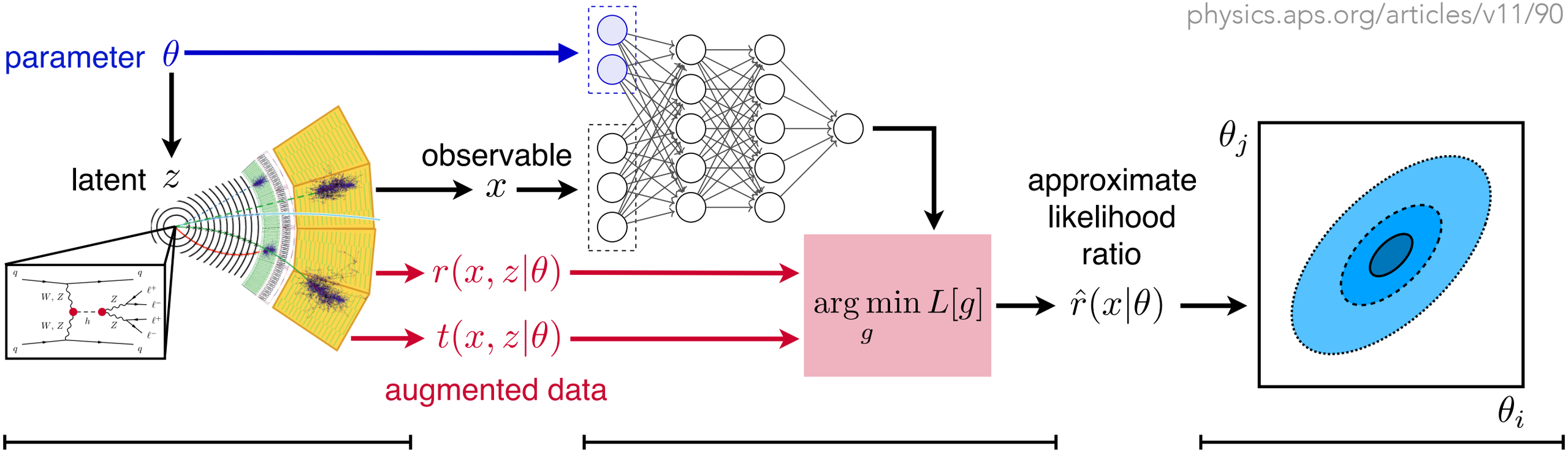
- classifier can easily distinguish unweighted distributions;
- exact weights are perfect (AUC~0.5)

Important:
Performance evaluated on independent testing sample



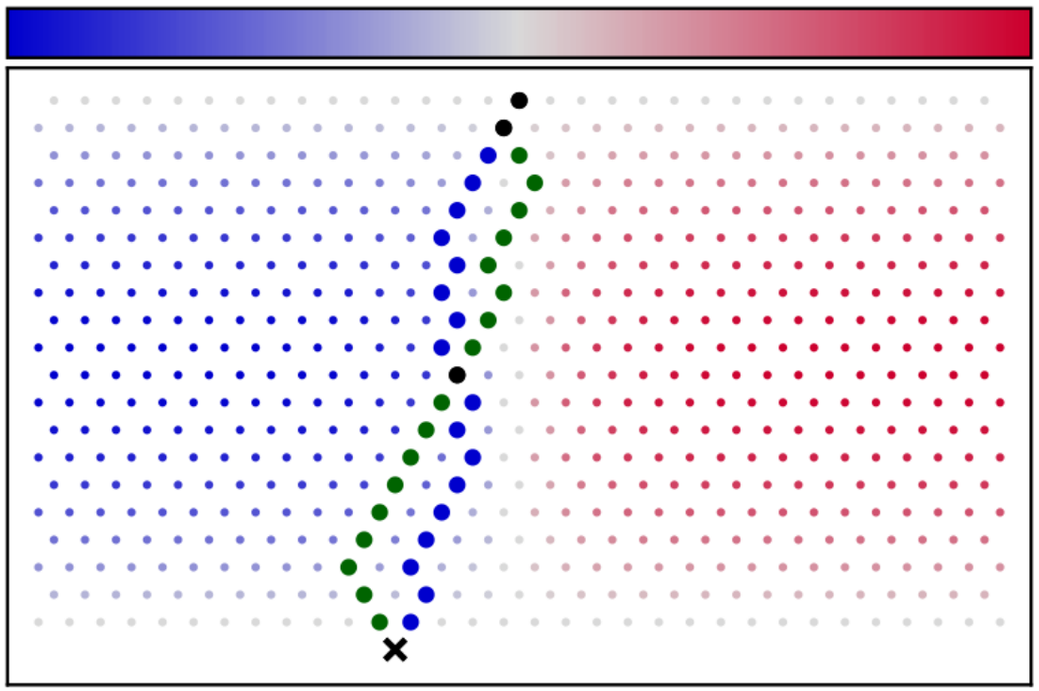
LEARNING THE LIKELIHOOD RATIO

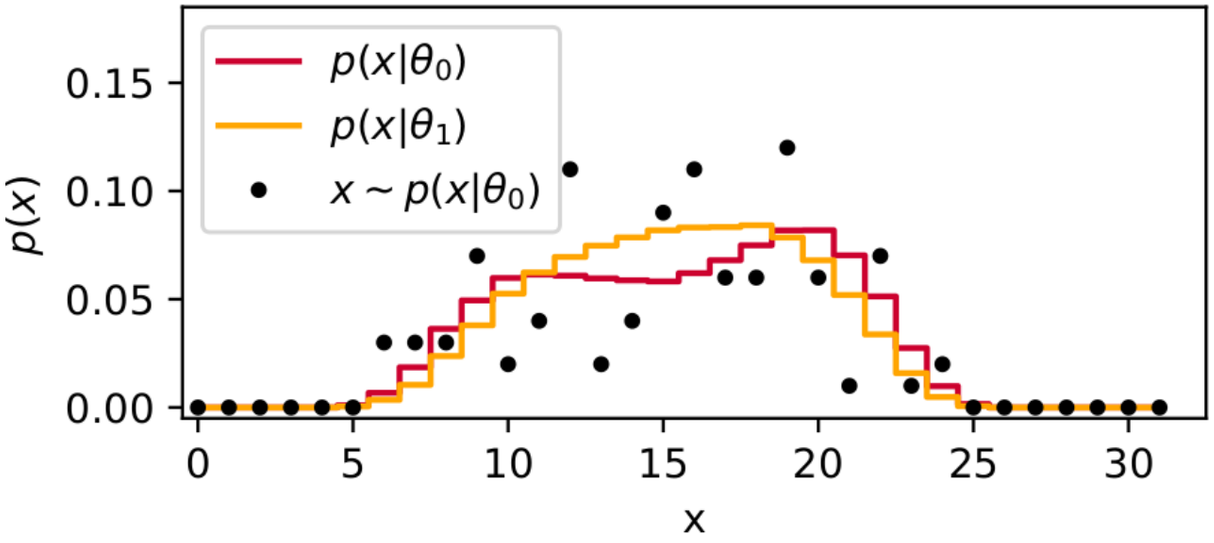
arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
physics.aps.org/articles/v11/90



Recently, we realized we can **extract more from the simulator**.
We can use **augmented data** to improve training

MINING GOLD





While implicit density is intractable

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

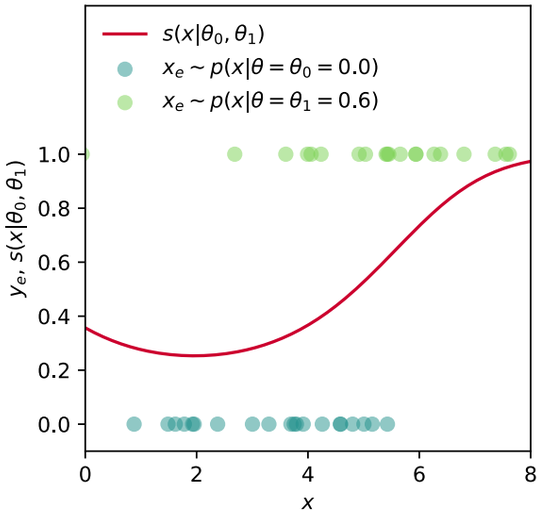
Some quantities conditioned on latent z are tractable:

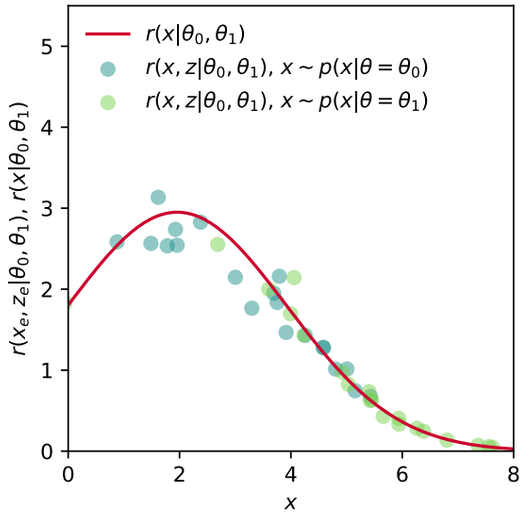
$$r(x, z|\theta_0, \theta_1) = \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)}$$

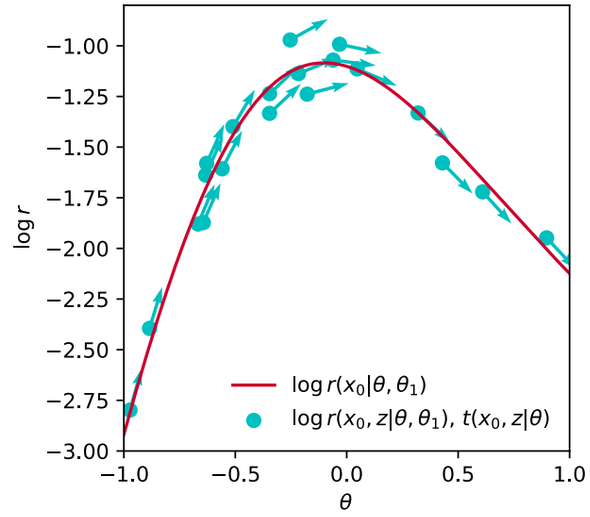
and similar to REINFORCE policy gradient

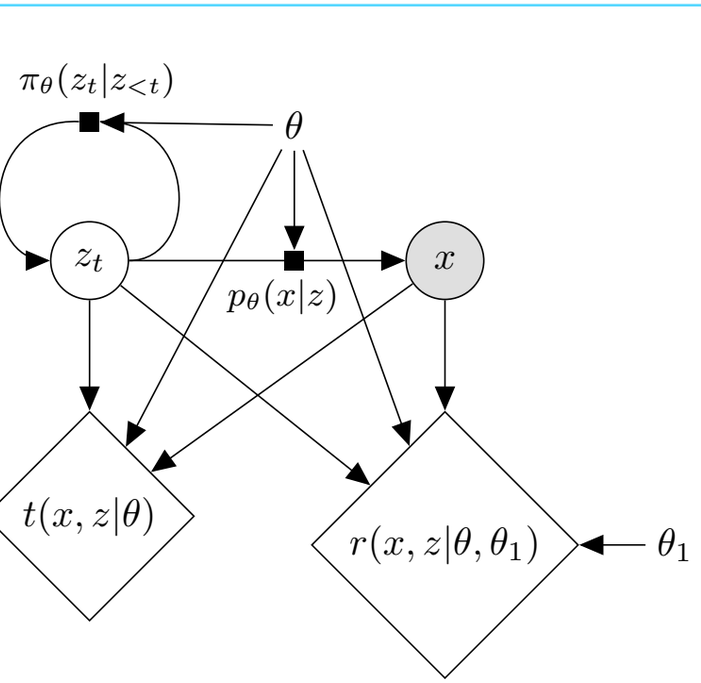
$$t(x, z|\theta_0) = \frac{\nabla_\theta p(x, z|\theta)|_{\theta_0}}{p(x, z|\theta_0)} = \nabla_\theta \log p(x, z|\theta)|_{\theta_0}$$

PUTTING IT ALL TOGETHER

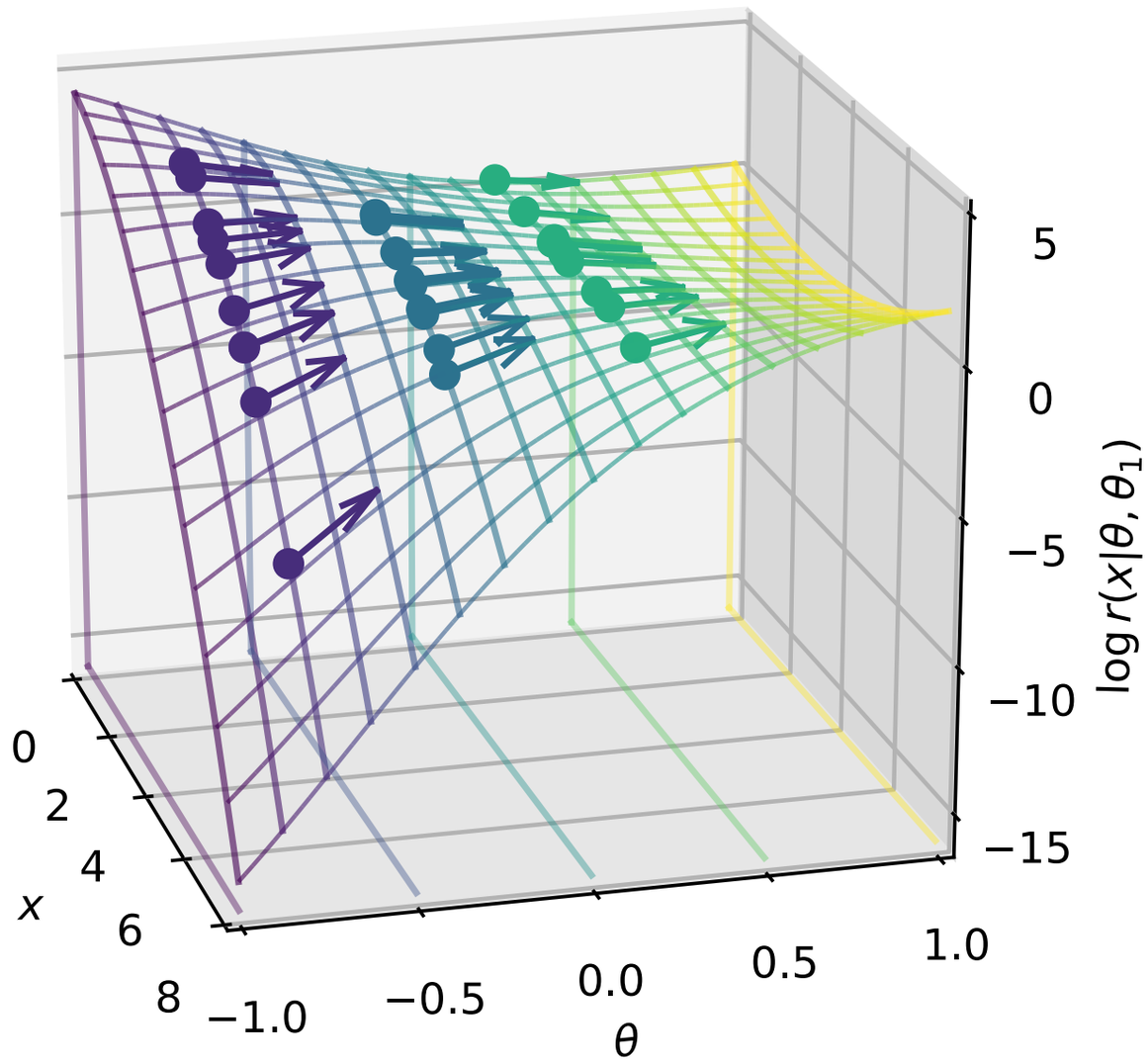








can think of simulator as policy πθ in language of reinforcement learning



HOW DO YOU USE THE GOLD?

We have joint likelihood ratio

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

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

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

One can show it is minimized by

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

We want likelihood ratio

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

LEARNING THE SCORE

Similar to the joint likelihood ratio, we can calculate the joint score

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

We want score

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

LEARNING THE SCORE

Similar to the joint likelihood ratio, we can calculate the joint score

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

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

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

One can show it is minimized by

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

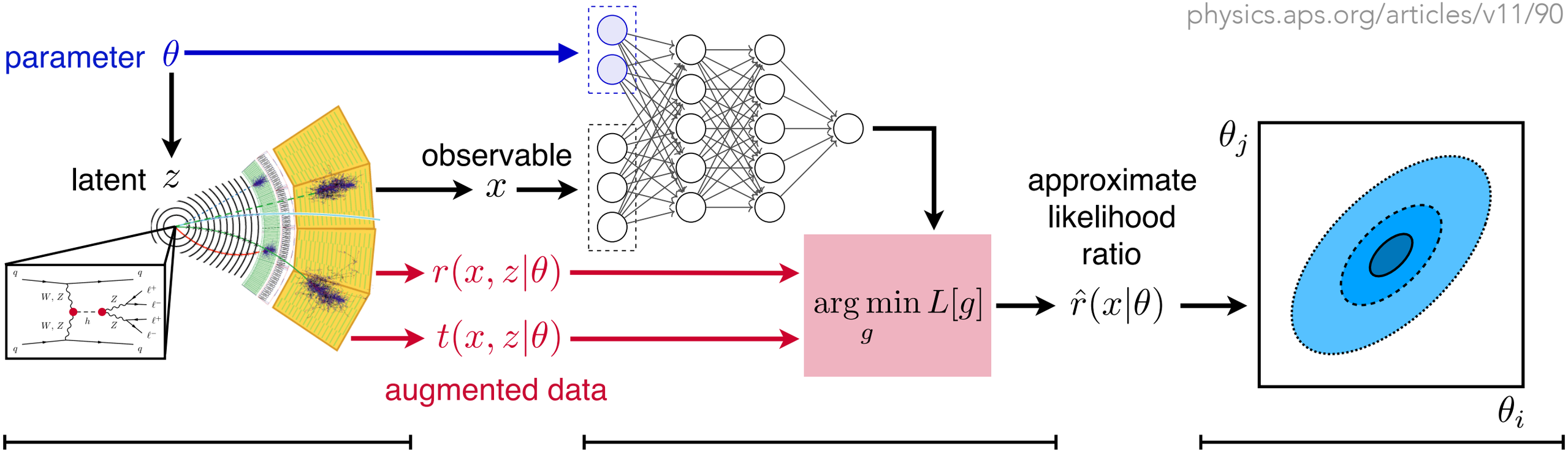
We want score

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

Again, we implement this minimization through machine learning

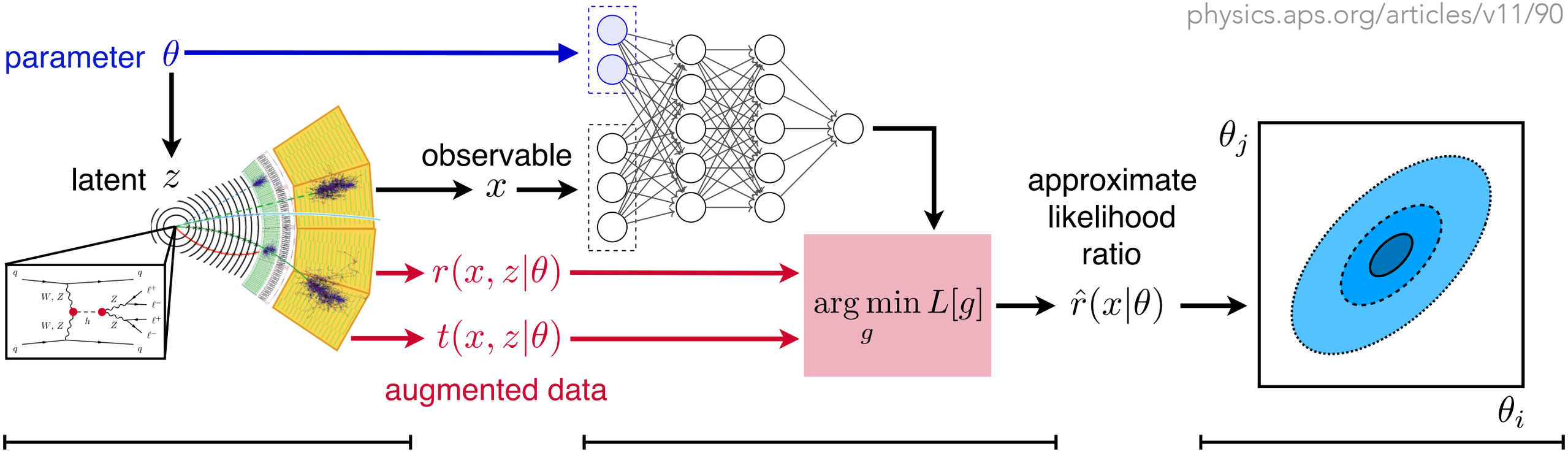
LEARNING THE LIKELIHOOD RATIO

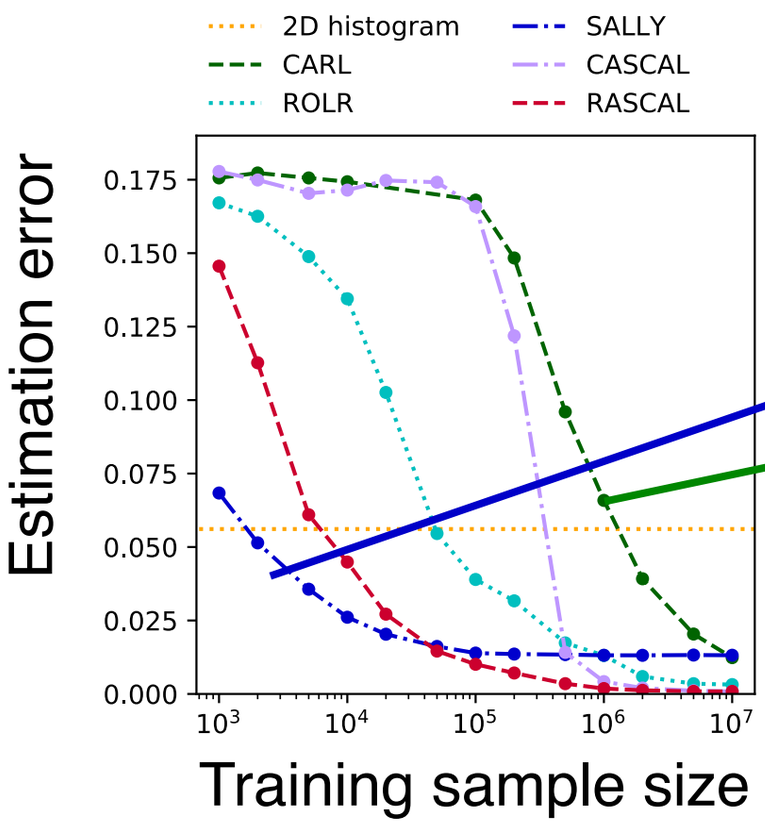
arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
physics.aps.org/articles/v11/90



