NYU Center for Data Science

UCL Université catholique de Louvain

Center for Cosmology and Particle Physics

CHAIRE GEORGES LEMAÎTRE 2017

PHYSICS, STATISTICS, AND MACHINE LEARNING

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

Menu of Topics

Statistical Topics

- probability, Bayes/Frequentist, Likelihood, transformation properties, correlation vs. mutual information, information geometry
- parameter estimation, bias/variance tradeoff, Cramér-Rao bound, James-Stein paradox
- Statistical Decision Theory
- Conceptual issues around Goodness of fit
- Hypothesis Testing, Neyman-Pearson, likelihood ratios
- Confidence intervals, coverage, Neyman Construction, Bayesian credible intervals, MCMC, CLs
- Systematics, profile-likelihood, asymptotic distributions
- Bayesian Posteriors, MCMC, and Variational Inference
- look-elsewhere effect, 1-d, 2-d, combination of experiments, …
- unfolding, inverse problems, regularization, connection to Gaussian Processes & RBKH

Probabilistic Modeling of Data: Classical and Machine Learning versions

- clarification of "correlated systematic" confusion
- Scientific Narratives: Monte-Carlo template based, parametrized function, data-driven, …
- Template approach & HistFactory, "experimental design"
- Kernel Density estimation
- Gaussian Processes & connection to unfolding
- neural density estimation, autoregressive models, normalizing flows
- the data manifold and auto-encoders, anomaly detection
- GANs and Variational Auto-encoders

ML ↔ Stats correspondence

- goodness of fit ↔ anomaly detection
- Hypothesis Testing ↔ classifiers
- parameter estimation ↔ regression (and neural networks as function approximations)
- statistical decision theory ↔ reinforcement learning
- Systematics: Learning to Profile and Learning to Pivot
- credible intervals with Bayesian neural networks & Gaussian Processes
- Auto-encoding variational Bayes

ML-based Likelihood-free approaches

- Kernel Density estimation
- Cox Process & Gaussian Processes https://arxiv.org/abs/1709.05681
- likelihood ratios from classifiers & parametrized learning
- conditional density estimation: autoregressive models, normalizing flows
- the data manifold and auto-encoders, anomaly detection
- Approximate Bayesian Computation
- Probabilistic Programming
- GANs and Variational Auto-encoders
- Adversarial Variational Optimization

Black box optimization

- Bayesian Optimization & Variational Optimization

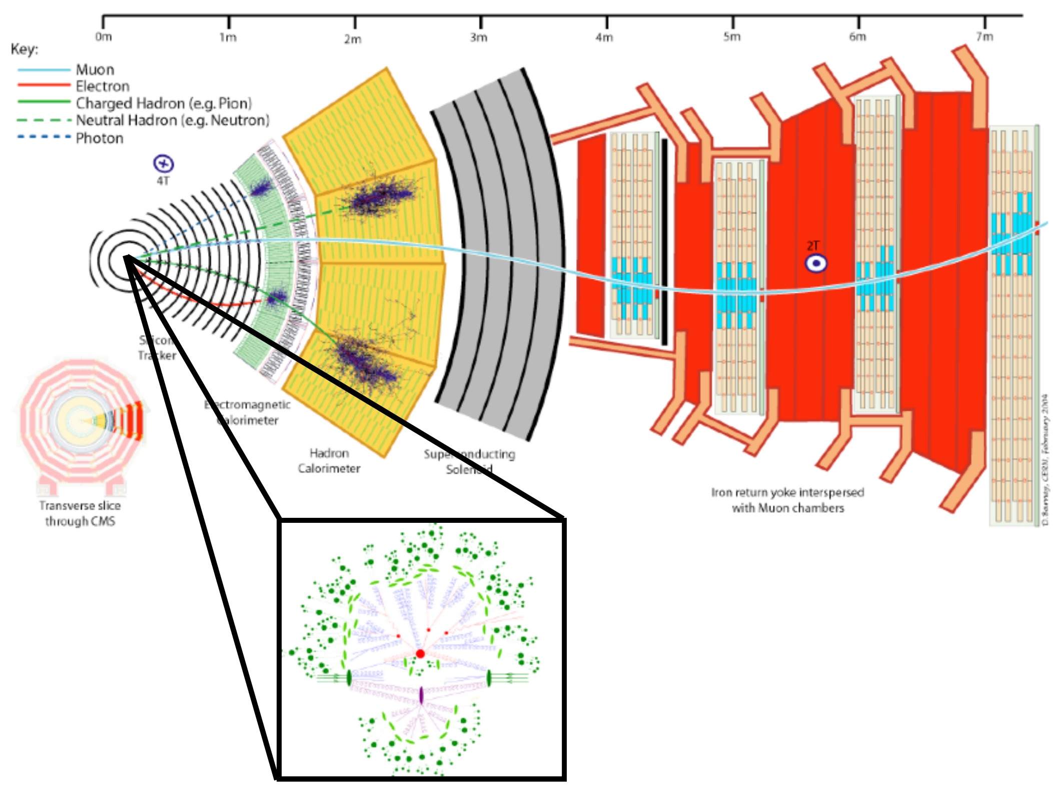
Recent ML Topics:

- Parametrized learning for classification
- Parametrized learning for likelihood-free inference
- High-dimensional reweighting
- Incorporating systematics into neural network training “Learning to pivot”
- Decorrelating neural networks from some variable (eg. mass of particle)
- Gaussian Processes for modeling backgrounds & generic localized signals
- Information geometry as a tool for phenomenology
- Adversarial Variational Optimization for tuning simulation
- QCD-aware neural networks
- Simplified likelihoods

TWO APPROACHES

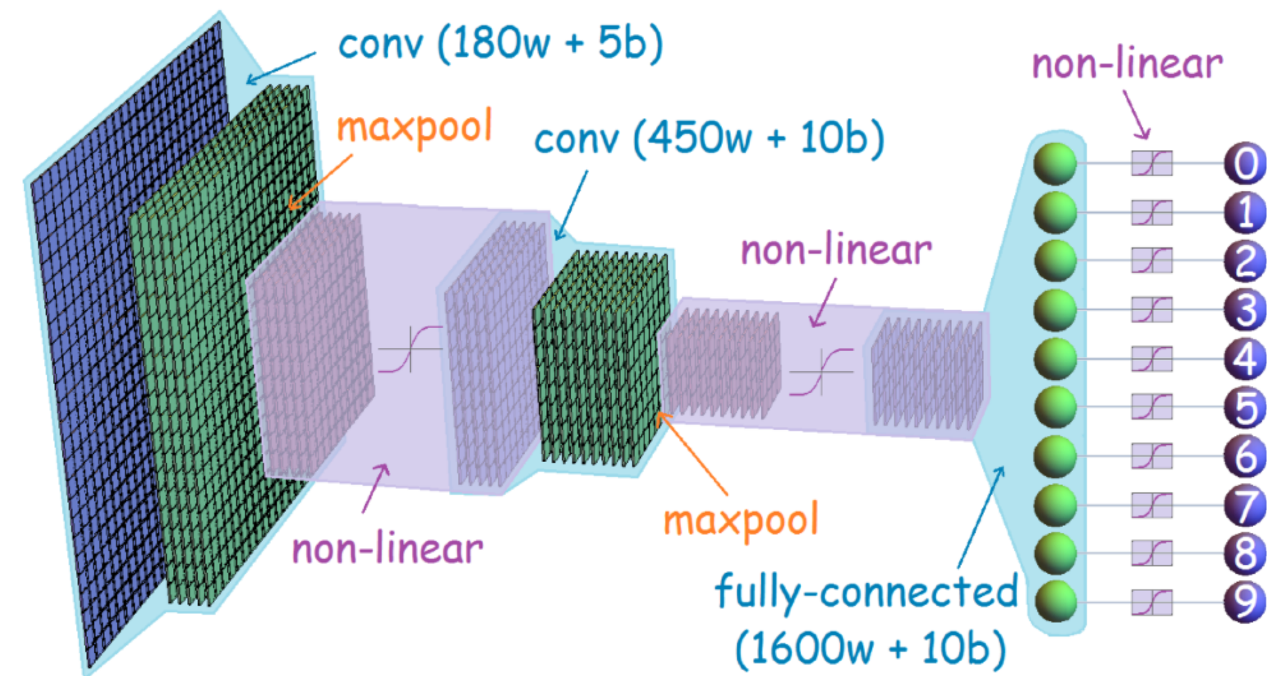
Use simulator

(much more efficiently)



Learn simulator

(with deep learning)



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

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

[image credit: A.P. Goucher]

LECTURE NOTES

Practical Statistics for the LHC

Kyle Cranmer
Center for Cosmology and Particle Physics, Physics Department, New York University, USA

Abstract
This document is a pedagogical introduction to statistics for particle physics. Emphasis is placed on the terminology, concepts, and methods being used at the Large Hadron Collider. The document addresses both the statistical tests applied to a model of the data and the modeling itself . I expect to release updated versions of this document in the future.