LEARNING THE LIKELIHOOD RATIO

arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
physics.aps.org/articles/v11/90

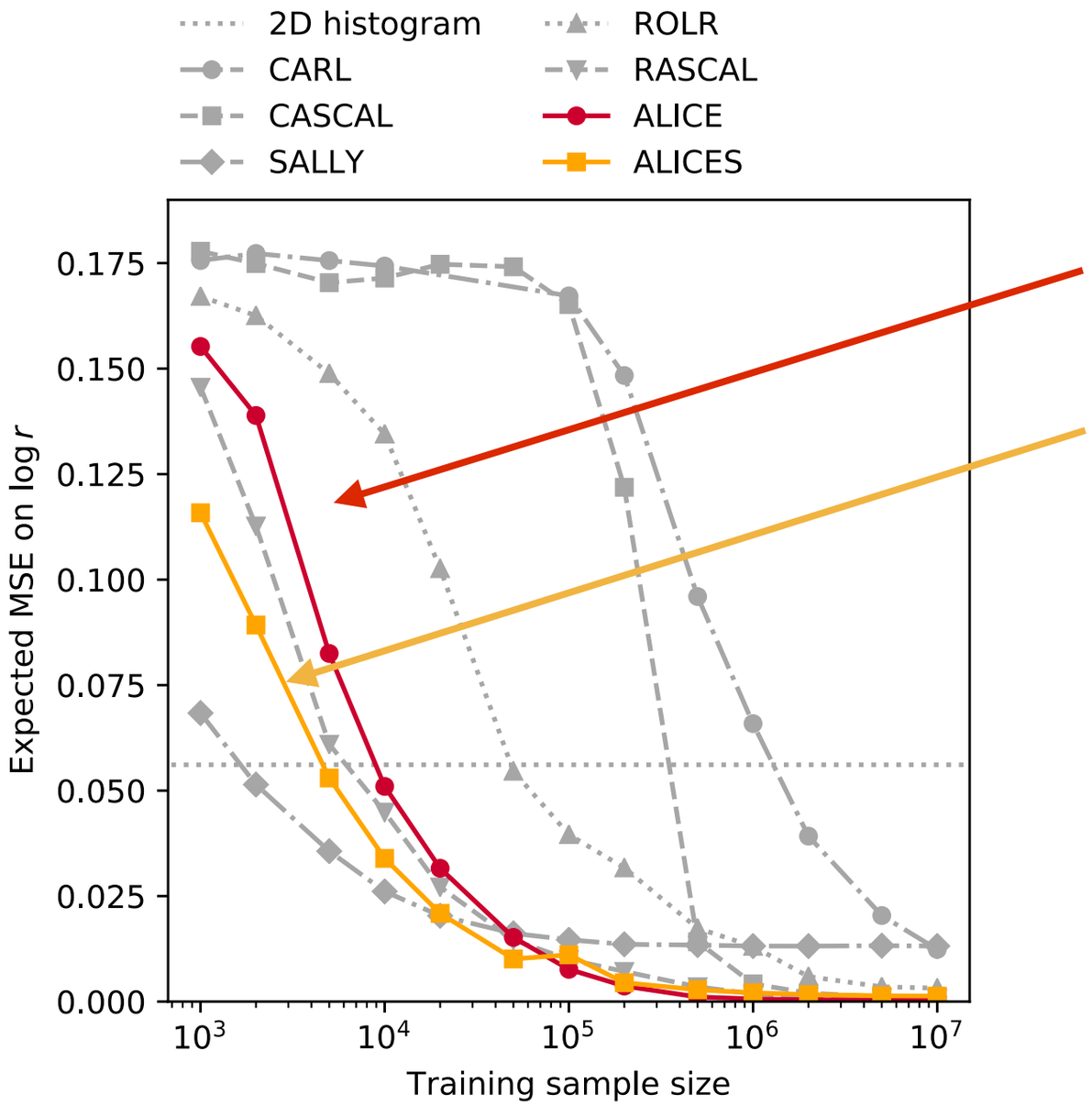




New techniques
require less data than
without augmented data

Traditional Approach no NN

RESULTS WITH IMPROVED CROSS-ENTROPY ESTIMATOR



The improved cross-entropy estimator performs better than the squared loss.

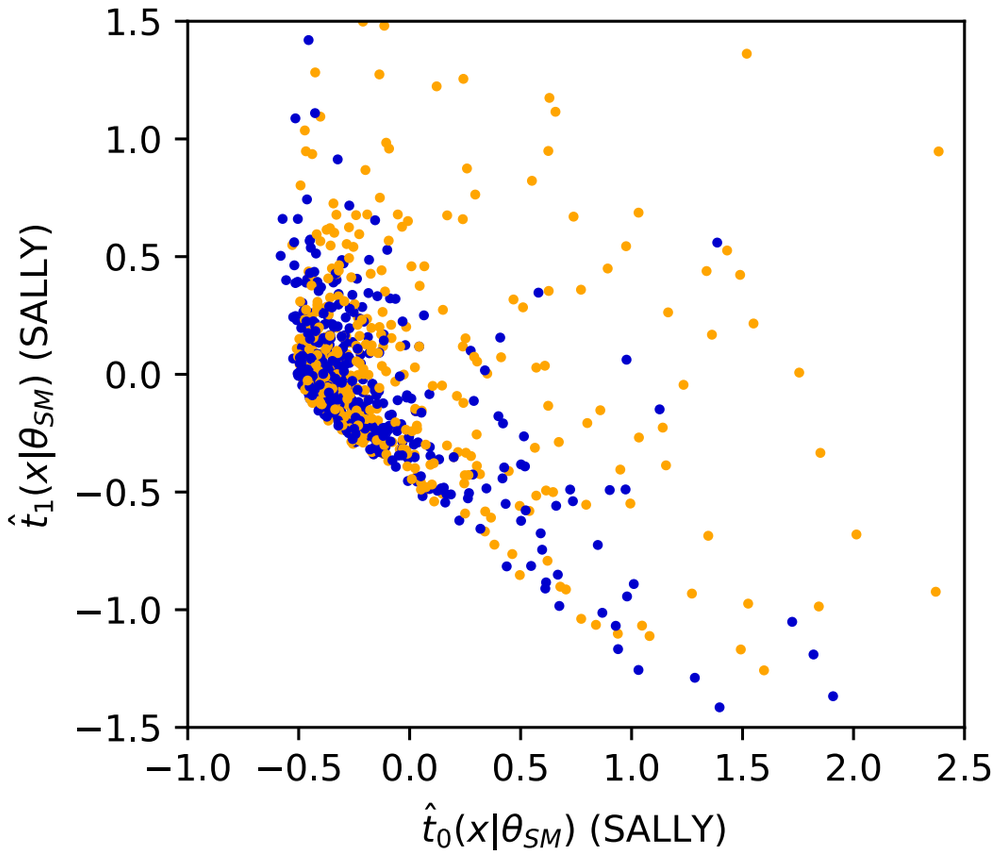
Better use of the joint ratio

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

LOCALLY SUFFICIENT STATISTICS

One of the initial motivations for using ML to approximate the likelihood is that most engineered features loose information.

However, the score provides "locally sufficient statistics" that capture all the information in the region of neighborhood of θ_0 (aka the standard model)



One summary statistic per parameter

$$p(x|\theta) \sim e^{t(x|\theta_{SM})\cdot(\theta-\theta_{SM})}$$

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

ML version of optimal observables

SIMILAR

MNRAS **000**, 1–14 (2017) Preprint 28 March 2018 Compiled using MNRAS LATEX style file v3.0

Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology

Justin Alsing,[1,2]★ Benjamin Wandelt[1,3,4,5] and Stephen Feeney[1]
[1] *Center for Computational Astrophysics, Flatiron Institute, 162 5th Ave, New York City, NY 10010, USA*
[2] *Imperial Centre for Inference and Cosmology, Department of Physics, Imperial College London, Blackett Laboratory, Prince Consort Road, London SW7 2AZ, UK*
[3] *Institut d'Astrophysique de Paris (IAP), UMR 7095, CNRS UPMC Universite Paris 6, Sorbonne Universitse, 98bis boulevard Arago, F-75014 Paris, France*
[4] *Institut Lagrange de Paris (ILP), Sorbonne Universitse, 98bis boulevard Arago, F-75014 Paris, France*
[5] *Department of Physics and Astronomy, University of Illinois at Urbana-Champaign, 1002 W Green St, Urbana, IL 61801, USA*

Automatic physical inference with information maximising neural networks

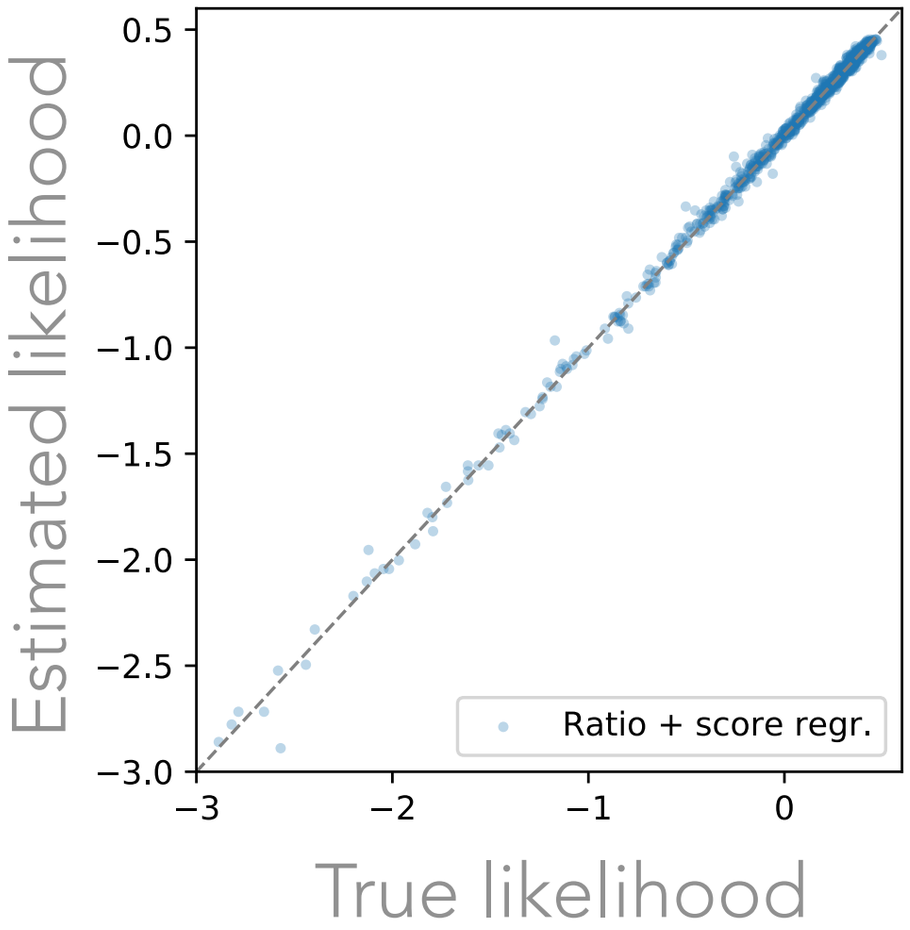
Tom Charnock*
Sorbonne Université, CNRS, UMR 7095, Institut d'Astrophysique de Paris, 98 bis boulevard Arago, 75014 Paris, France

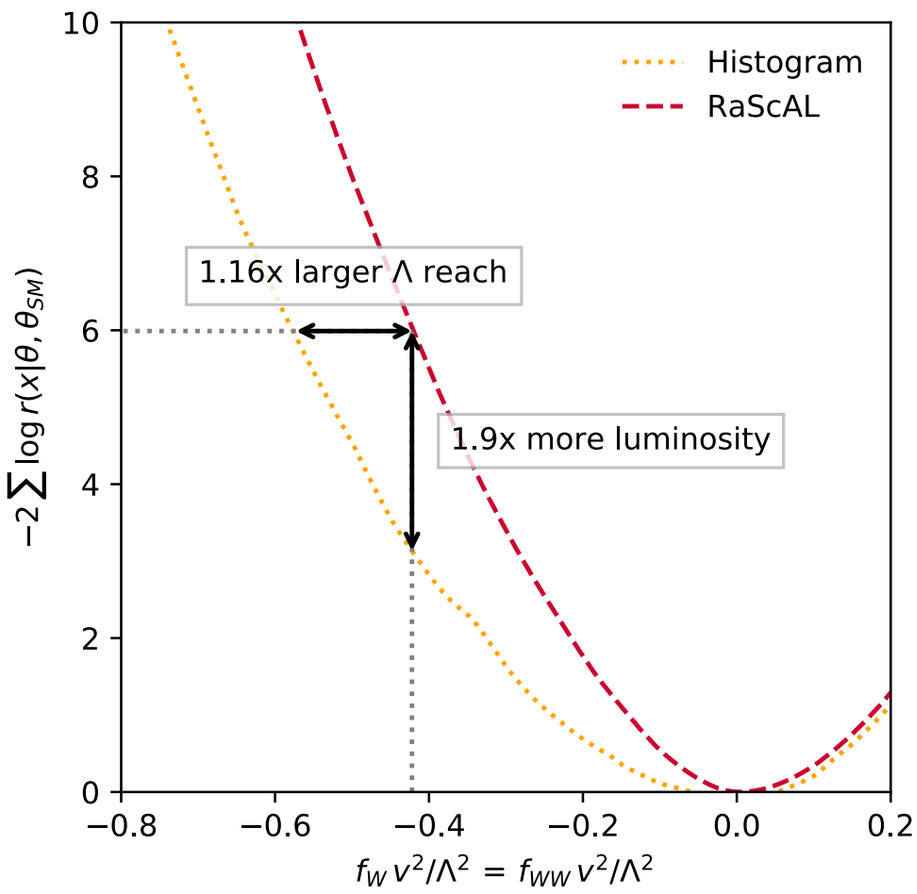
Guilhem Lavaux†
Sorbonne Université, CNRS, UMR 7095, Institut d'Astrophysique de Paris, 98 bis boulevard Arago, 75014 Paris, France and Sorbonne Universités, Institut Lagrange de Paris, 98 bis boulevard Arago, 75014 Paris, France

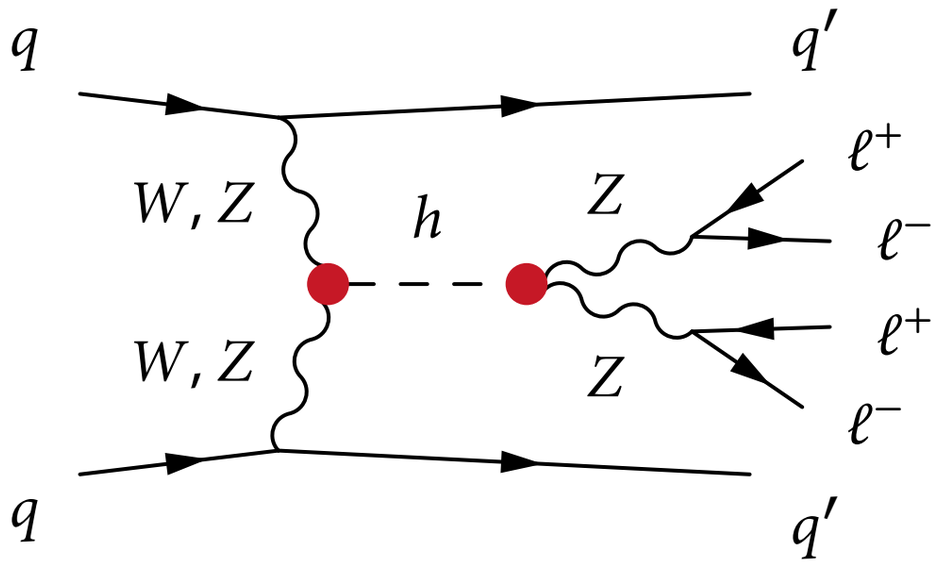
Benjamin D. Wandelt‡
Center for Computational Astrophysics, Flatiron Institute, 162 5th Avenue, 10010, New York, NY, USA
Sorbonne Université, CNRS, UMR 7095, Institut d'Astrophysique de Paris, 98 bis boulevard Arago, 75014 Paris, France
Sorbonne Universités, Institut Lagrange de Paris, 98 bis boulevard Arago, 75014 Paris, France and Department of Astrophysical Sciences, 4 Ivy Lane, Princeton University, Princeton, NJ 08544, USA

IMPACT ON STUDIES OF THE HIGGS BOSON

(based on a 42-Dim observation ***x***)







J Brehmer, J Pavez, G Louppe, K.C. PRL & PRD 2018 [arXiv:1805.00013 & arXiv:1805.00020]
"Better Higgs Measurements Through Information Geometry" [arXiv:1612.05261] & CARL [arxiv:1506.02169]

Approximate Likelihood Ratios In Bayesian Setting

BAYESIAN APPROACHES

"ABC" ε-likelihood

Approx Likelihood Ratio

MCMC

Standard ABC

VI

arXiv:1606.08549

Automatic Variational ABC

Alexander Moreno[*1], **Tameem Adel**[2,3], **Edward Meeds**[2,4], **James M. Rehg**[1] and **Max Welling**[2,5]

[1]College of Computing, Georgia Institute of Technology
[2]Machine Learning Group, University of Amsterdam
[3]Data Science, Radboud University
[4]Center for Integrative Bioinformatics IBIVU, VU University
[5]Canadian Institute for Advanced Research (CIFAR)

arXiv:1611.10242

Likelihood-free inference by ratio estimation

Owen Thomas*, Ritabrata Dutta†, Jukka Corander*, Samuel Kaski‡ and Michael U. Gutmann§,¶

arXiv:1903.04057

Likelihood-free MCMC with Approximate Likelihood Ratios

Joeri Hermans
joeri.hermans@doct.uliege.be
University of Liège, Belgium

Volodimir Begy
volodimir.begy@cern.ch
University of Vienna, Austria
CERN, Switzerland

Gilles Louppe
g.louppe@uliege.be
University of Liège, Belgium

arXiv:1702.08896

Hierarchical Implicit Models and Likelihood-Free Variational Inference

Dustin Tran
Columbia University

Rajesh Ranganath
Princeton University

David M. Blei
Columbia University

[78] M.-N. Tran, D. J. Nott, and R. Kohn. Variational Bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*, 26(4):873–882, 2017.

Hierarchical Implicit Models and Likelihood-Free Variational Inference

Dustin Tran
Columbia University

Rajesh Ranganath
Princeton University

David M. Blei
Columbia University

Abstract

Implicit probabilistic models are a flexible class of models defined by a simulation process for data. They form the basis for theories which encompass our understanding of the physical world. Despite this fundamental nature, the use of implicit models remains limited due to challenges in specifying complex latent structure in them, and in performing inferences in such models with large data sets. In this paper, we first introduce *hierarchical implicit models* (HIMS). HIMS combine the idea of implicit densities with hierarchical Bayesian modeling, thereby defining models via simulators of data with rich hidden structure. Next, we develop *likelihood-free variational inference* (LFVI), a scalable variational inference algorithm for HIMS. Key to LFVI is specifying a variational family that is also implicit. This matches the model's flexibility and allows for accurate approximation of the posterior. We demonstrate diverse applications: a large-scale physical simulator for predator-prey populations in ecology; a Bayesian generative adversarial network for discrete data; and a deep implicit model for text generation.

$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{\beta},\mathbf{z}\,|\,\mathbf{x})}[\log p(\mathbf{x},\mathbf{z},\boldsymbol{\beta}) - \log q(\boldsymbol{\beta},\mathbf{z}\,|\,\mathbf{x})].$$

$$\mathcal{L} \propto \mathbb{E}_{q(\boldsymbol{\beta})}[\log p(\boldsymbol{\beta}) - \log q(\boldsymbol{\beta})] + \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\beta})q(\mathbf{z}_n\,|\,\mathbf{x}_n,\boldsymbol{\beta})}\left[\log \frac{p(\mathbf{x}_n,\mathbf{z}_n\,|\,\boldsymbol{\beta})}{q(\mathbf{x}_n,\mathbf{z}_n\,|\,\boldsymbol{\beta})}\right]. \quad (4)$$

(Here the proportionality symbol means equality up to additive constants.) Thus the ELBO is a function of the ratio of two intractable densities. If we can form an estimator of this ratio, we can proceed with optimizing the ELBO.

[52] A. Moreno, T. Adel, E. Meeds, J. M. Rehg, and M. Welling. Automatic variational ABC. *arXiv:1606.08549*, 2016.

[78] M.-N. Tran, D. J. Nott, and R. Kohn. Variational Bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*, 26(4):873–882,

arxiv (December 2018)

Likelihood-free inference by ratio estimation

Owen Thomas*, Ritabrata Dutta†, Jukka Corander*, Samuel Kaski‡ and Michael U. Gutmann§,¶

Abstract. We consider the problem of parametric statistical inference when likelihood computations are prohibitively expensive but sampling from the model is possible. Several so-called likelihood-free methods have been developed to perform inference in the absence of a likelihood function. The popular synthetic likelihood approach infers the parameters by modelling summary statistics of the data by a Gaussian probability distribution. In another popular approach called approximate Bayesian computation, the inference is performed by identifying parameter values for which the summary statistics of the simulated data are close to those of the observed data. Synthetic likelihood is easier to use as no measure of "closeness" is required but the Gaussianity assumption is often limiting. Moreover, both approaches require judiciously chosen summary statistics. We here present an alternative inference approach that is as easy to use as synthetic likelihood but not as restricted in its assumptions, and that, in a natural way, enables automatic selection of relevant summary statistic from a large set of candidates. The basic idea is to frame the problem of estimating the posterior as a problem of estimating the ratio between the data generating distribution and the marginal distribution. This problem can be solved by logistic regression, and including regularising penalty terms enables automatic selection of the summary statistics relevant to the inference task. We illustrate the general theory on canonical examples and employ it to perform inference for challenging stochastic nonlinear dynamical systems and high-dimensional summary statistics.

Keywords: approximate Bayesian computation, density-ratio estimation, likelihood-free inference, logistic regression, stochastic dynamical systems, summary statistics selection, synthetic likelihood

The basic idea is to frame the original problem of estimating the posterior as a problem of estimating the ratio $r(x,\theta)$ between the data generating pdf $p(x|\theta)$ and the marginal distribution $p(x)$, in the context of a Bayesian belief update

$$r(x,\theta) = \frac{p(x|\theta)}{p(x)}. \quad (2)$$

arXiv:1903.04057

Likelihood-free MCMC with Approximate Likelihood Ratios

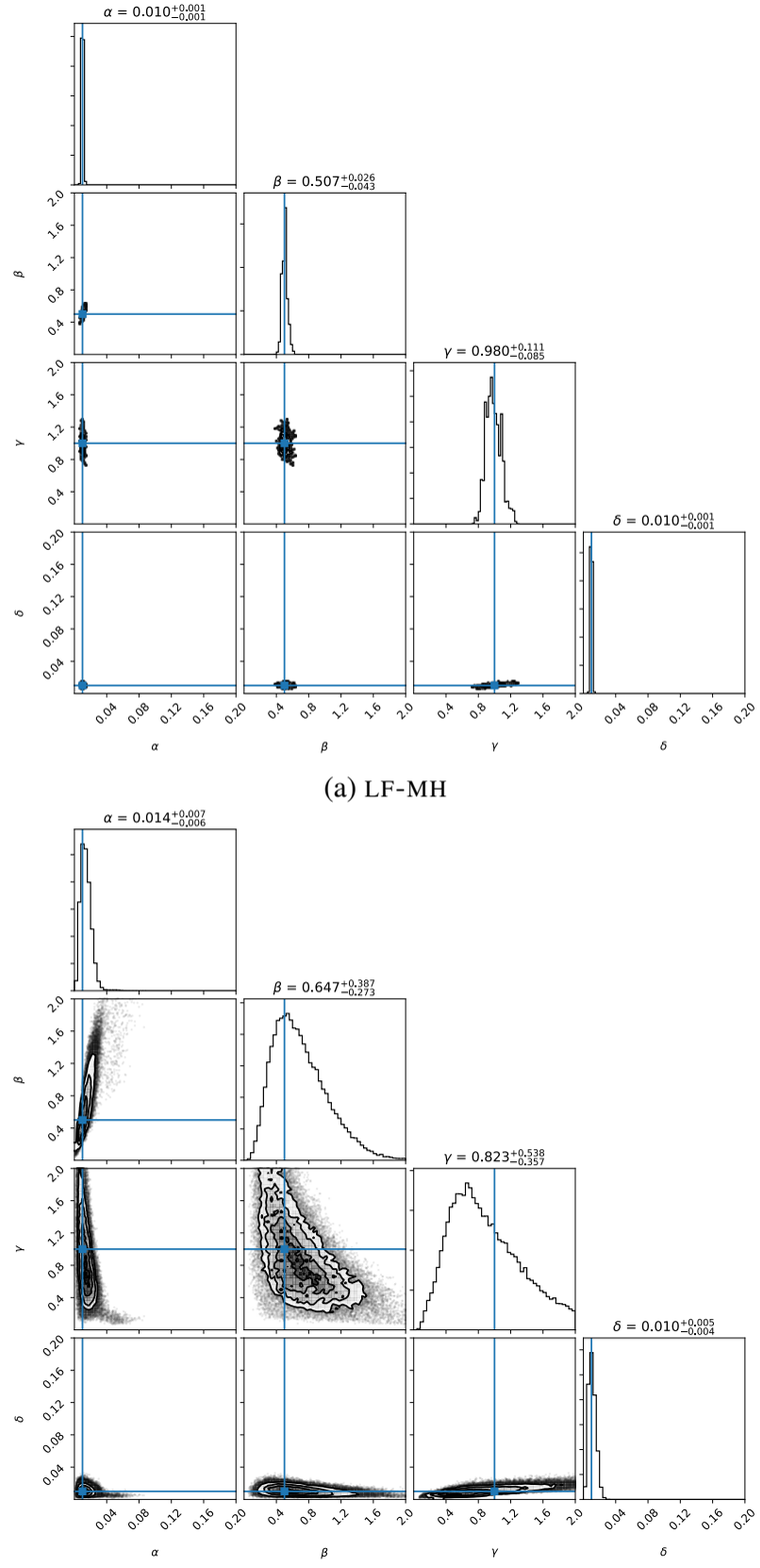
Joeri Hermans
joeri.hermans@doct.uliege.be
University of Liège, Belgium

Volodimir Begy
volodimir.begy@cern.ch
University of Vienna, Austria
CERN, Switzerland

Gilles Louppe
g.louppe@uliege.be
University of Liège, Belgium

Approximate likelihood ratio used Metropolis-Hastings

And when trained with a parametrized classifier, it is differentiabl eand can be used in Hamiltonian (Hybrid) Monte Carlo



(a) LF-MH

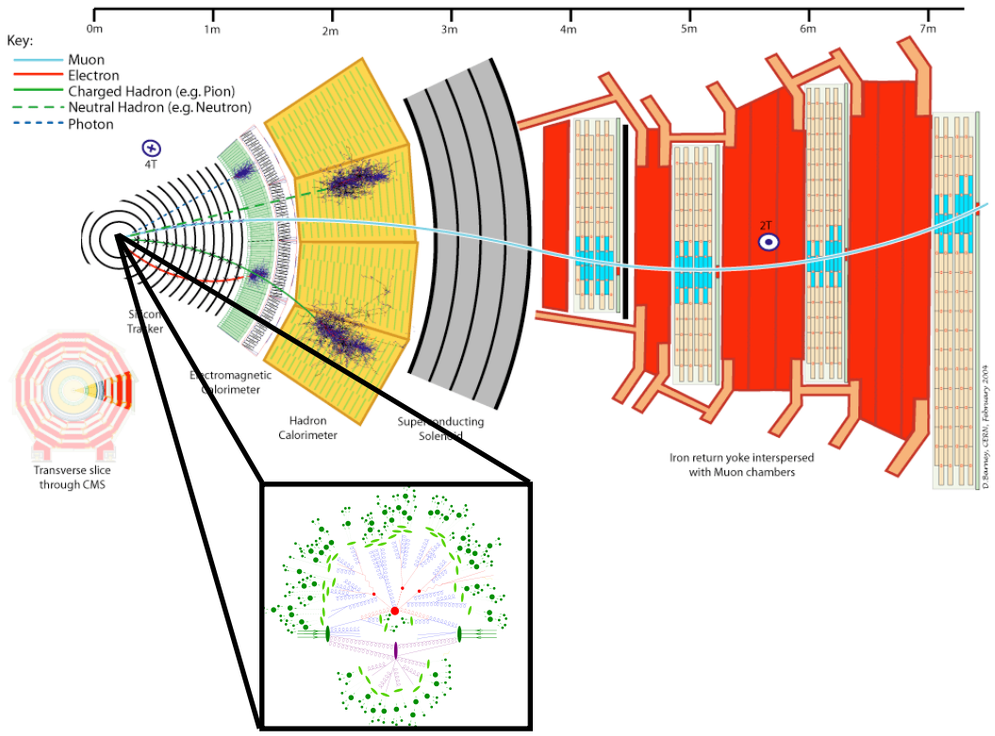
(b) ABC

Figure 5: Approximate posterior for the Lotka-Volterra population model. ABC samples have been drawn under an acceptance threshold of $\epsilon = 2.5$.

TWO APPROACHES

Use simulator

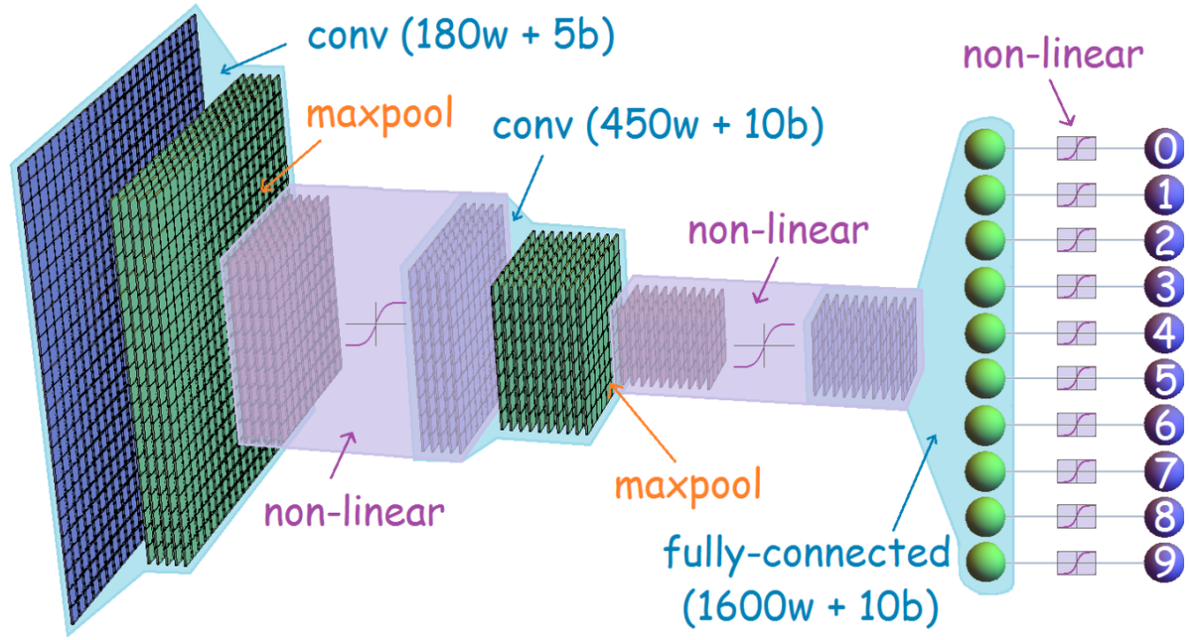
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator

(with deep learning)



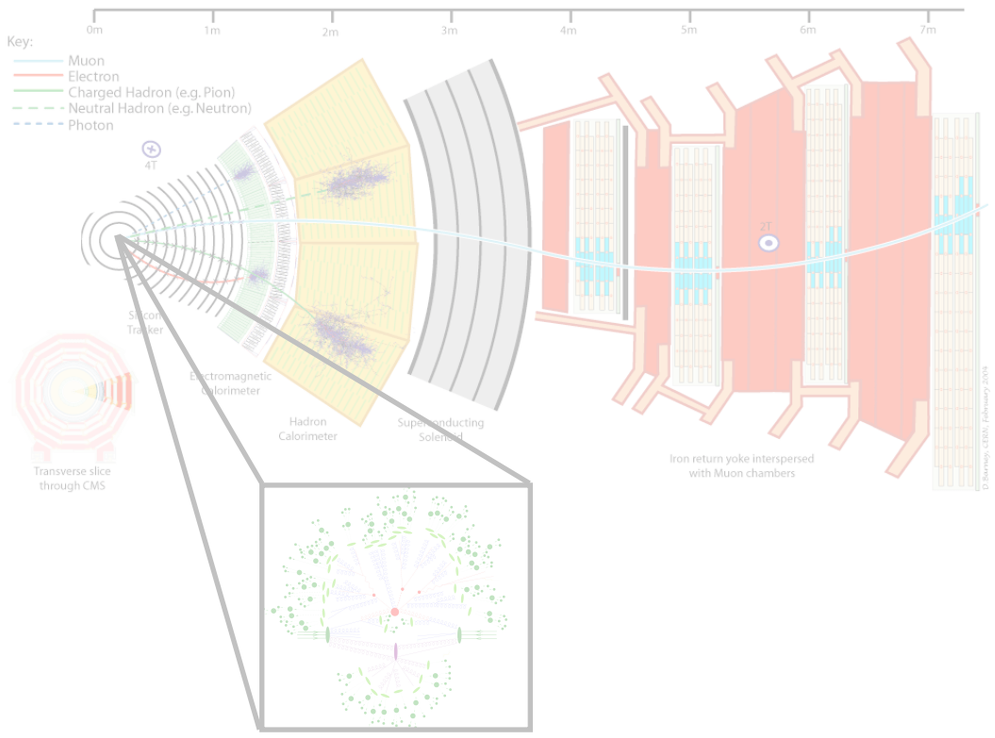
- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

TWO APPROACHES

Use simulator

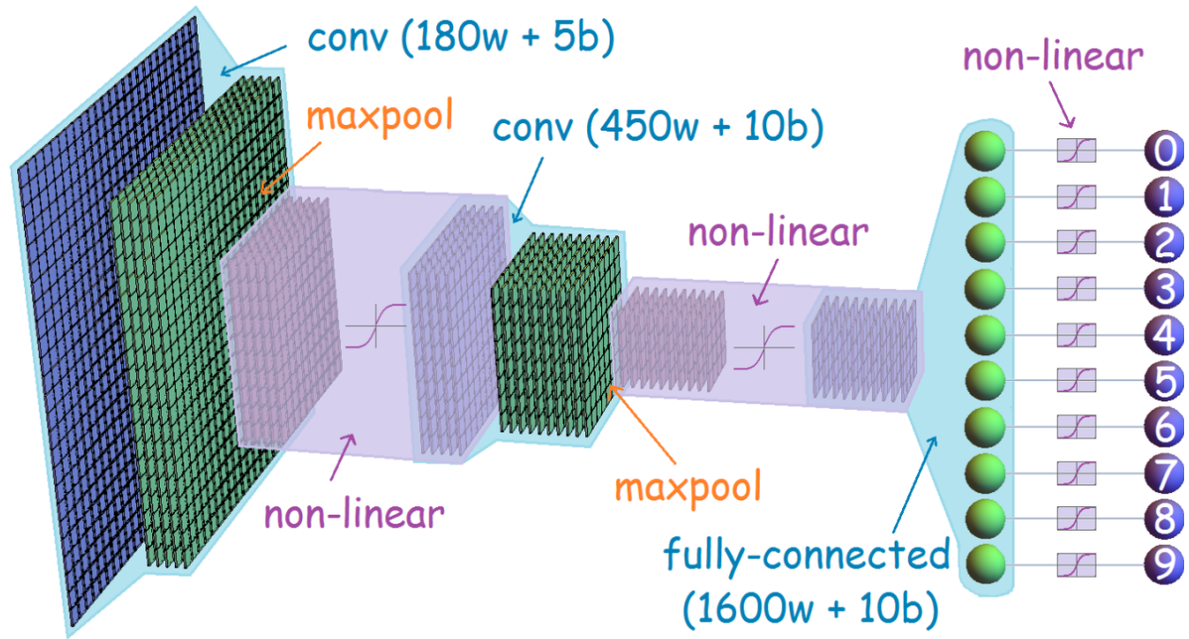
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator

(with deep learning)



- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

GENERATIVE MODELS

arXiv:1712.10321
arXiv:1705.02355
DOI: 10.5281/zenodo.198541

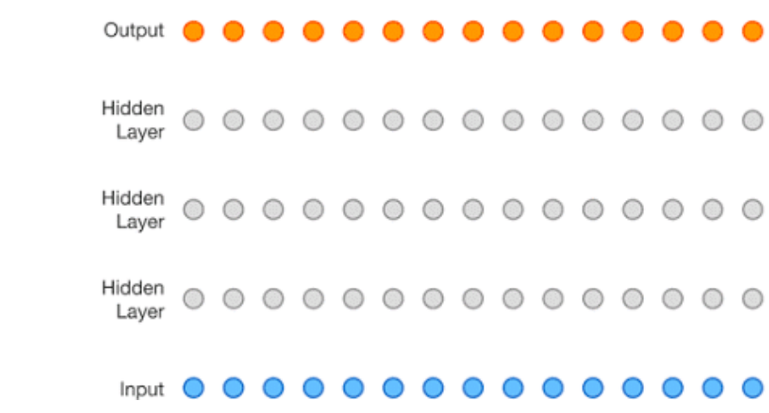
Generative Adversarial Networks use neural networks to perform a complicated change of variables.

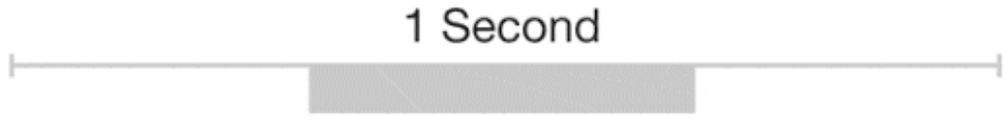
Implicit density for GAN is intractable, hence need for adversary

Autoregressive models defined by

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, \ldots, x_1).$$

have a tractable density. Train via maximum likelihood





GENERATIVE MODELS

arXiv:1712.10321
arXiv:1705.02355
DOI: 10.5281/zenodo.198541

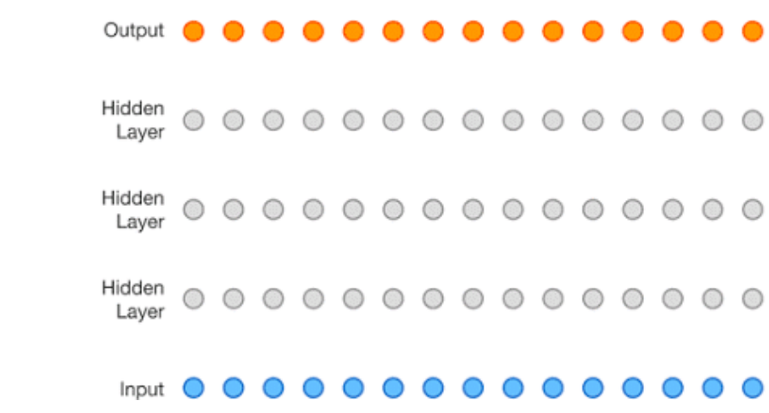
Generative Adversarial Networks use neural networks to perform a complicated change of variables.

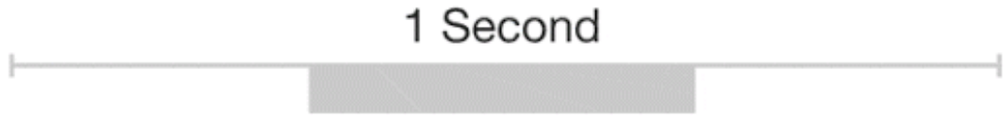
Implicit density for GAN is intractable, hence need for adversary

Autoregressive models defined by

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, \ldots, x_1).$$

have a tractable density. Train via maximum likelihood



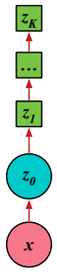


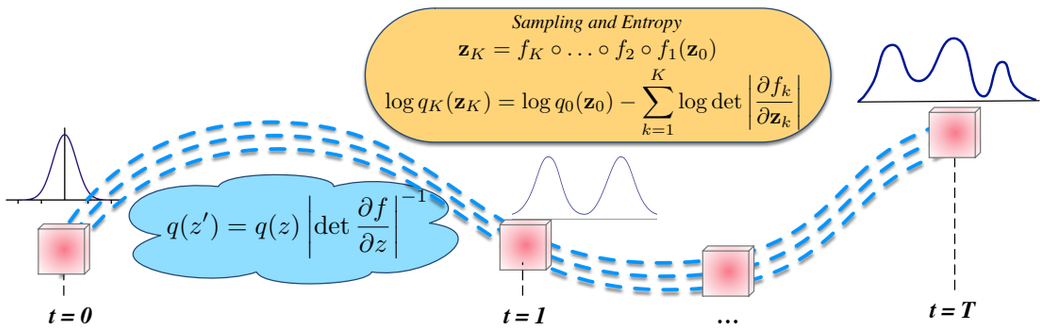
Normalizing flows and autoregressive models

Approximations using Change-of-variables

Exploit the rule for change of variables for random variables:

- Begin with an initial distribution $q_0(\mathbf{z}_0|\mathbf{x})$.
- Apply a sequence of K invertible functions f_k.





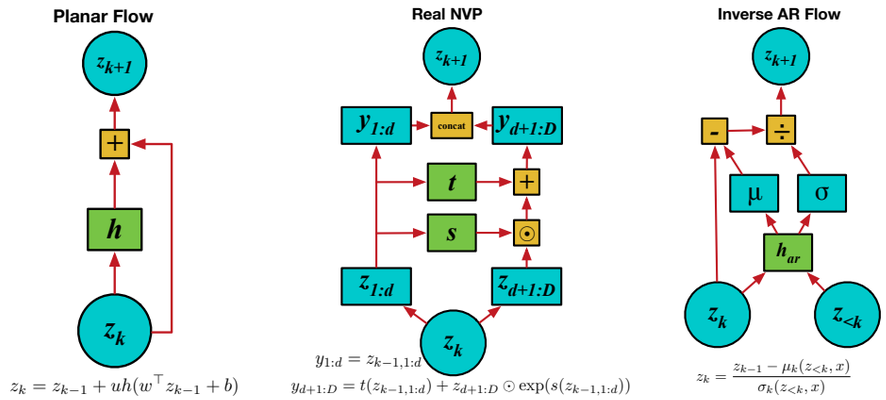
Distribution flows through a sequence of invertible transforms

[Rezende and Mohamed, 2015]

Choice of Transformation Function

$$\mathscr{L} = \mathbb{E}_{q_0(\mathbf{z}_0)}[\log p(\mathbf{x}, \mathbf{z}_K)] - \mathbb{E}_{q_0(\mathbf{z}_0)}[\log q_0(\mathbf{z}_0)] - \mathbb{E}_{q_0(\mathbf{z}_0)}\left[\sum_{k=1}^{K} \log \det \left| \frac{\partial f_k}{\partial \mathbf{z}_k} \right|\right]$$

- Begin with a fully-factorised Gaussian and improve by change of variables.
- Triangular Jacobians allow for computational efficiency.



[Rezende and Mohamed, 2016; Dinh et al., 2016; Kingma et al., 2016]

Linear time computation of the determinant and its gradient.

We will hear more from George Papamakarios !

AN ALTERNATIVE APPROACH

Unifying generative models and exact likelihood-free inference with conditional bijections

By Kyle Cranmer, Gilles Louppe

machine learning | likelihood-free inference | density estimation

Recent work in density estimation uses a bijection $f : X \to Z$ (e.g. an invertible flow or autoregressive model) and a tractable density $p(z)$ (e.g. [1] [2] [3] [4]).

$$p(x) = p(f_\phi(x)) \left| \det \left(\frac{\partial f_\phi(x)}{\partial x_T} \right) \right| ,$$

where ϕ are the internal network parameters for the bijection f_ϕ. Learning proceeds via gradient ascent $\nabla_\phi \sum_i \log p(x_i)$ with data x_i (i.e. maximum likelihood wrt. the internal parameters ϕ). Since f is invertible, then this model can also be used as a generative model for X.

This can be generalized to the conditional density $p(x|\theta)$ by utilizing a family of bijections $f_\theta : X \to Z$ parametrized by θ (e.g. [5] [6]).

$$p(x|\theta) = p(f_{\phi;\theta}(x)) \left| \det \left(\frac{\partial f_{\phi;\theta}(x)}{\partial x_T} \right) \right|$$

Here θ and x are input to the network (and its inverse) and ϕ are internal network parameters. Again, learning proceeds via gradient ascent $\nabla_\phi \sum_i \log p(x_i|\theta_i)$ with data x_i, θ_i.

We observe that not only can this model be used as a conditional generative model $p(x|\theta)$, but it can also be used to perform asymptotically exact, amortized likelihood-free inference on θ.

This is particularly interesting when θ is identified with the parameters of an intractable, non-differentiable computer simulation or the conditions of some real world data collection process.