Links:

On Authorea

arxiv:1503.07622

Contents

1 Introduction . . . 3
2 Conceptual building blocks for modeling . . . 3
2.1 Probability densities and the likelihood function . . . 3
2.2 Auxiliary measurements . . . 5
2.3 Frequentist and Bayesian reasoning . . . 6
2.4 Consistent Bayesian and Frequentist modeling of constraint terms . . . 7
3 Physics questions formulated in statistical language . . . 8
3.1 Measurement as parameter estimation . . . 8
3.2 Discovery as hypothesis tests . . . 9
3.3 Excluded and allowed regions as confidence intervals . . . 11
4 Modeling and the Scientific Narrative . . . 14
4.1 Simulation Narrative . . . 15
4.2 Data-Driven Narrative . . . 25
4.3 Effective Model Narrative . . . 27
4.4 The Matrix Element Method . . . 27
4.5 Event-by-event resolution, conditional modeling, and Punzi factors . . . 28
5 Frequentist Statistical Procedures . . . 28
5.1 The test statistics and estimators of μ and $\boldsymbol{\theta}$ . . . 29
5.2 The distribution of the test statistic and p-values . . . 31
5.3 Expected sensitivity and bands . . . 32
5.4 Ensemble of pseudo-experiments generated with “Toy” Monte Carlo . . . 33
5.5 Asymptotic Formulas . . . 33
5.6 Importance Sampling . . . 36
5.7 Look-elsewhere effect, trials factor, Bonferoni . . . 37
5.8 One-sided intervals, CLs, power-constraints, and Negatively Biased Relevant Subsets . . . 37
6 Bayesian Procedures . . . 38
6.1 Hybrid Bayesian-Frequentist methods . . . 39
6.2 Markov Chain Monte Carlo and the Metropolis-Hastings Algorithm . . . 40
6.3 Jeffreys’s and Reference Prior . . . 40
6.4 Likelihood Principle . . . 41
7 Unfolding . . . 42
8 Conclusions . . . 42

Probability & Statistics
Terminology & Definitions

Terms

The next lectures will rely on a clear understanding of these terms:

- Random variables / “observables” x
- Probability mass and probability density function (pdf) $p(x)$ or $f(x)$
- Parametrized Family of pdfs / “model” $p(x|\alpha)$
- Parameter α
- Likelihood $L(\alpha)$
- Estimate (of a parameter) $\hat{\alpha}(x)$

When dealing with discrete random variables, define a **Probability Mass Function** as probability for i^{th} possibility

$$P(x_i) = p_i$$

Defined as limit of long term frequency

- probability of rolling a 3 := $\text{limit}_{\#\text{trials} \to \infty}$ (# rolls with 3 / # trials)
  - you don't need an infinite sample for definition to be useful

And it is normalized

$$\sum_i P(x_i) = 1$$

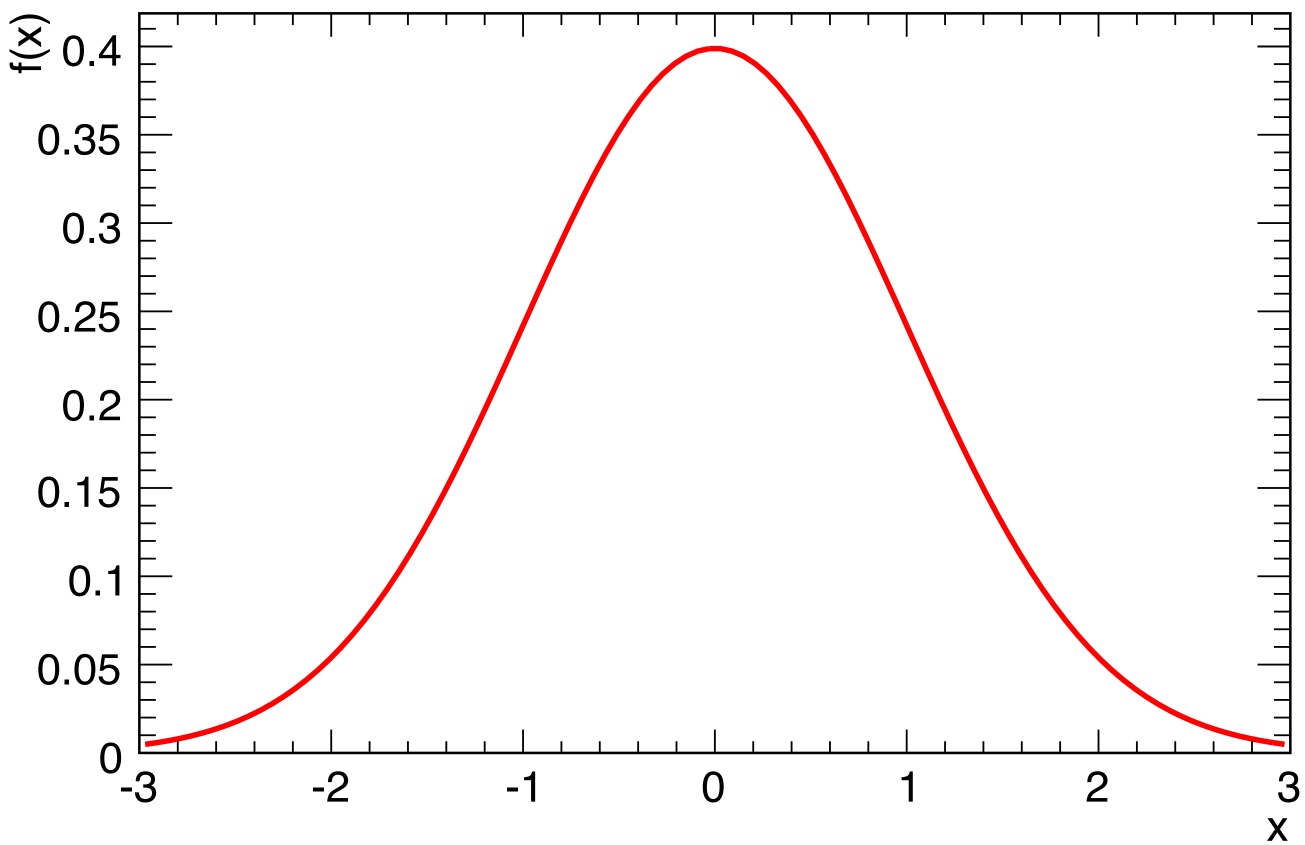
When dealing with continuous random variables, need to introduce the notion of a **Probability Density Function**

$$P(x \in [x, x+dx]) = f(x)dx$$

Note, $f(x)$ is NOT a probability

PDFs are always normalized to unity:

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

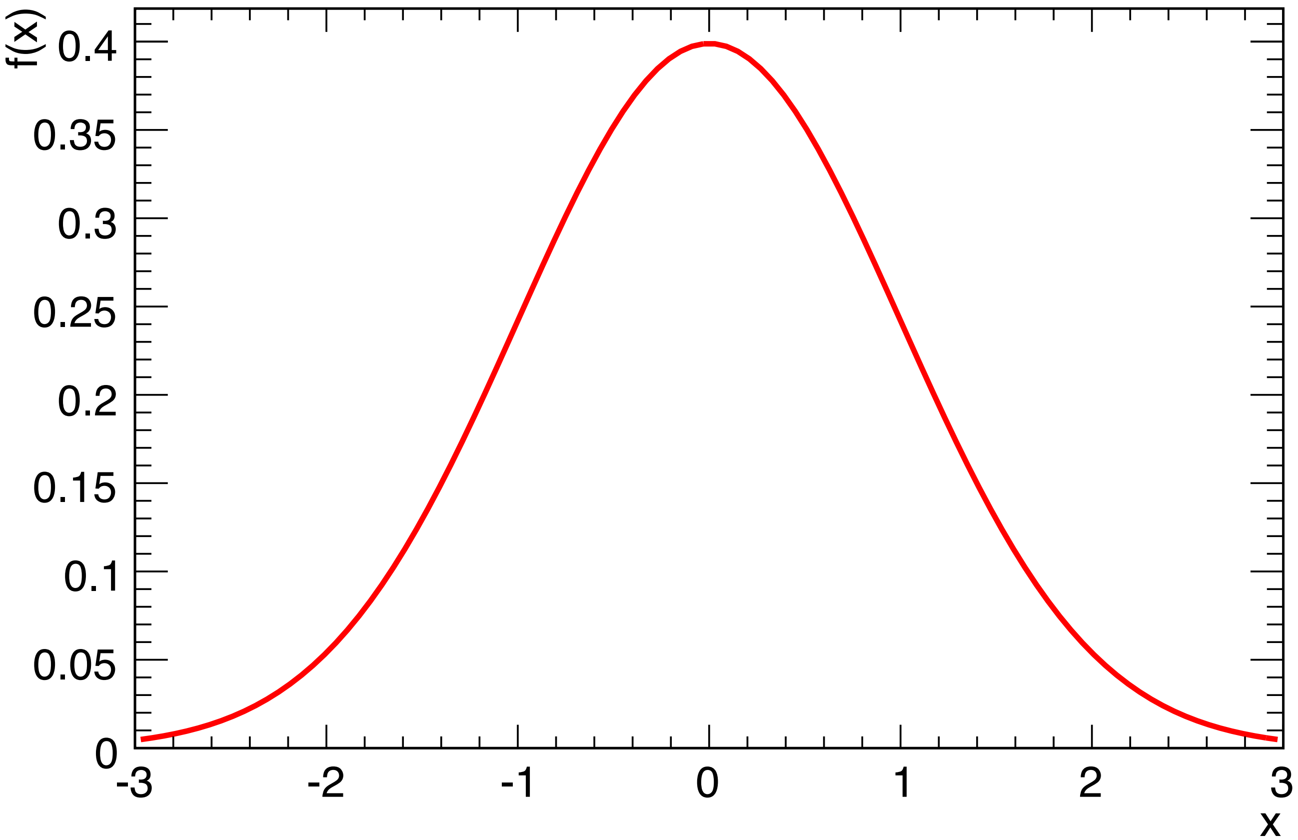


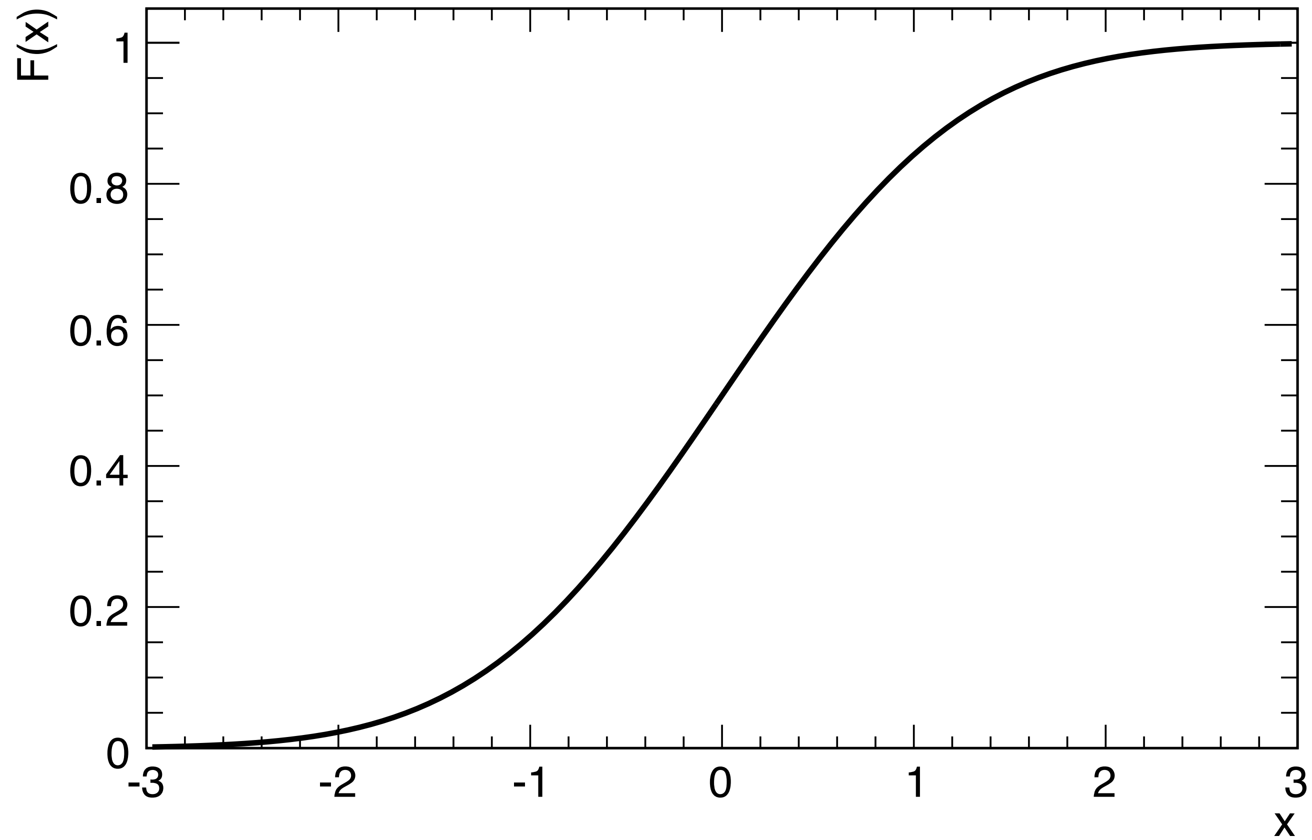
CUMULATIVE DENSITY FUNCTIONS

Often useful to use a cumulative distribution:

- in 1-dimension:

$$\int_{-\infty}^{x} f(x')dx' = F(x)$$



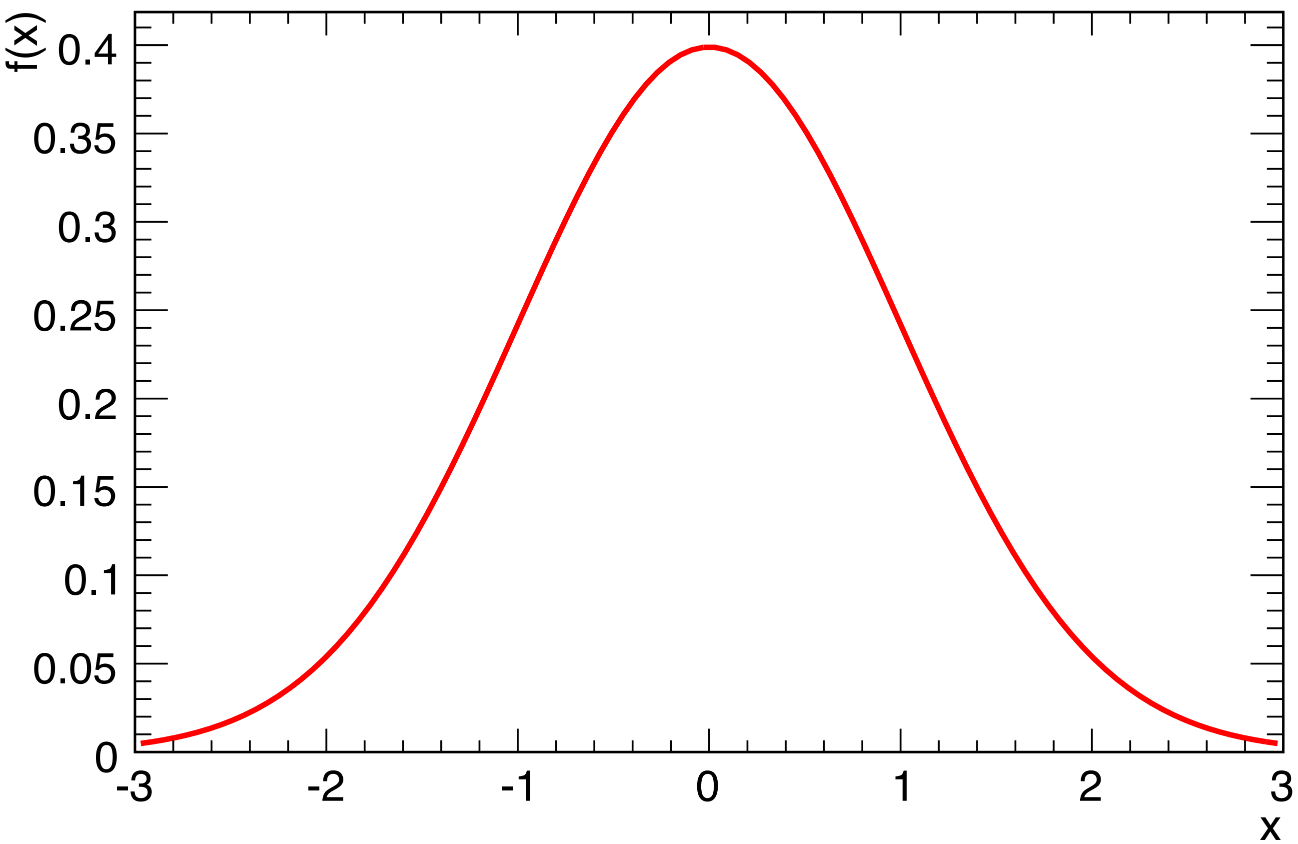