Comments

Many thanks to Durk Kingma, Max Welling, Ian Goodfellow, and Shakir Mohamed for enlightening discussions at NIPS2016.

Kyle Cranmer · 9 Dec, 2016

Authors

Kyle Cranmer, Gilles Louppe

Metadata

DOI https:/doi.org/10.5281/zenodo.198541

Published: 8 Dec, 2016

Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation

George Papamakarios
School of Informatics
University of Edinburgh
g.papamakarios@ed.ac.uk

Iain Murray
School of Informatics
University of Edinburgh
i.murray@ed.ac.uk

2.2 Learning the posterior

Rather than using simulations from the model in order to estimate an approximate likelihood, $p(\|\mathbf{x} - \mathbf{x}_o\| < \epsilon \,|\, \boldsymbol{\theta})$, we will use the simulations to directly estimate $p(\boldsymbol{\theta} \,|\, \mathbf{x} = \mathbf{x}_o)$. We will run simulations for parameters drawn from a distribution, $\tilde{p}(\boldsymbol{\theta})$, which we shall refer to as the *proposal prior*. The proposition below indicates how we can then form a consistent estimate of the exact posterior, using a flexible family of conditional densities, $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x})$, parameterized by a vector $\boldsymbol{\phi}$.

Proposition 1. *We assume that each of a set of N pairs $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ was independently generated by*

$$\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta}) \quad \text{and} \quad \mathbf{x}_n \sim p(\mathbf{x} \,|\, \boldsymbol{\theta}_n). \tag{1}$$

In the limit $N \to \infty$, the probability of the parameter vectors $\prod_n q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_n \,|\, \mathbf{x}_n)$ is maximized w.r.t. $\boldsymbol{\phi}$ if and only if

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x}) \propto \frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\, p(\boldsymbol{\theta} \,|\, \mathbf{x}), \tag{2}$$

provided a setting of $\boldsymbol{\phi}$ that makes $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x})$ proportional to $\frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\, p(\boldsymbol{\theta} \,|\, \mathbf{x})$ exists.

Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows

George Papamakarios
University of Edinburgh

David C. Sterratt
University of Edinburgh

Iain Murray
University of Edinburgh

Abstract

We present Sequential Neural Likelihood (SNL), a new method for Bayesian inference in simulator models, where the likelihood is intractable but simulating data from the model is possible. SNL trains an autoregressive flow on simulated data in order to learn a model of the likelihood in the region of high posterior density. A sequential training procedure guides simulations and reduces simulation cost by orders of magnitude. We show that SNL is more robust, more accurate and requires less tuning than related neural-based methods, and we discuss diagnostics for assessing calibration, convergence and goodness-of-fit.

3 Sequential Neural Likelihood

Our new method, *Sequential Neural Likelihood (SNL)*, avoids the bias introduced by the proposal, by opting to learn a model of the likelihood instead of the posterior.

MNRAS **000**, 1–20 (2019) Preprint 4 March 2019 Compiled using MNRAS LATEX style file v3.0

Fast likelihood-free cosmology with neural density estimators and active learning

Justin Alsing,[1,2,3]★ Tom Charnock[4], Stephen Feeney[2] and Benjamin Wandelt[2,5]

[1] *Oskar Klein Centre for Cosmoparticle Physics, Stockholm University, Stockholm SE-106 91, Sweden*

[2] *Center for Computational Astrophysics, Flatiron Institute, 162 5th Ave, New York City, NY 10010, USA*

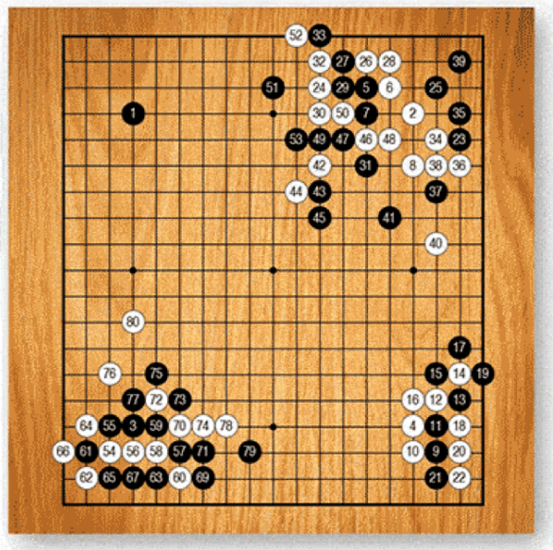
[3] *Imperial Centre for Inference and Cosmology, Department of Physics, Imperial College London, Blackett Laboratory, Prince Consort Road, London SW7 2AZ, UK*

[4] *Sorbonne Université, CNRS, UMR 7095, Institut dâĂŹAstrophysique de Paris, 98 bis bd Arago, 75014 Paris, France*

[5] *Sorbonne Université, Institut Lagrange de Paris (ILP), 98bis boulevard Arago, F-75014 Paris, France*

Active Sciencing

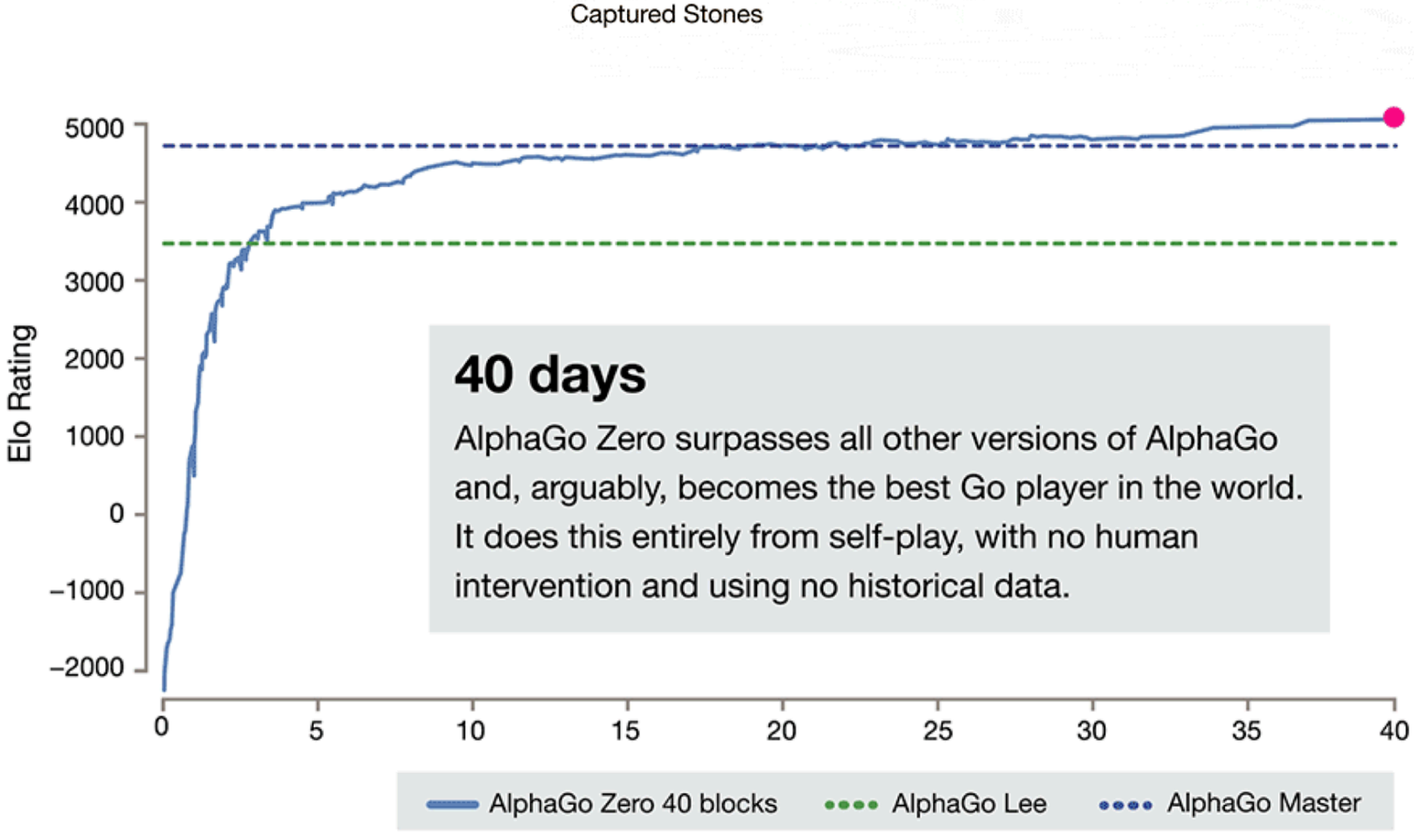
REINFORCEMENT LEARNING & SCIENTIFIC METHOD



68 at 61

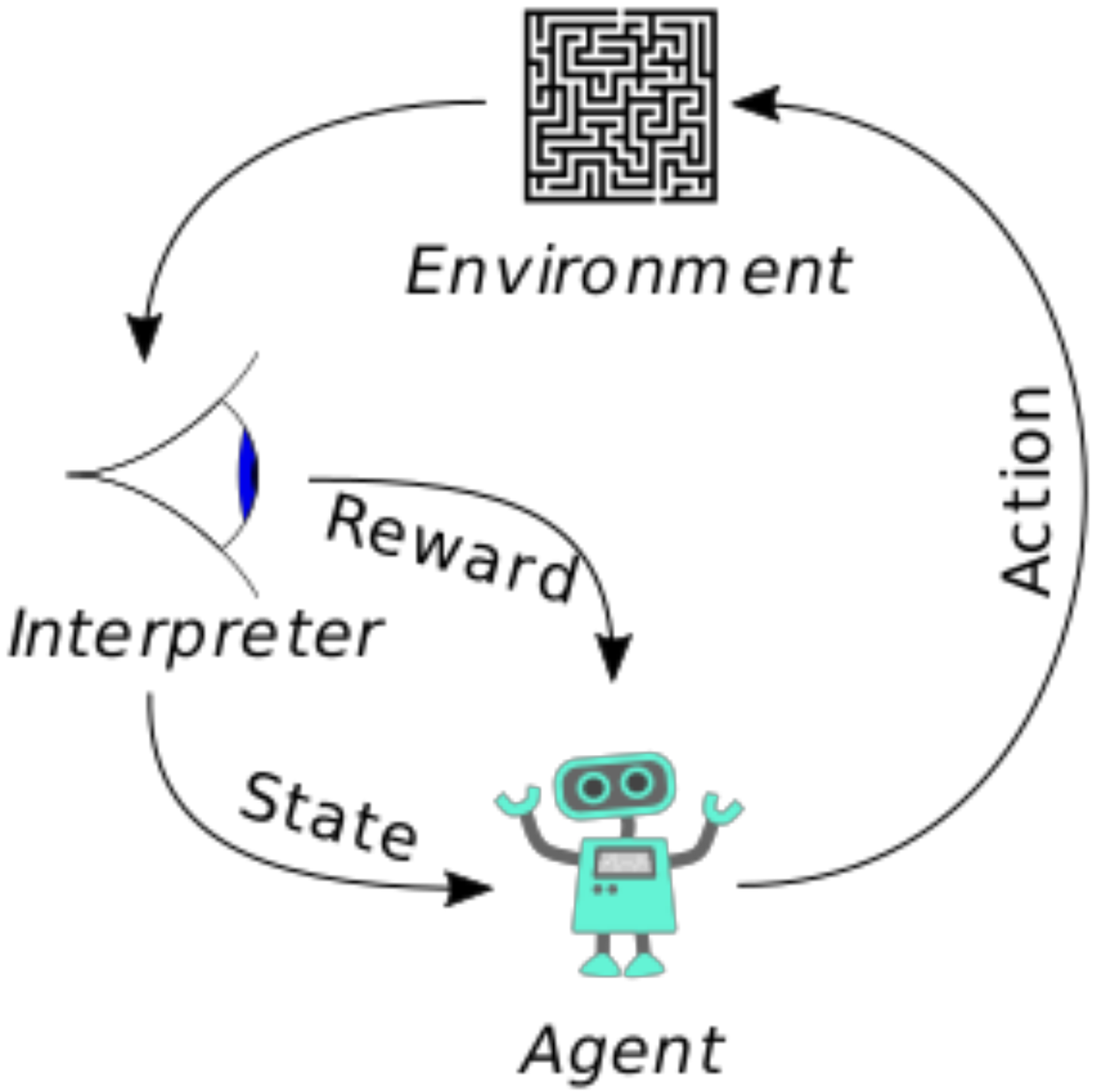
70 hours

AlphaGo Zero plays at super-human level. The game is disciplined and involves multiple challenges across the board.



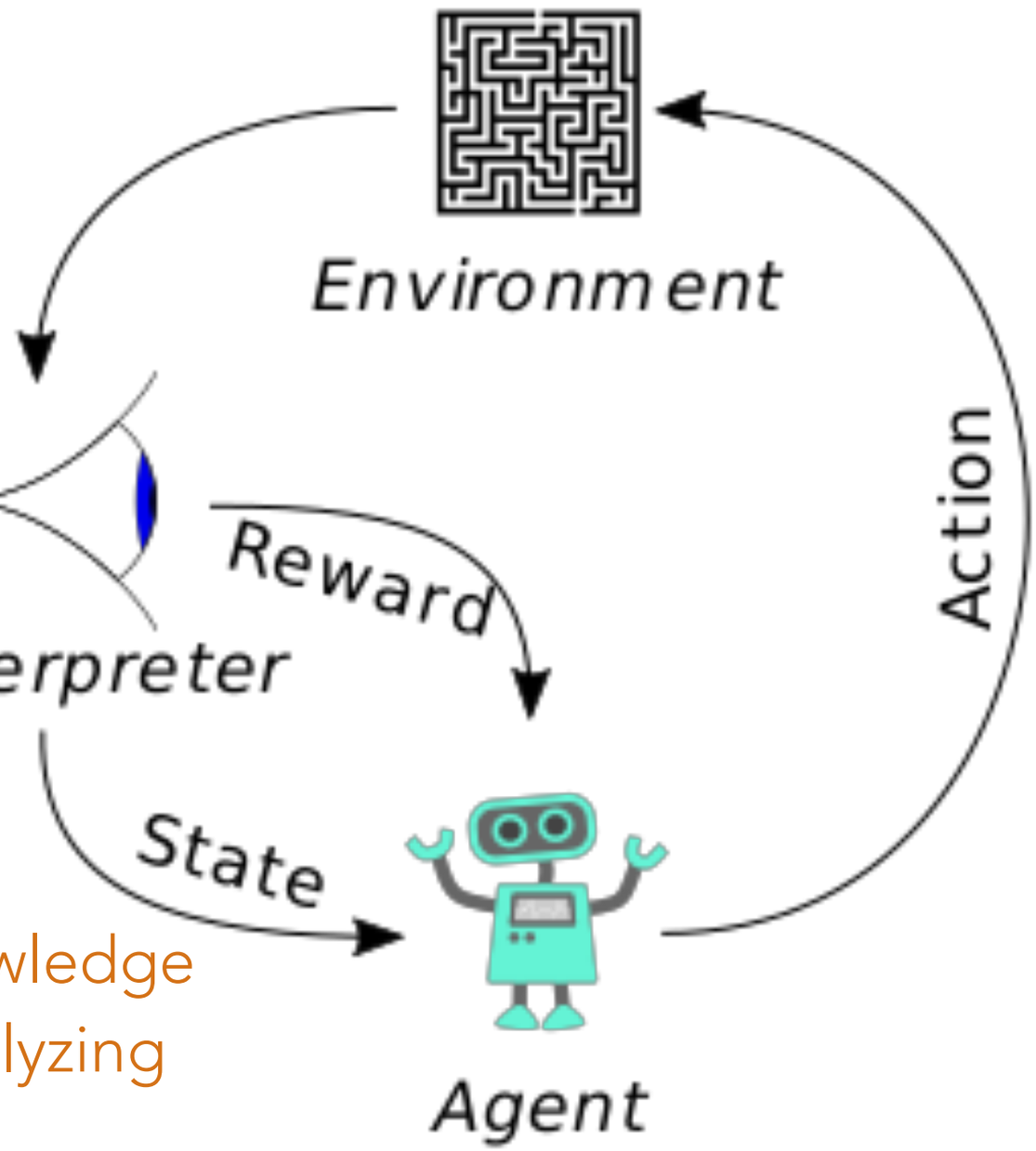
REINFORCEMENT LEARNING & SCIENTIFIC METHOD

Scientist trying to decide what experiment to do next



Scientist trying to decide what experiment to do next

perform experiment,
gather data



statistical analysis

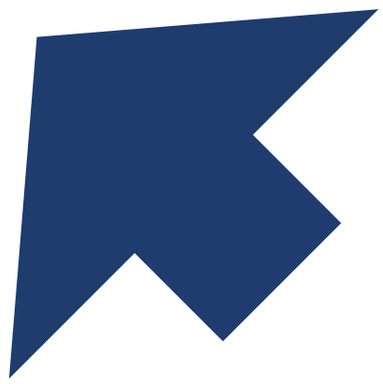
decide which experiment to perform

updated knowledge based on analyzing data

active learning

Active Sciencing

reusable workflows

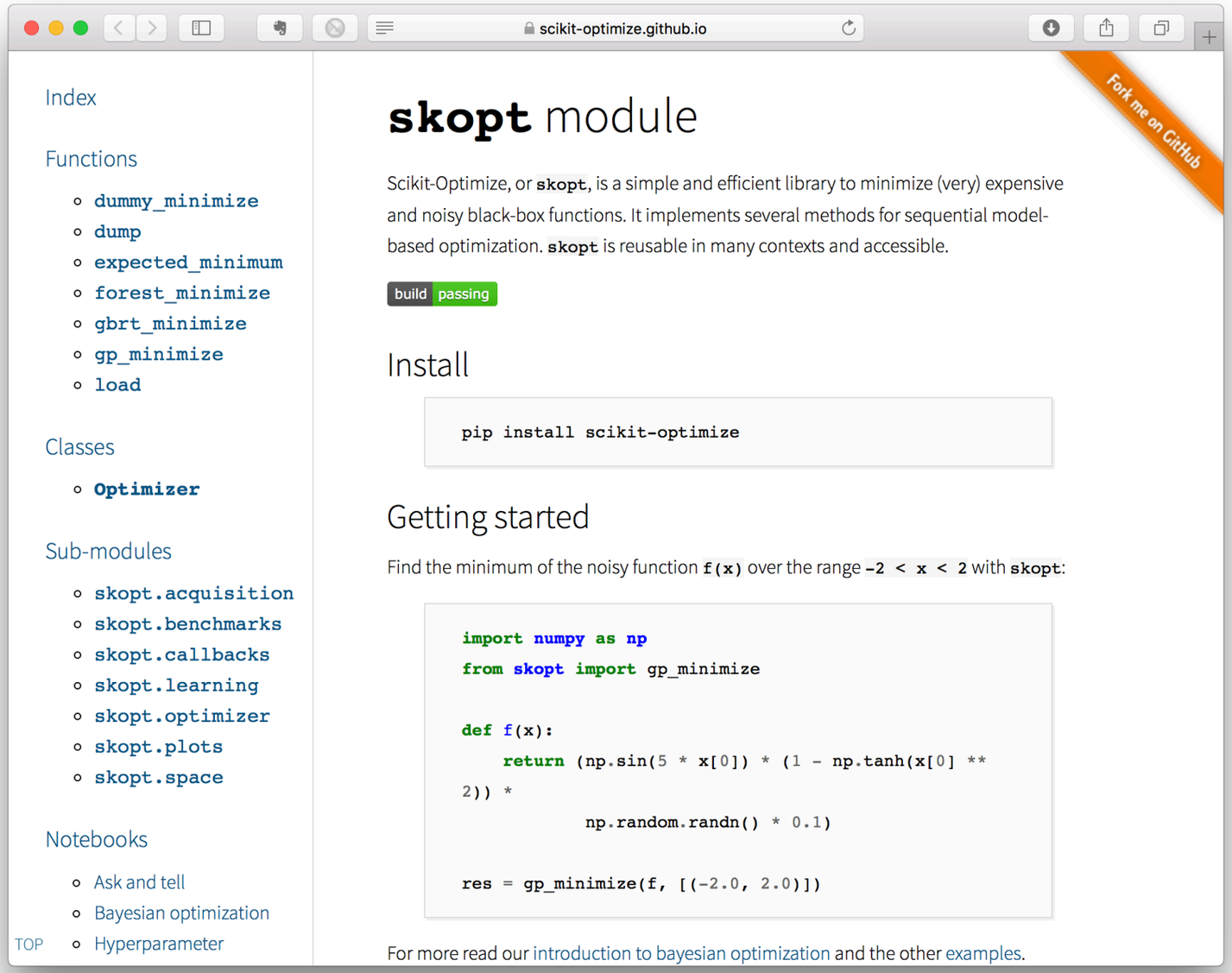


simulation-based
inference engines

https://github.com/cranmer/active_sciencing

OPTIMIZATION SOFTWARE

- Python
  - Spearmint https://github.com/JasperSnoek/spearmint
  - GPyOpt https://github.com/SheffieldML/GPyOpt
  - RoBO https://github.com/automl/RoBO
  - scikit-optimize https://github.com/MechCoder/scikit-optimize (work in progress)
- C++
  - MOE https://github.com/yelp/MOE



GitHub Repo for previous slides:

https://github.com/glouppe/talk-bayesian-optimisation

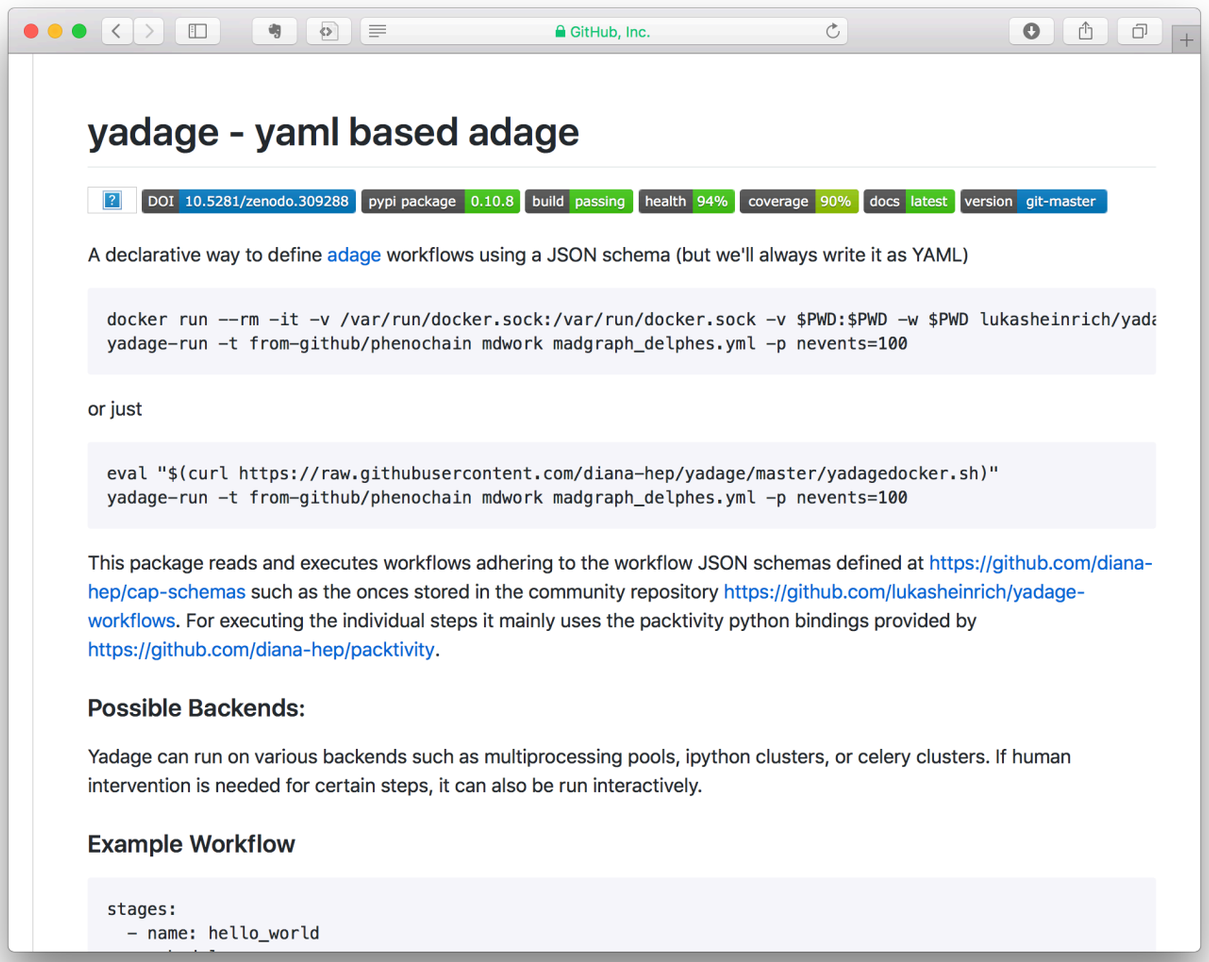
Yadage and Packtivity – analysis preservation using parametrized workflows

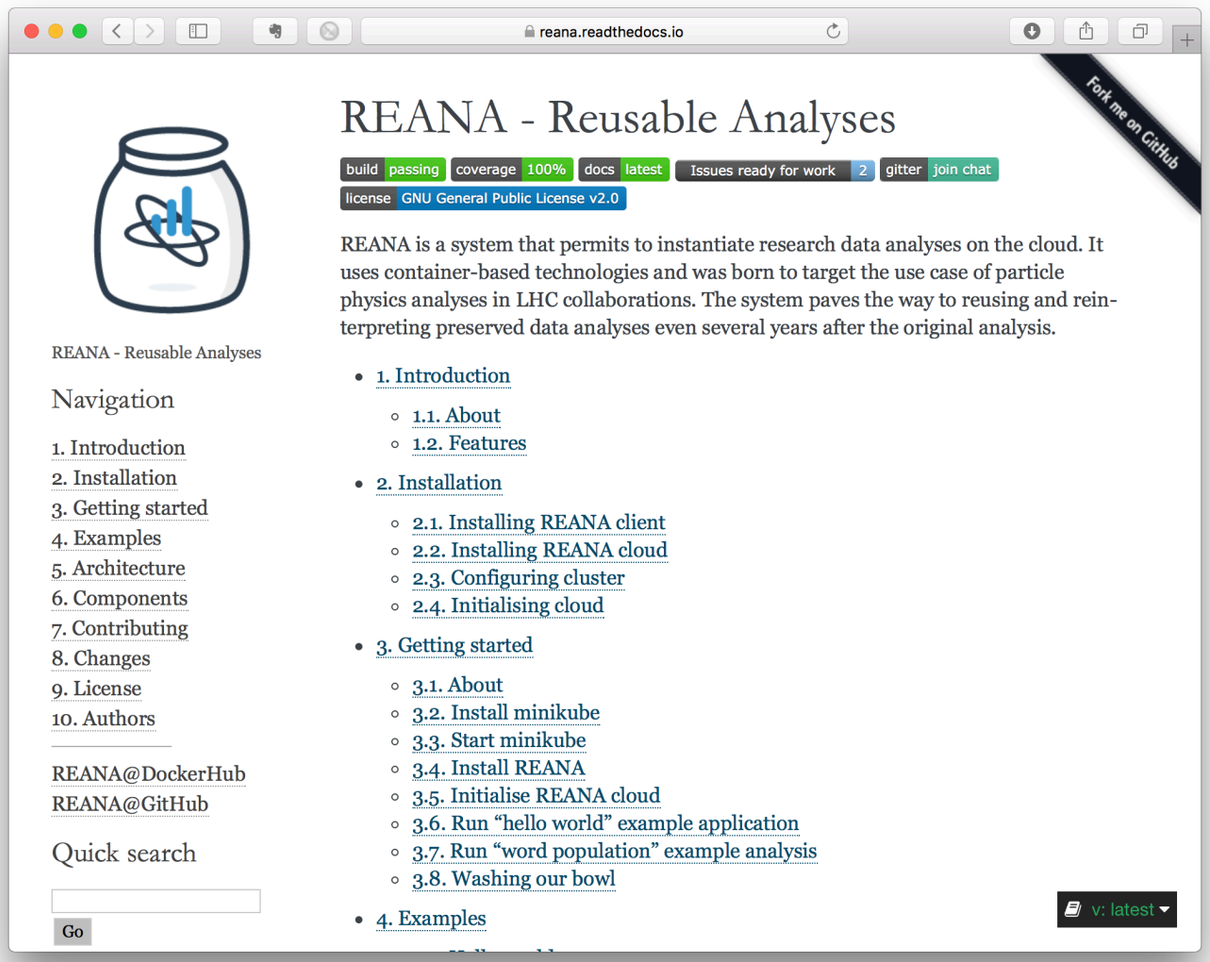
Kyle Cranmer[1] **and Lukas Heinrich**[1]

[1] Department of Physics, New York University, New York, USA

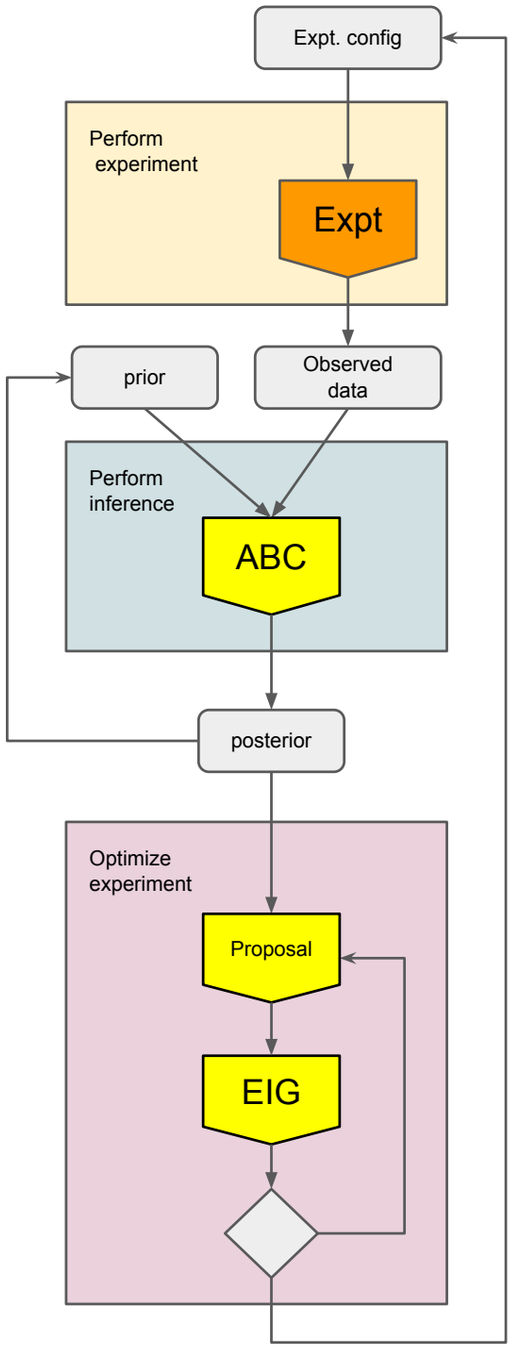
E-mail: `lukas.heinrich@cern.ch`

Abstract. Preserving data analyses produced by the collaborations at LHC in a parametrized fashion is crucial in order to maintain reproducibility and re-usability. We argue for a declarative description in terms of individual processing steps – "packtivities" – linked through a dynamic directed acyclic graph (DAG) and present an initial set of JSON schemas for such a description and an implementation – "yadage" – capable of executing workflows of analysis preserved via Linux containers.

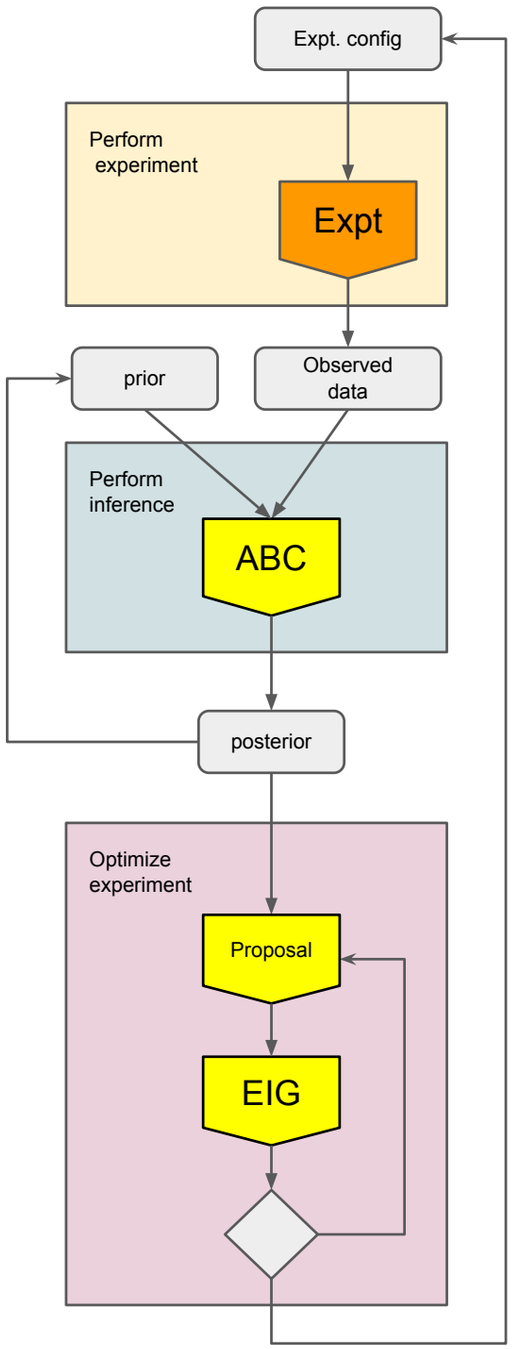


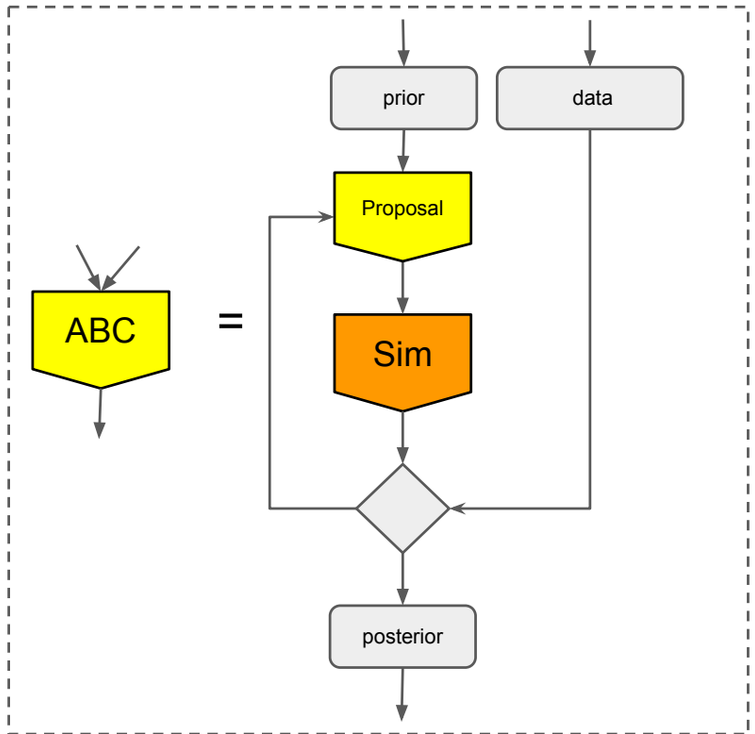


https://arxiv.org/pdf/1706.01878.pdf

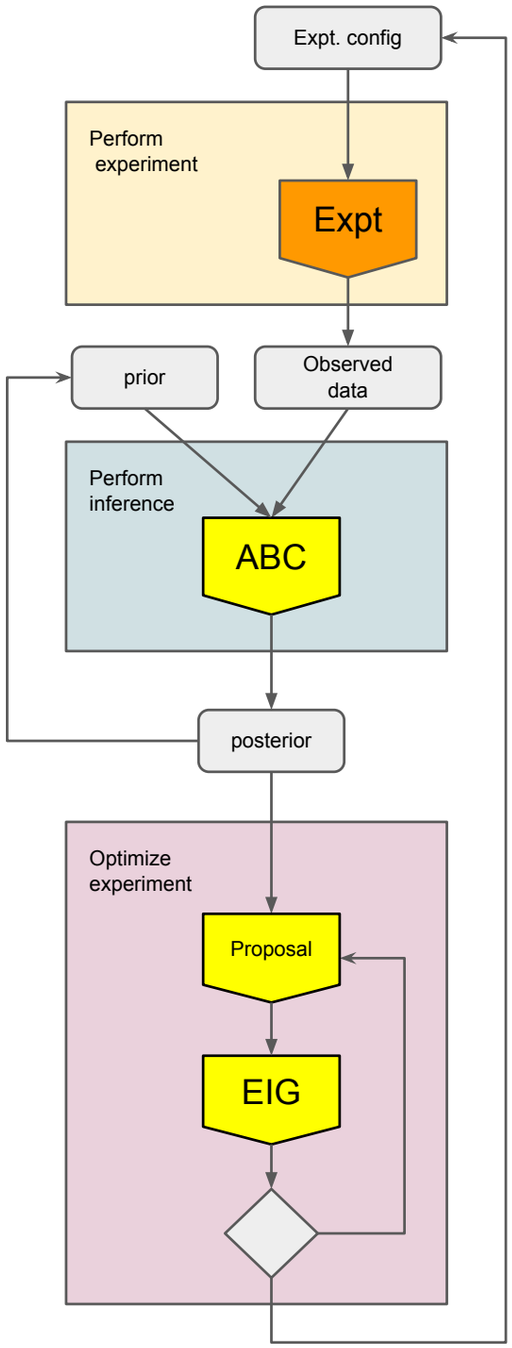


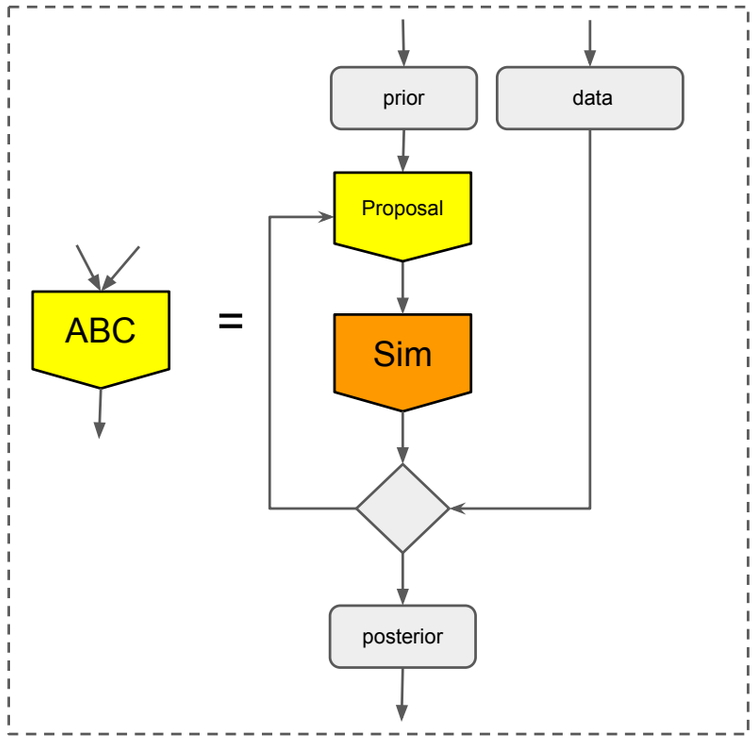
https://github.com/cranmer/active_sciencing (2017)

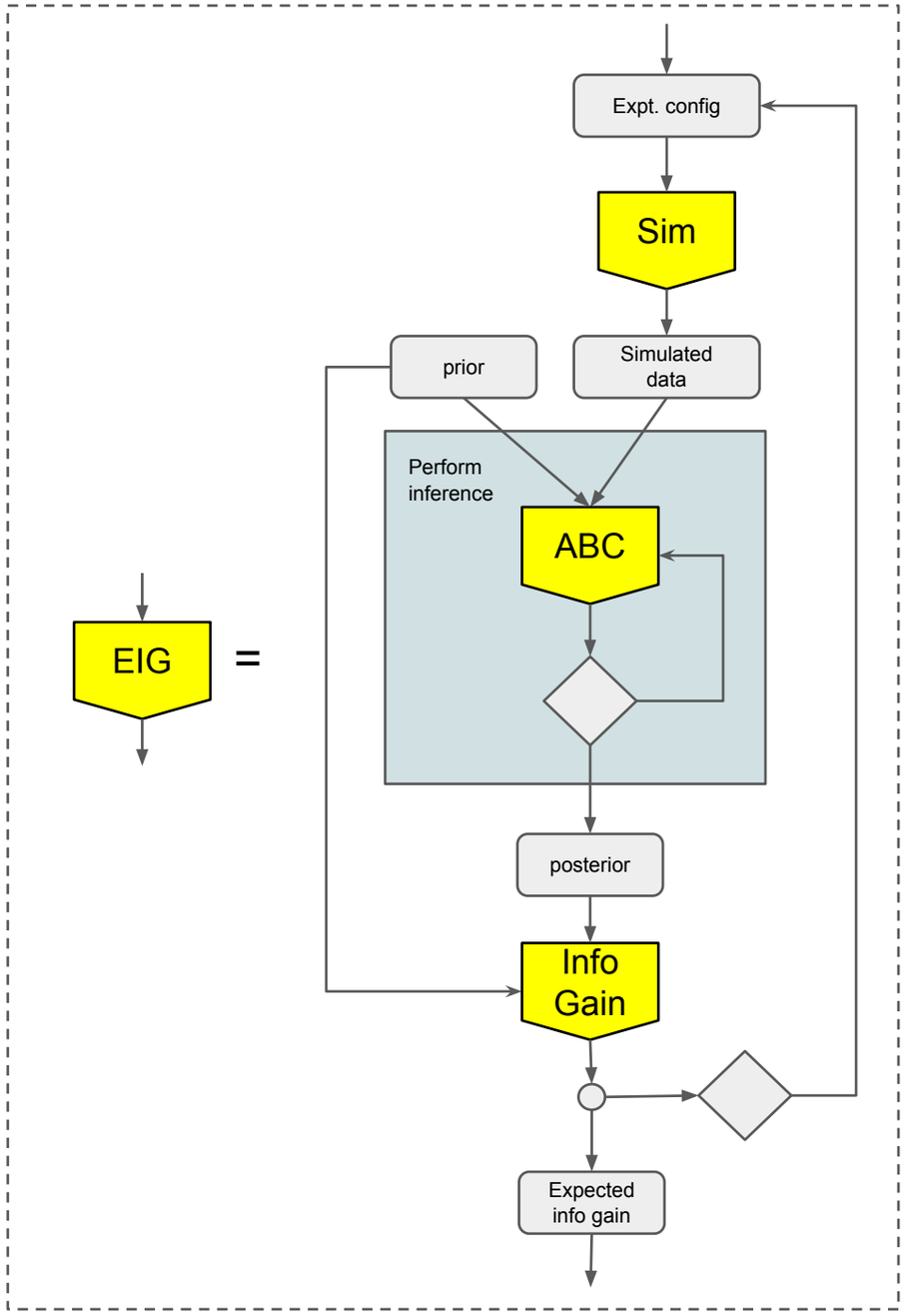




https://github.com/cranmer/active_sciencing (2017)







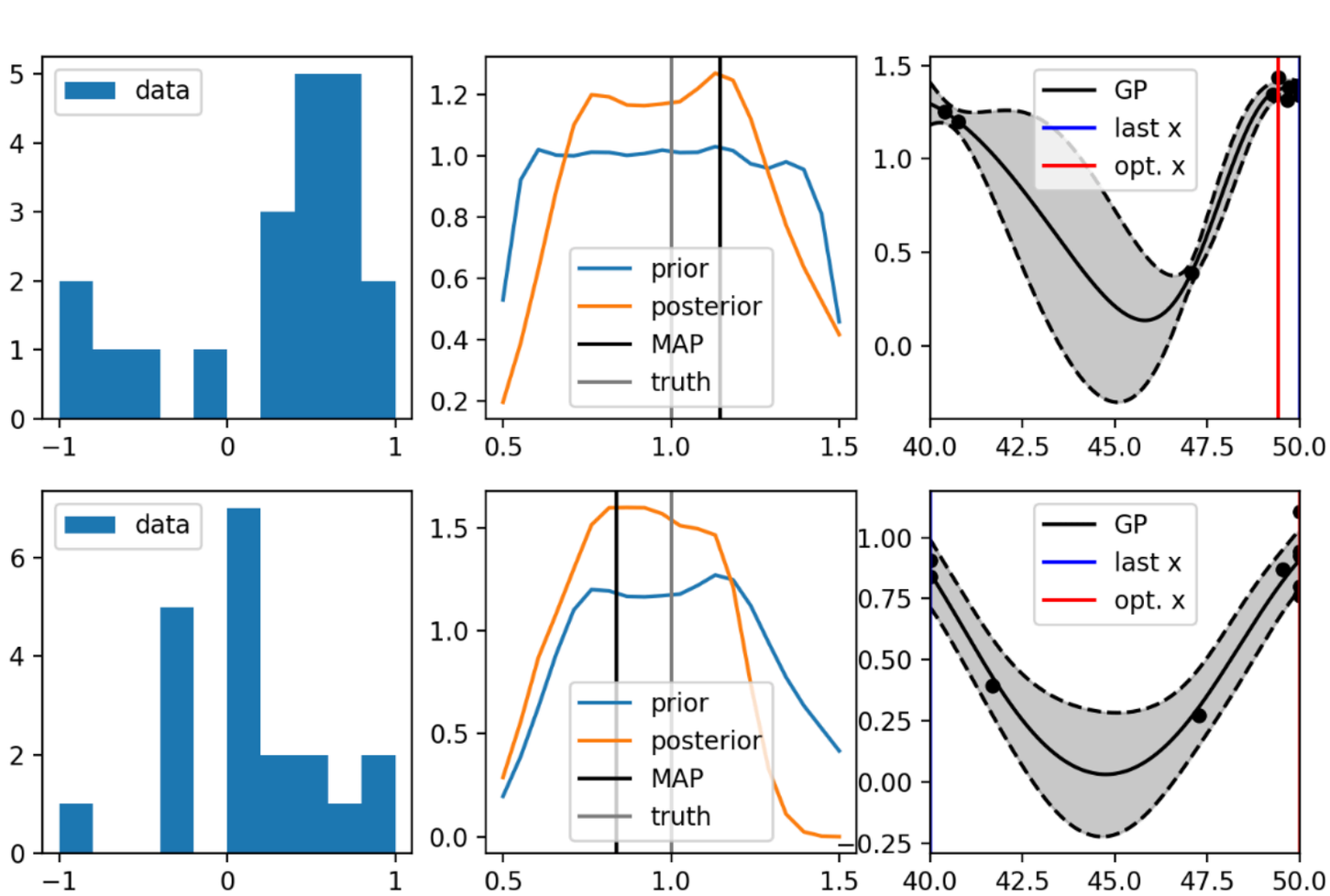
https://github.com/cranmer/active_sciencing (2017)

ACTIVE SCIENCING DEMO

Input:

- workflow for performing “real” experiment that returns data
- workflow for running simulator given parameters of theory and experimental configuration

Demo shows use of likelihood-free inference technique & Bayesian Optimization to measure the Weinberg angle and optimize beam energy (eg. just above or below $M_Z/2$)



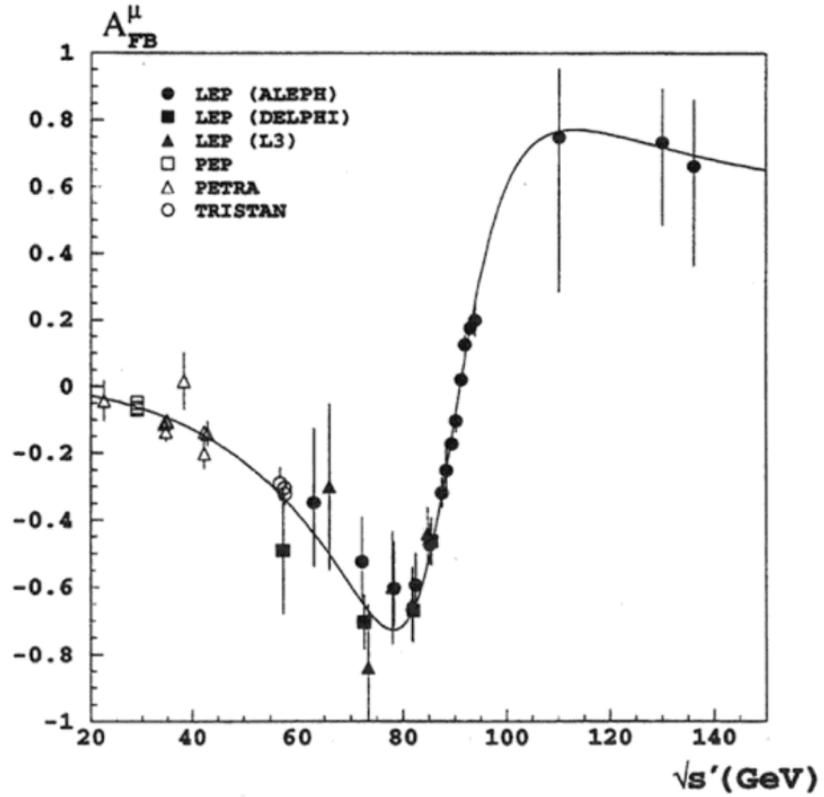


Figure 2: Measured forward-backward asymmetries of muon-pair production compared with the model independent fit results.

REMINDER TO SELF / COMMENTS

Amortized vs. not amortized

More on 2-sample approach (CARL, GAN, etc)

- Overlapping support
- Calibration

Local Model / Sufficient statistics

Point estimates only

- AVO, Recurrent Inference Machines

Universal ProbProg

- Deep interpretability,
- when “ABC likelihood” isn’t needed

Interpretability

TAKE AWAYS

Many areas of science have simulations based on some well-motivated mechanistic model.

However, the aggregate effect of many interactions between these low-level components leads to an intractable inverse problem.

The developments in machine learning and AI have the potential to effectively bridge the microscopic - macroscopic divide & aid in the inverse problem.

- they can provide effective statistical models that describe emergent macroscopic phenomena that are tied back to the low-level microscopic (reductionist) model

- generative models and likelihood-free inference are two particularly exciting areas

CONCLUSIONS

Our understanding of how to leverage our prior physics knowledge while letting machine learning do what it's good at is maturing.

- build in robustness to systematic uncertainties
- ability to inject and extract physics knowledge from models
- exploit symmetries, hierarchical structure of data

Harnessing the full potential of these techniques will require deep integration into our scientific workflow

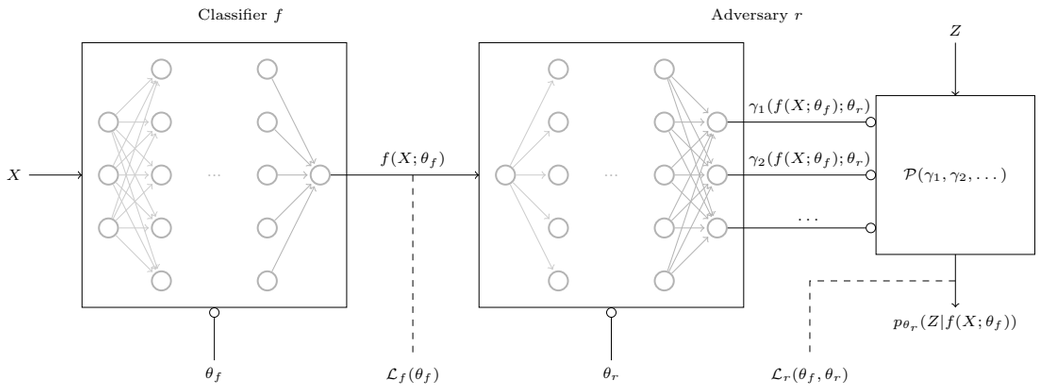
Example:

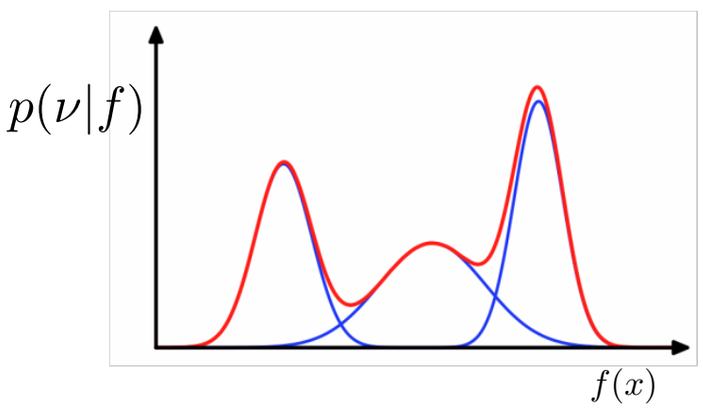
Systematics Uncertainty
Continuous Domain Adaptation
Fairness on Continuous Attributes

LEARNING TO PIVOT WITH ADVERSARIAL NETWORKS

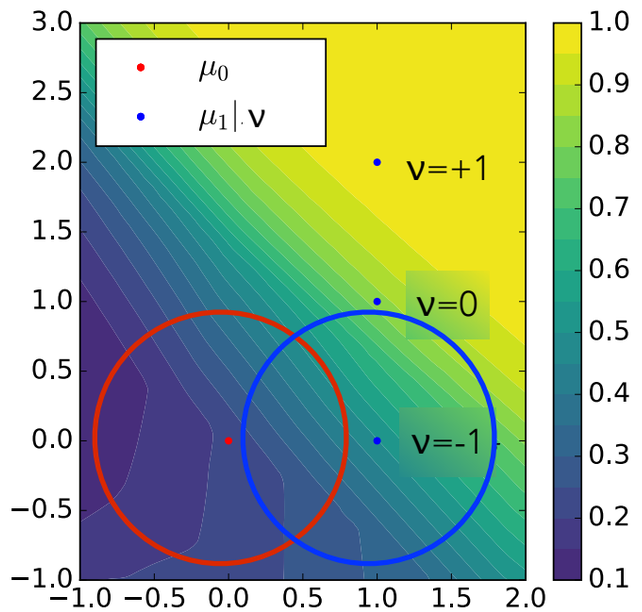
Typically classifier **f(x)** trained to minimize loss $\mathbf{L_f}$.

- want classifier output to be insensitive to systematics (nuisance parameter **ν**)
- introduce an **adversary r** that tries to predict ν based on f.
- setup as a minimax game:

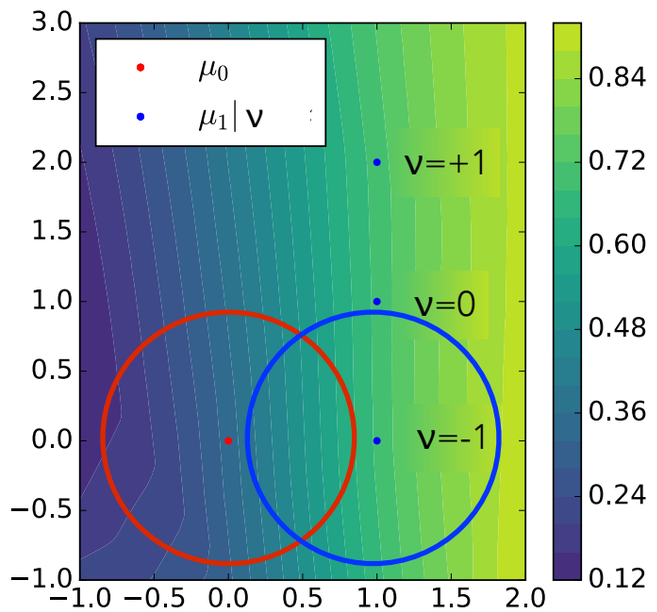


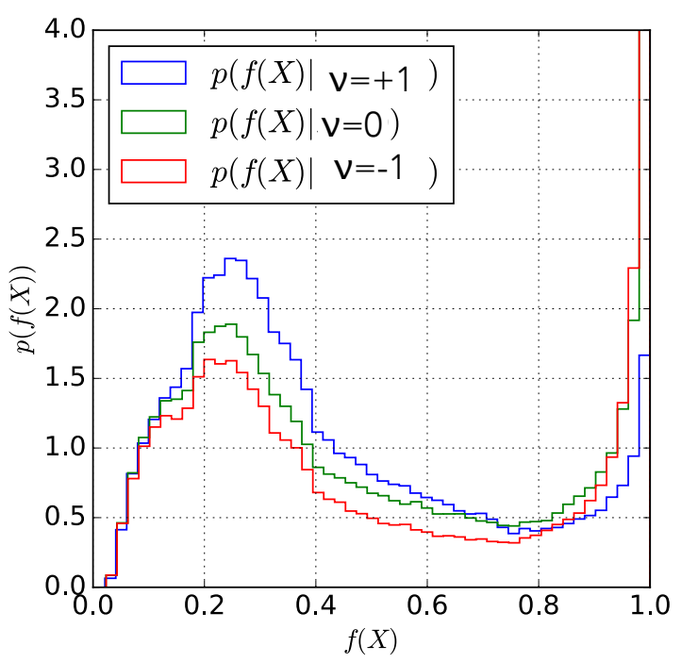


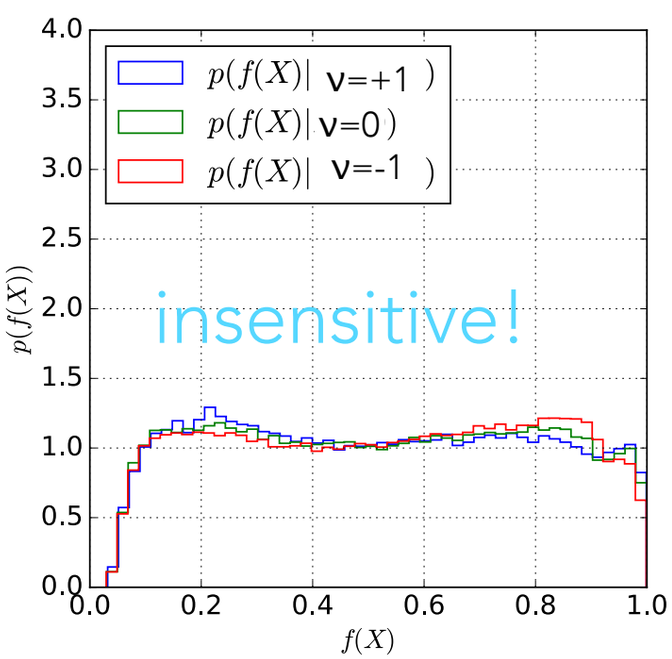
normal training



adversarial training





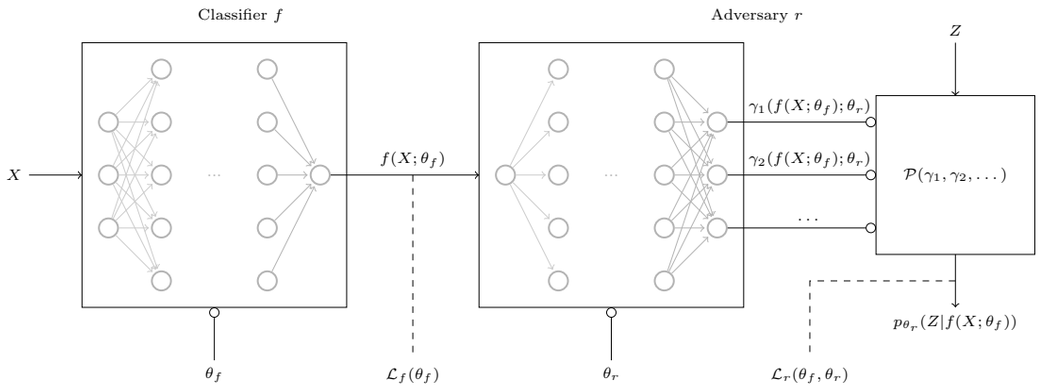


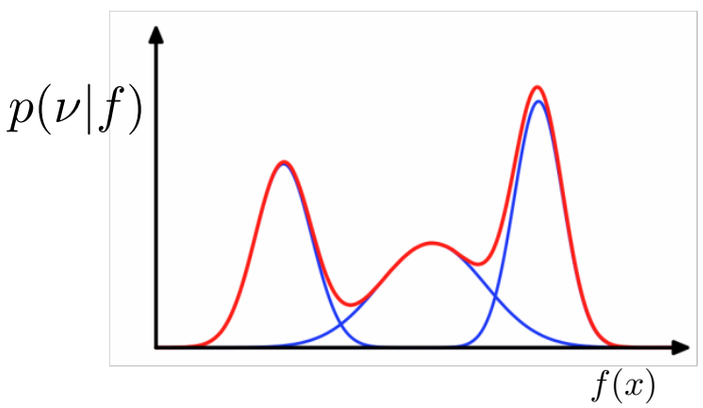
G. Louppe, M. Kagan, K. Cranmer, Learning to Pivot with Adversarial Networks [arXiv:1611.01046]

LEARNING TO PIVOT WITH ADVERSARIAL NETWORKS

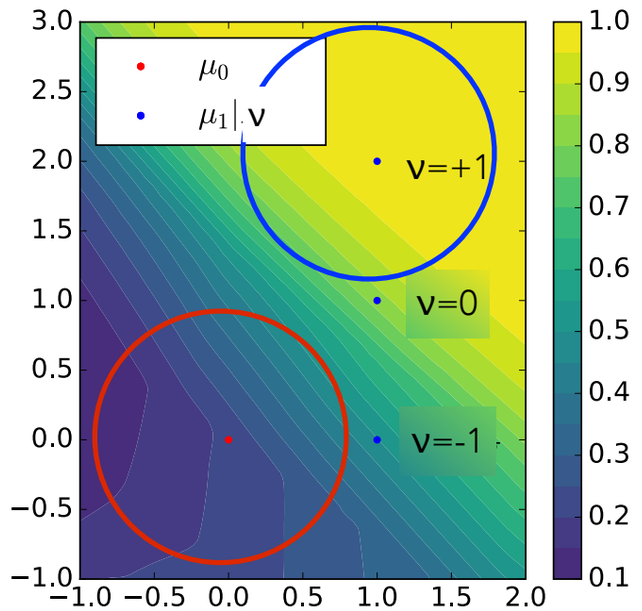
Typically classifier **f(x)** trained to minimize loss $\mathbf{L_f}$.

- want classifier output to be insensitive to systematics (nuisance parameter **ν**)
- introduce an **adversary r** that tries to predict ν based on f.
- setup as a minimax game:

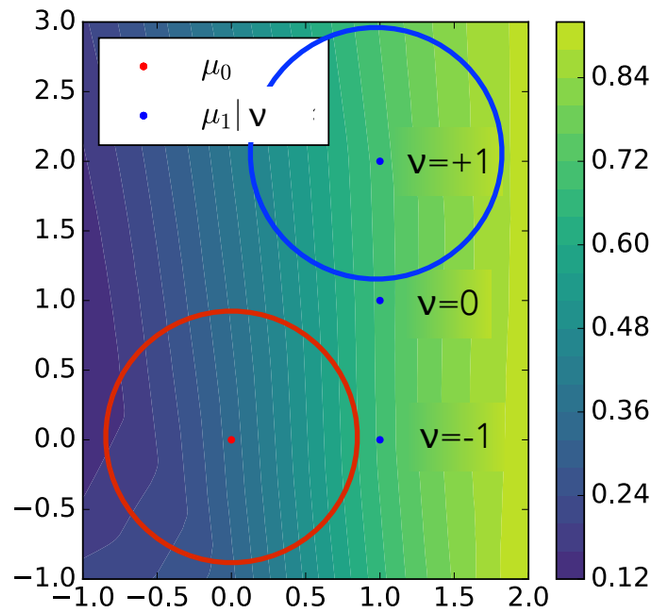


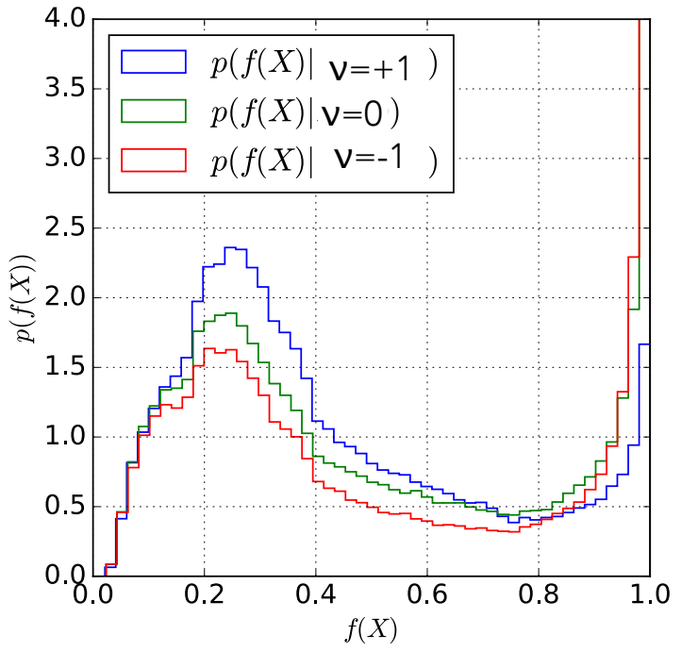


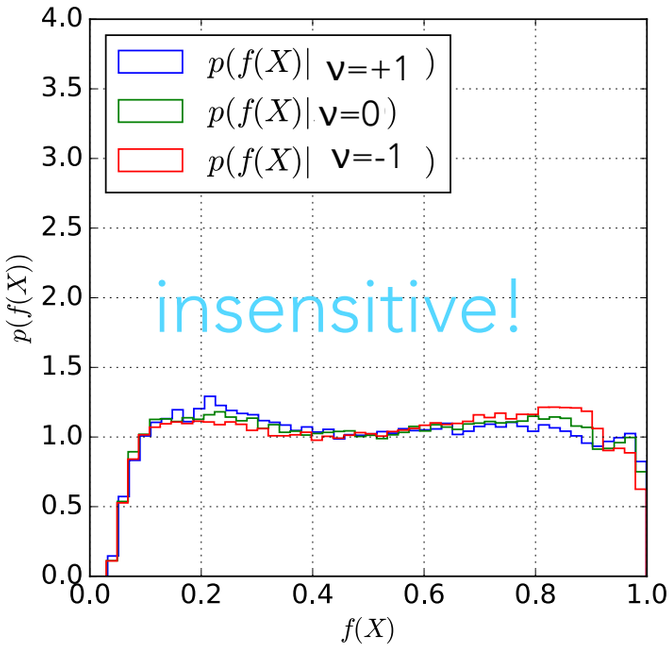
normal training



adversarial training



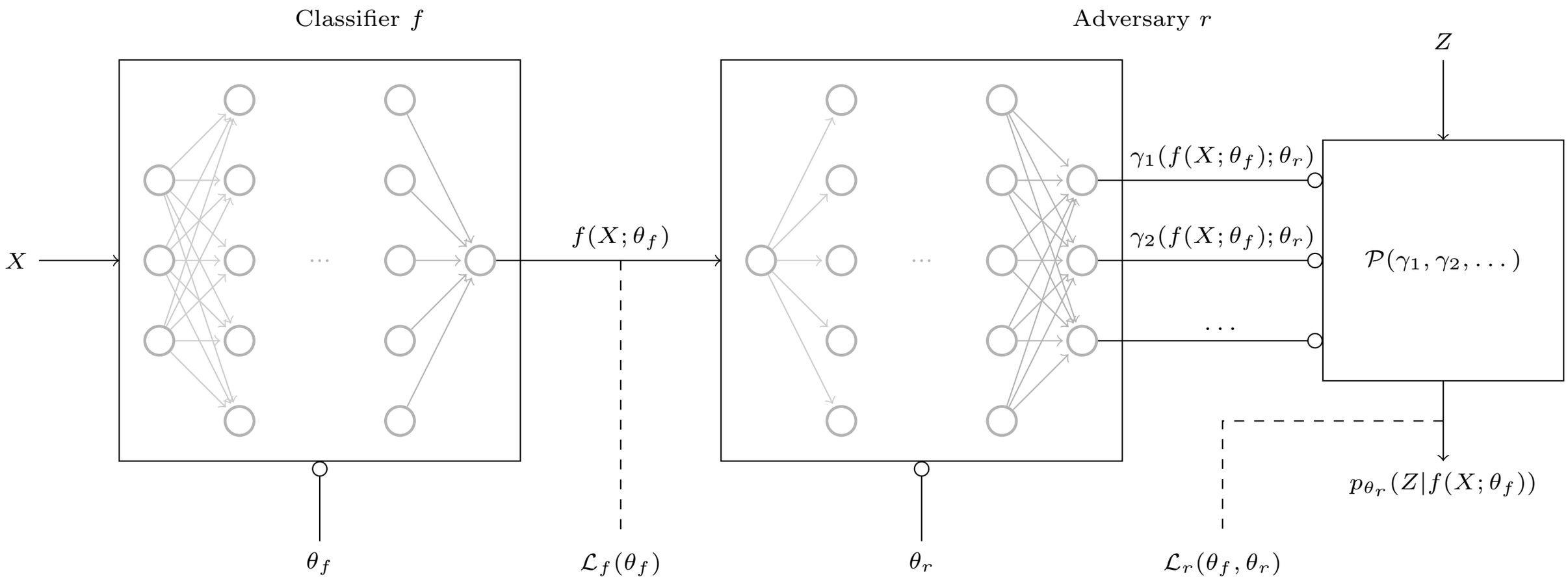




G. Louppe, M. Kagan, K. Cranmer, Learning to Pivot with Adversarial Networks [arXiv:1611.01046]

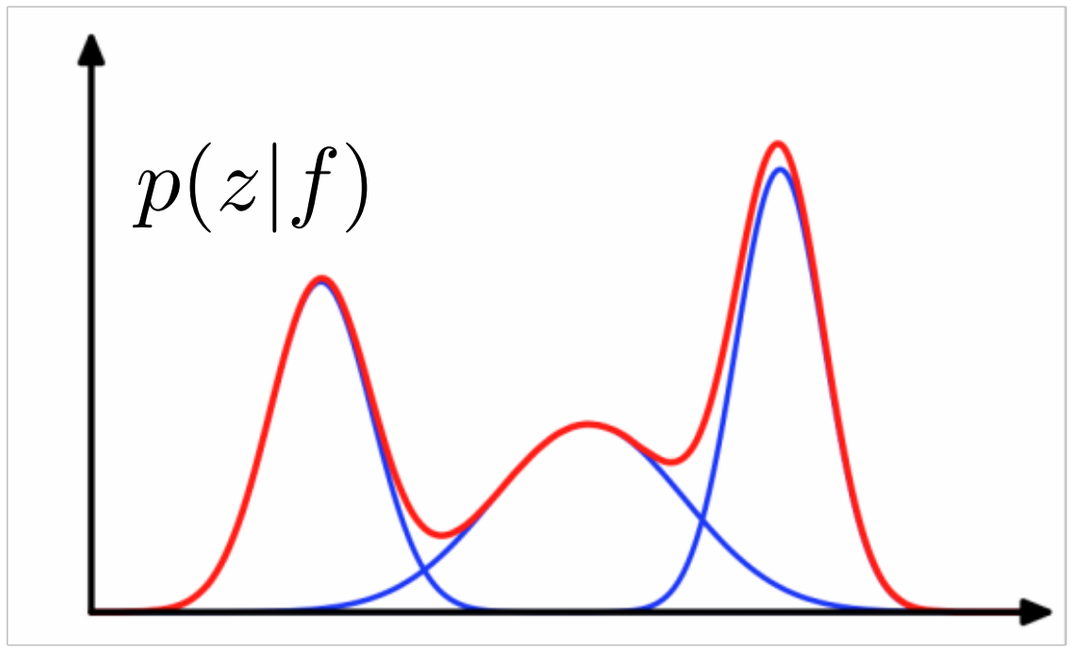
THE ADVERSARIAL MODEL

G. Louppe, M. Kagan, K. Cranmer, arXiv:1611.01046



the γ_1, γ_2, … are the mean, standard deviation, and amplitude for the Gaussian Mixture Model.

- the neural network takes in f and predicts γ_1, γ_2, …

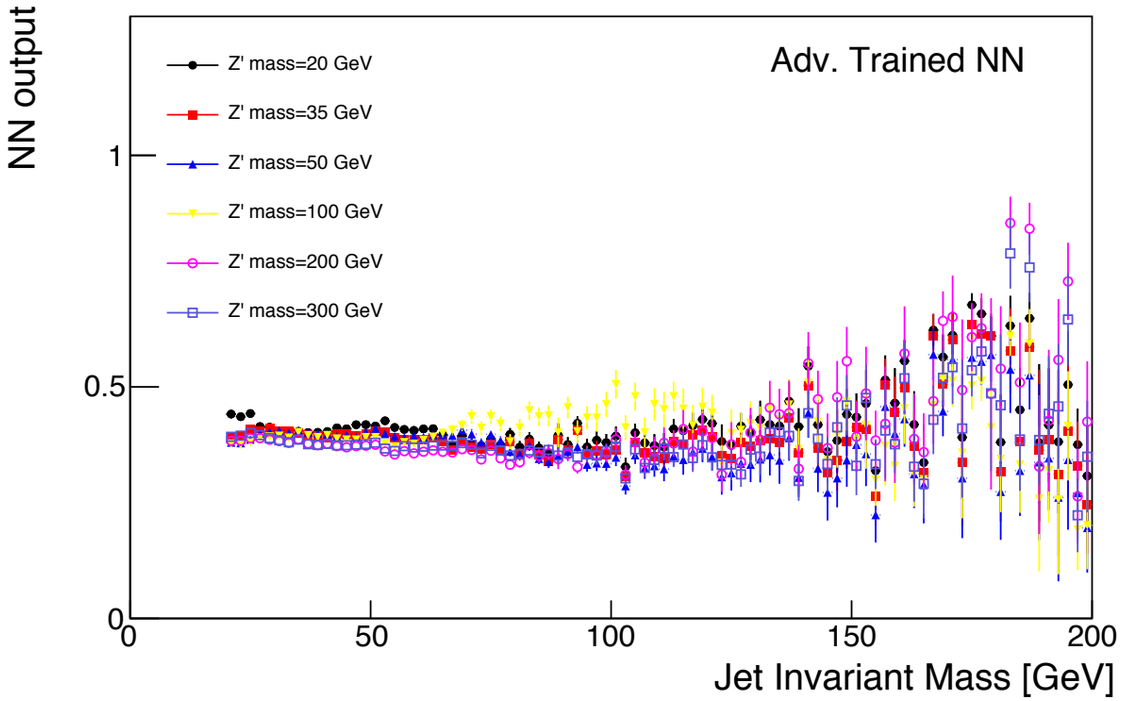


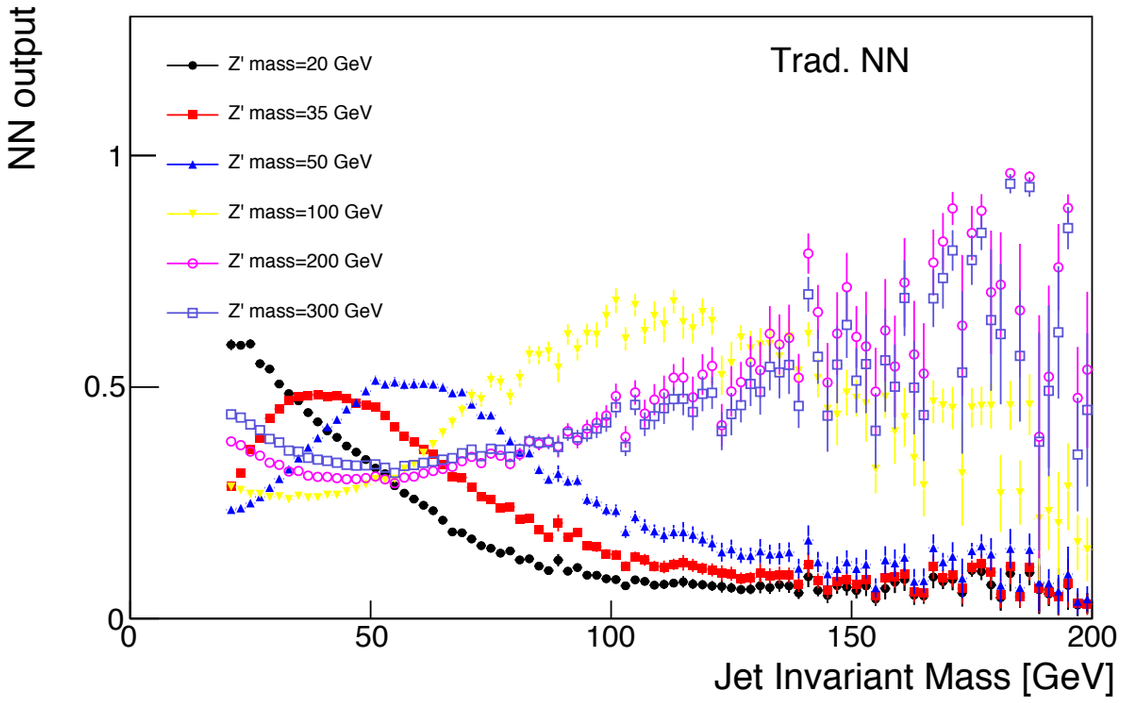
FAIR CLASSIFIERS

K.C, J. Pavez, and G. Louppe, arXiv:1506.02169
P. Baldi, K.C, T. Faucett, P. Sadowski, D. Whiteson arXiv:1601.07913
G. Louppe, M. Kagan, K.C, arXiv:1611.01046
Shimmin, et. al. arXiv:1703.03507

Adversarial approach of "Learning to Pivot" can also be used to train a classifier that is independent from some other continuous variable.

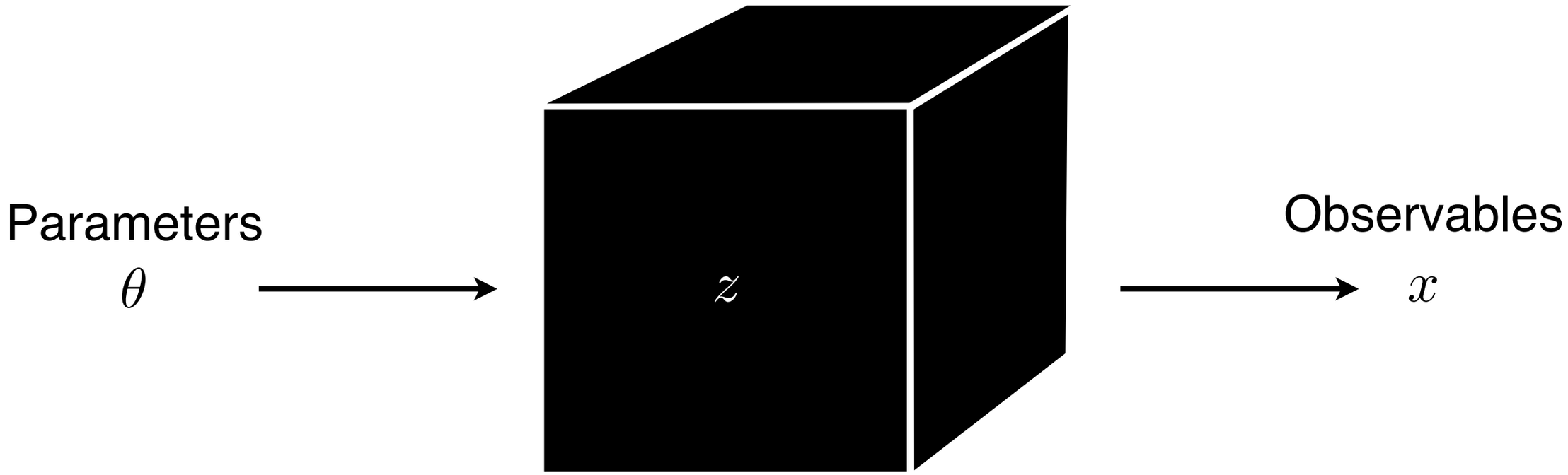
- fairness to continuous attribute
- motivation for doing this is related to robustnesss to uncertainties and interpretability





Other examples

LIKELIHOOD-FREE INFERENCE



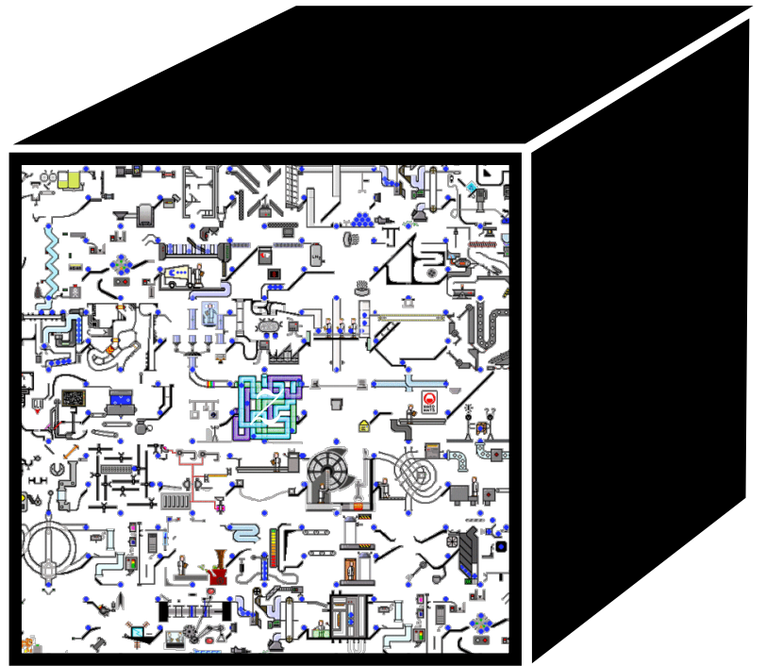
Prediction (simulation):

- Well-understood mechanistic model
- Simulator can generate samples

LIKELIHOOD-FREE INFERENCE

Parameters

θ



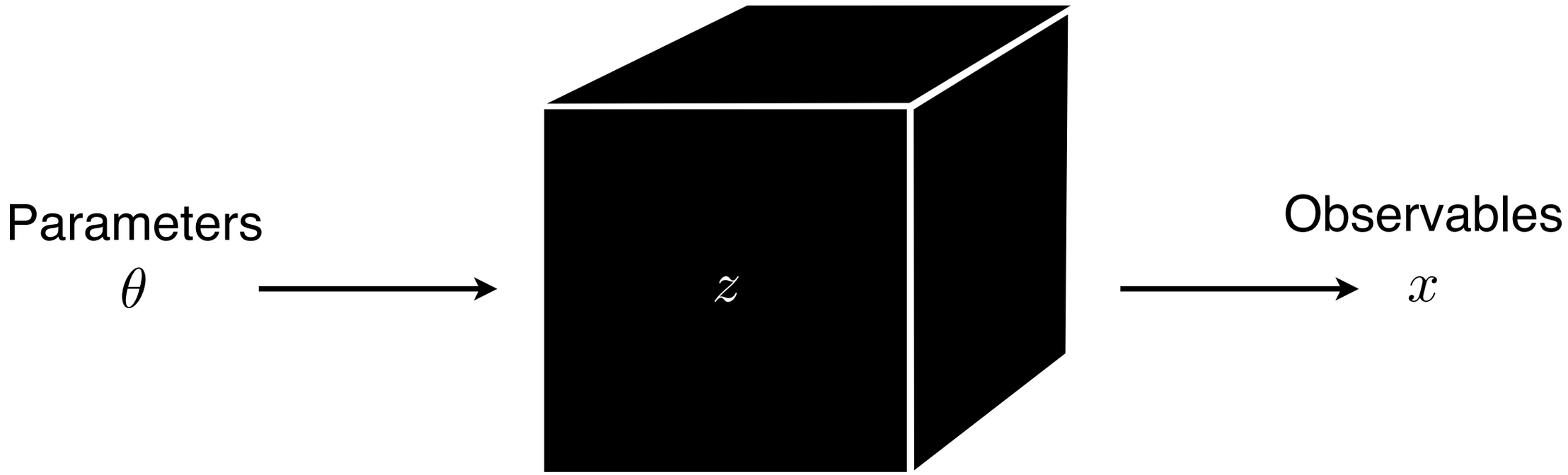
Observables

x

Prediction (simulation):

- Well-understood mechanistic model
- Simulator can generate samples

LIKELIHOOD-FREE INFERENCE



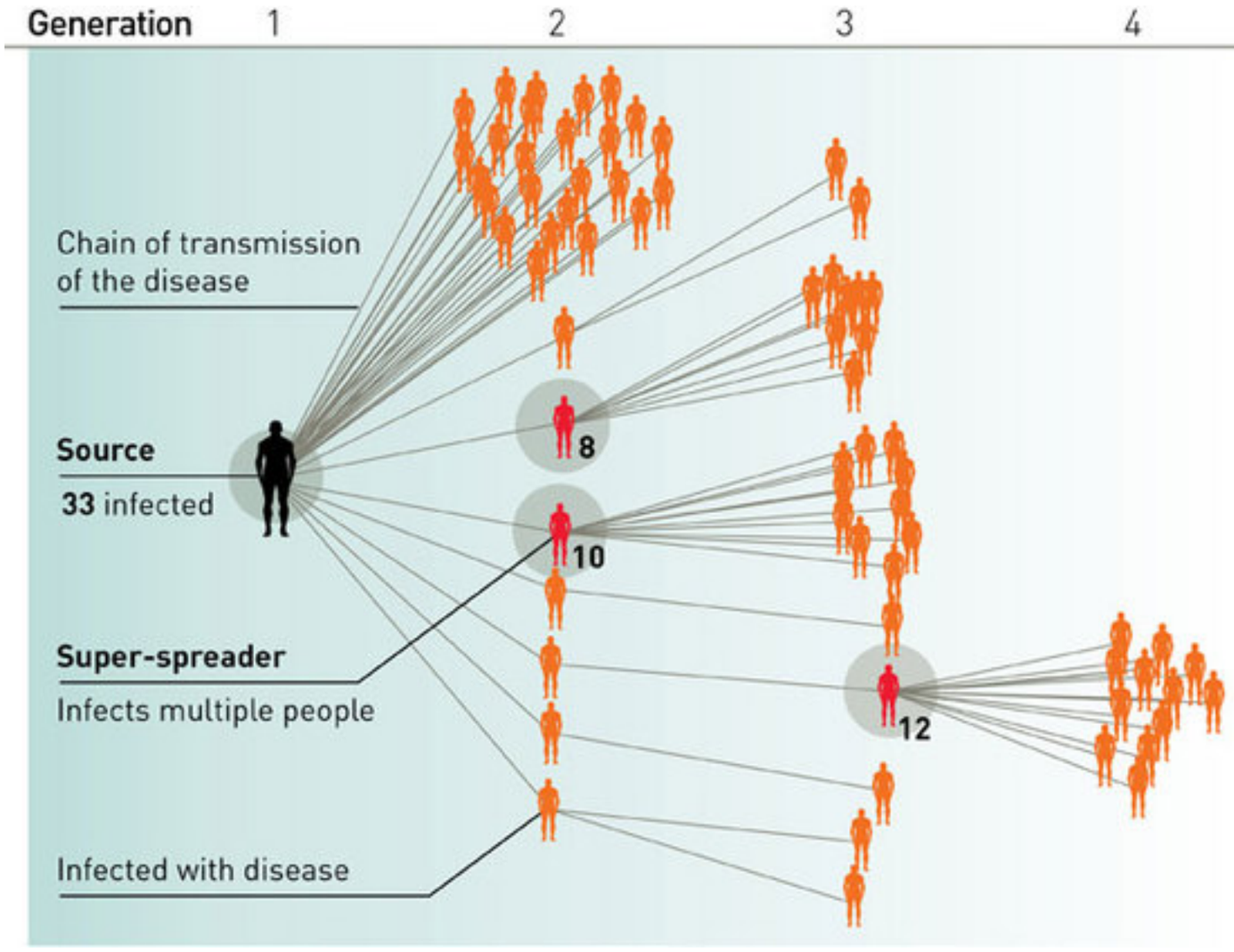
Prediction (simulation):

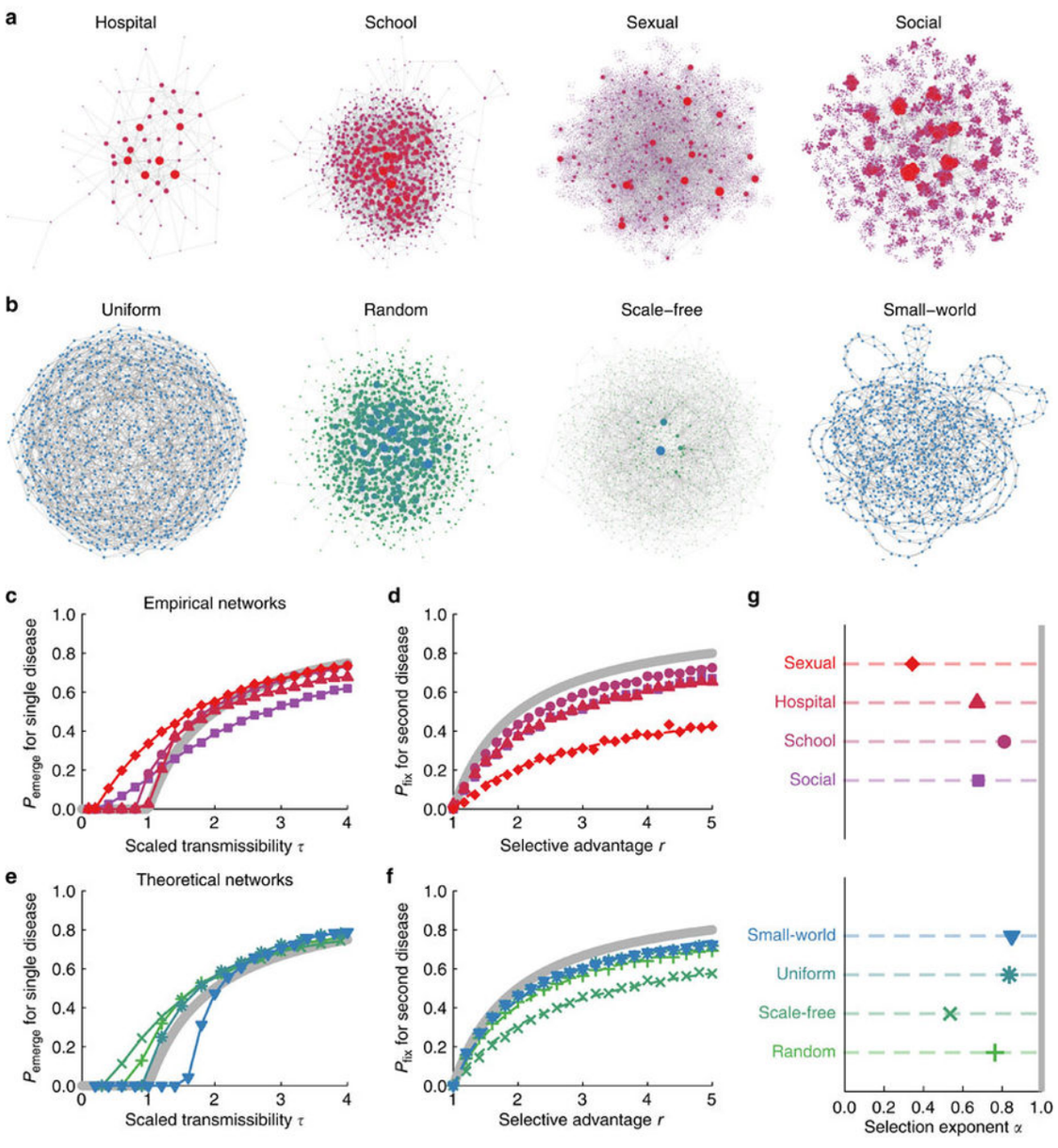
- Well-understood mechanistic model
- Simulator can generate samples

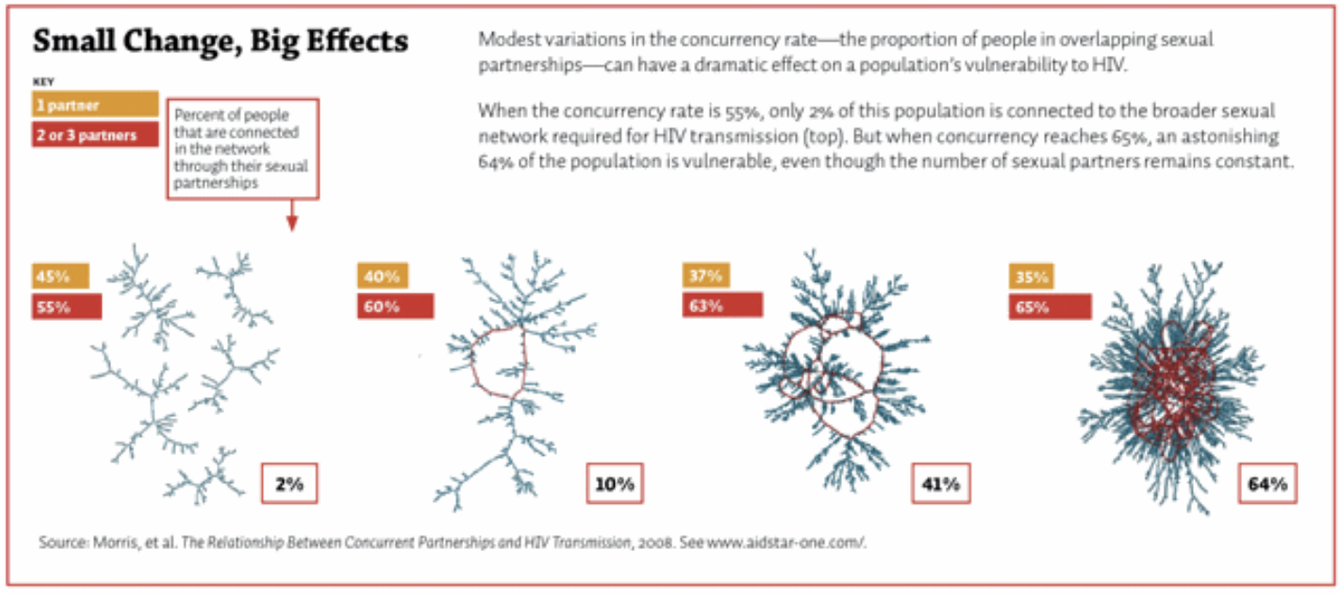
Inference:

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

EPIDEMIOLOGY & POPULATION GENETICS



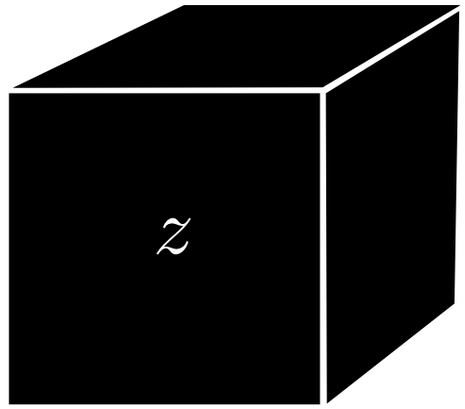




COSMOLOGICAL N-BODY SIMULATIONS

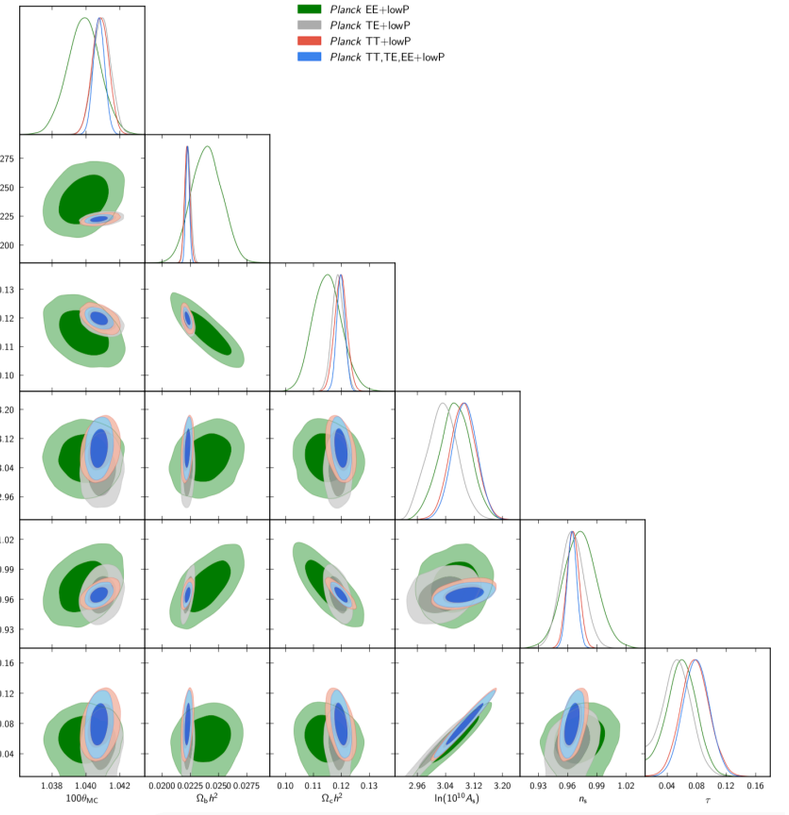
Parameters

θ

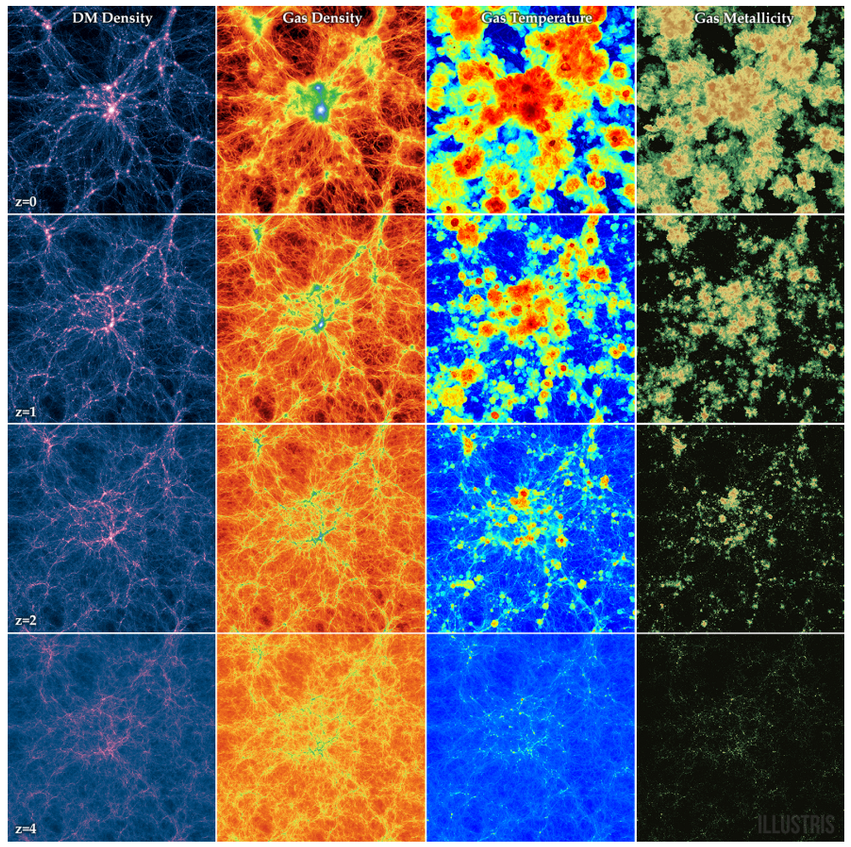


Observables

x



[Source: Planck 1502.01589]



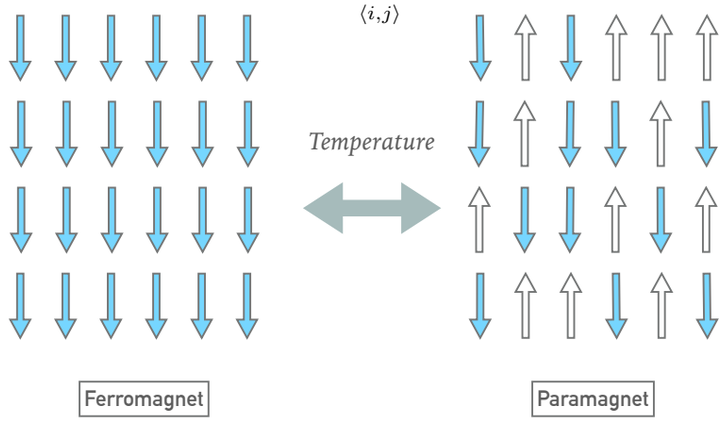
[Source: Illustris 1405.2921]

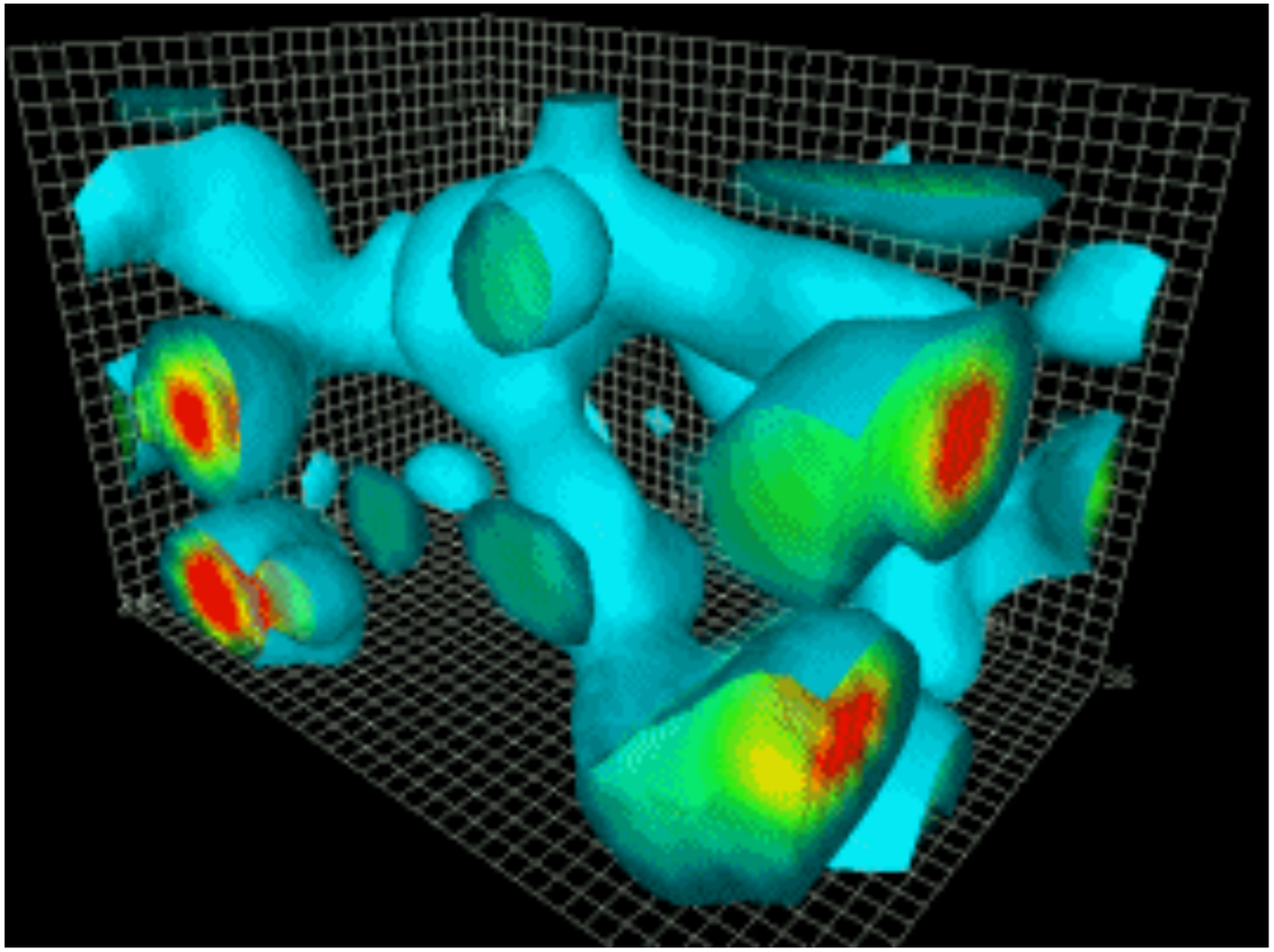
LATTICE FIELD THEORY

PHASES, PHASE TRANSITIONS, AND THE ORDER PARAMETER

Ising ferromagnet in two dimensions

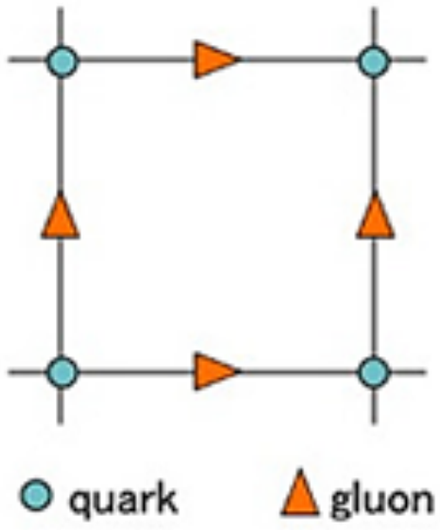
$$E = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j$$

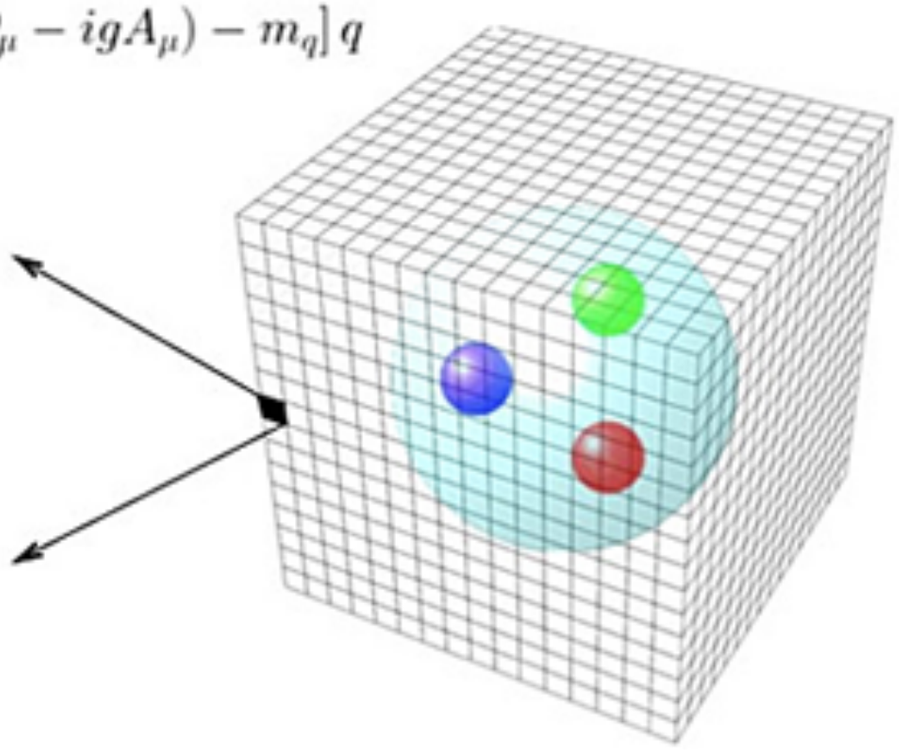




QCD Lagrangian

$$\mathcal{L} = -\frac{1}{4} F^{\mu\nu} F_{\mu\nu} + \sum_{q=u,d,s,c,b,t} \bar{q} \left[i\gamma^\mu (\partial_\mu - igA_\mu) - m_q \right] q$$



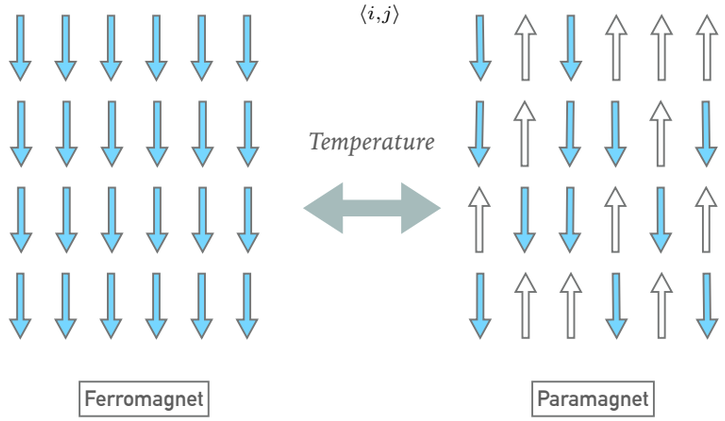


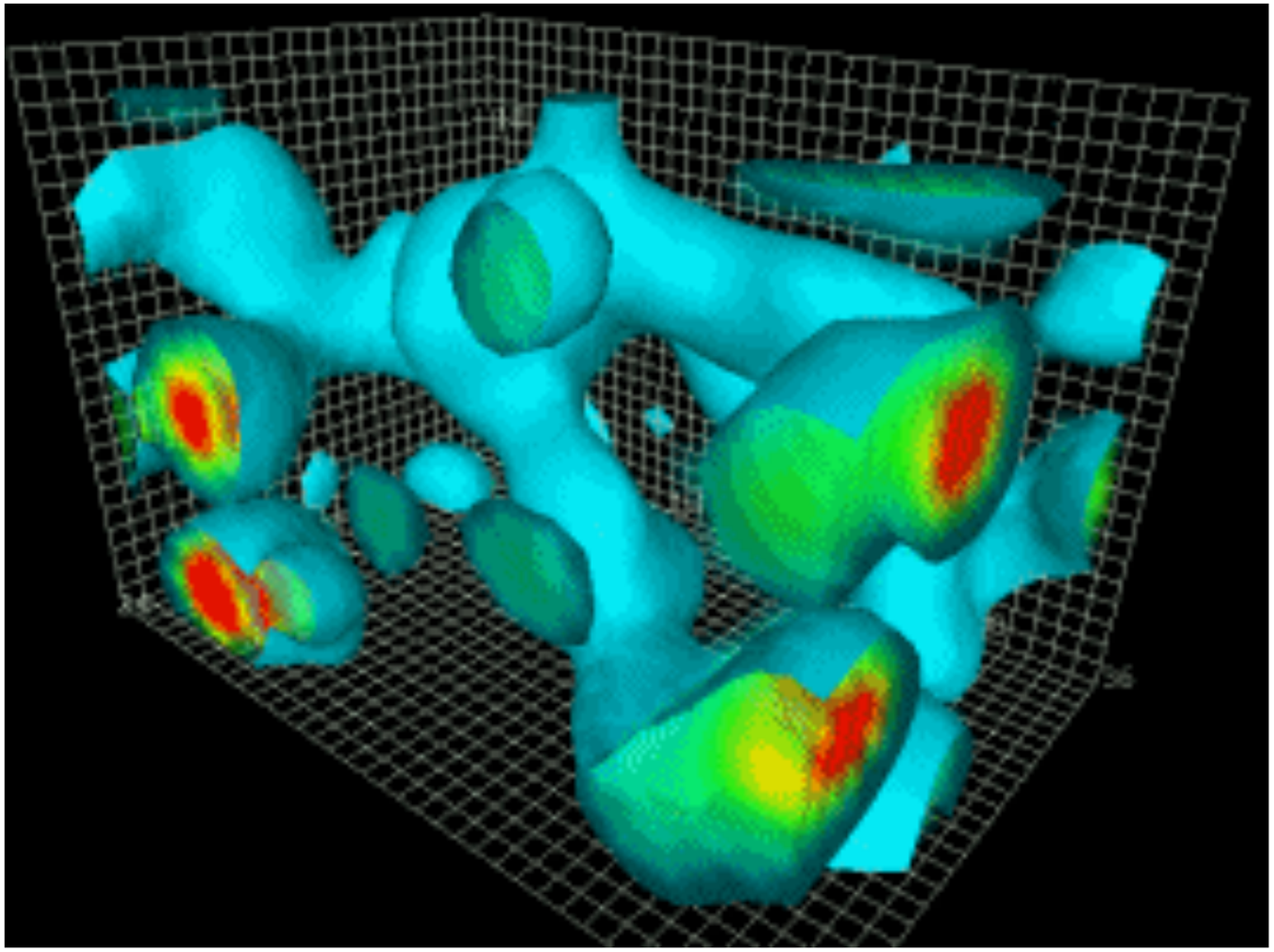
LATTICE FIELD THEORY

PHASES, PHASE TRANSITIONS, AND THE ORDER PARAMETER

Ising ferromagnet in two dimensions

$$E = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j$$





QCD Lagrangian

$$\mathcal{L} = -\frac{1}{4} F^{\mu\nu} F_{\mu\nu} + \sum_{q=u,d,s,c,b,t} \bar{q} \left[i\gamma^\mu (\partial_\mu - igA_\mu) - m_q \right] q$$