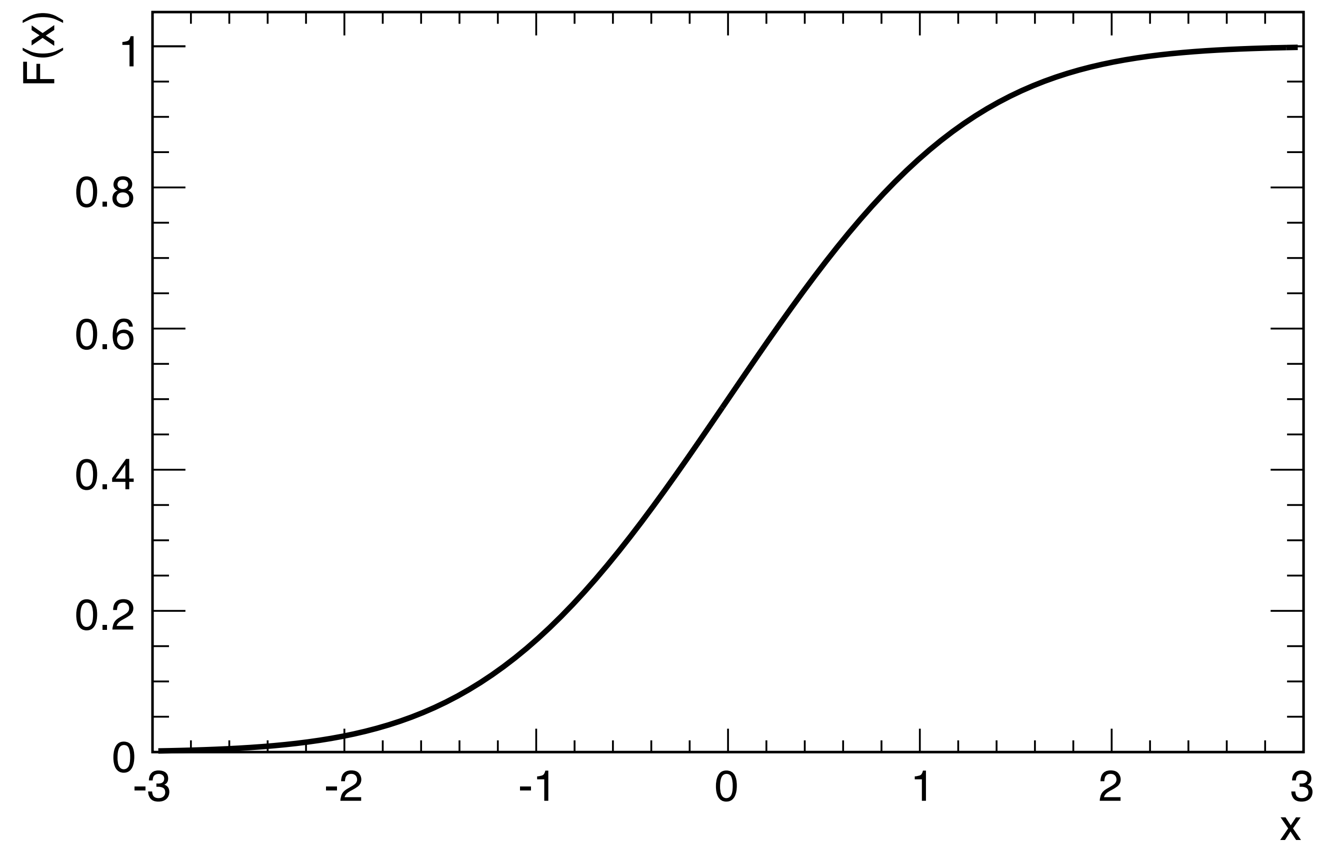
Cumulative Density Functions

Often useful to use a cumulative distribution:

- in 1-dimension:

$$\int_{-\infty}^{x} f(x')dx' = F(x)$$





- alternatively, define density as partial of cumulative:

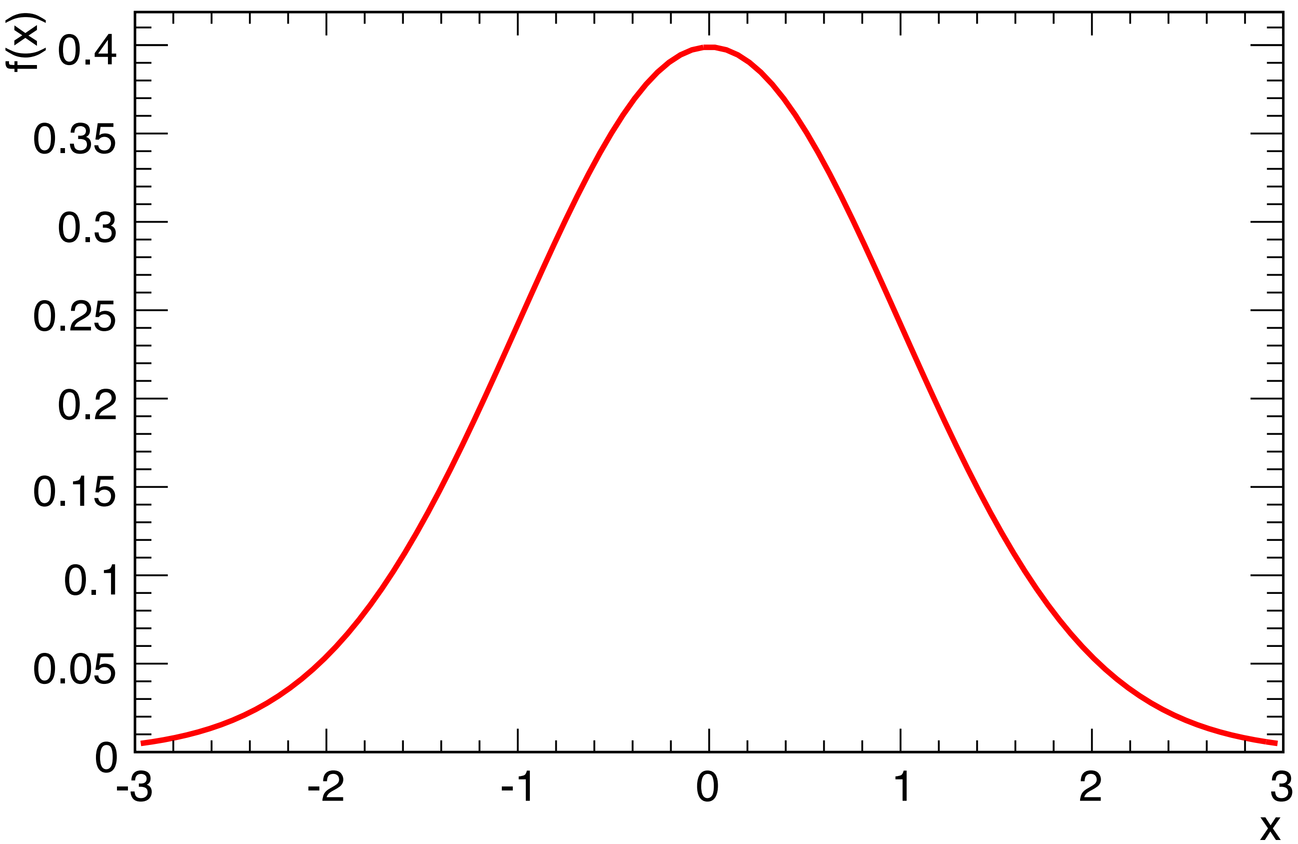
$$f(x) = \frac{\partial F(x)}{\partial x}$$

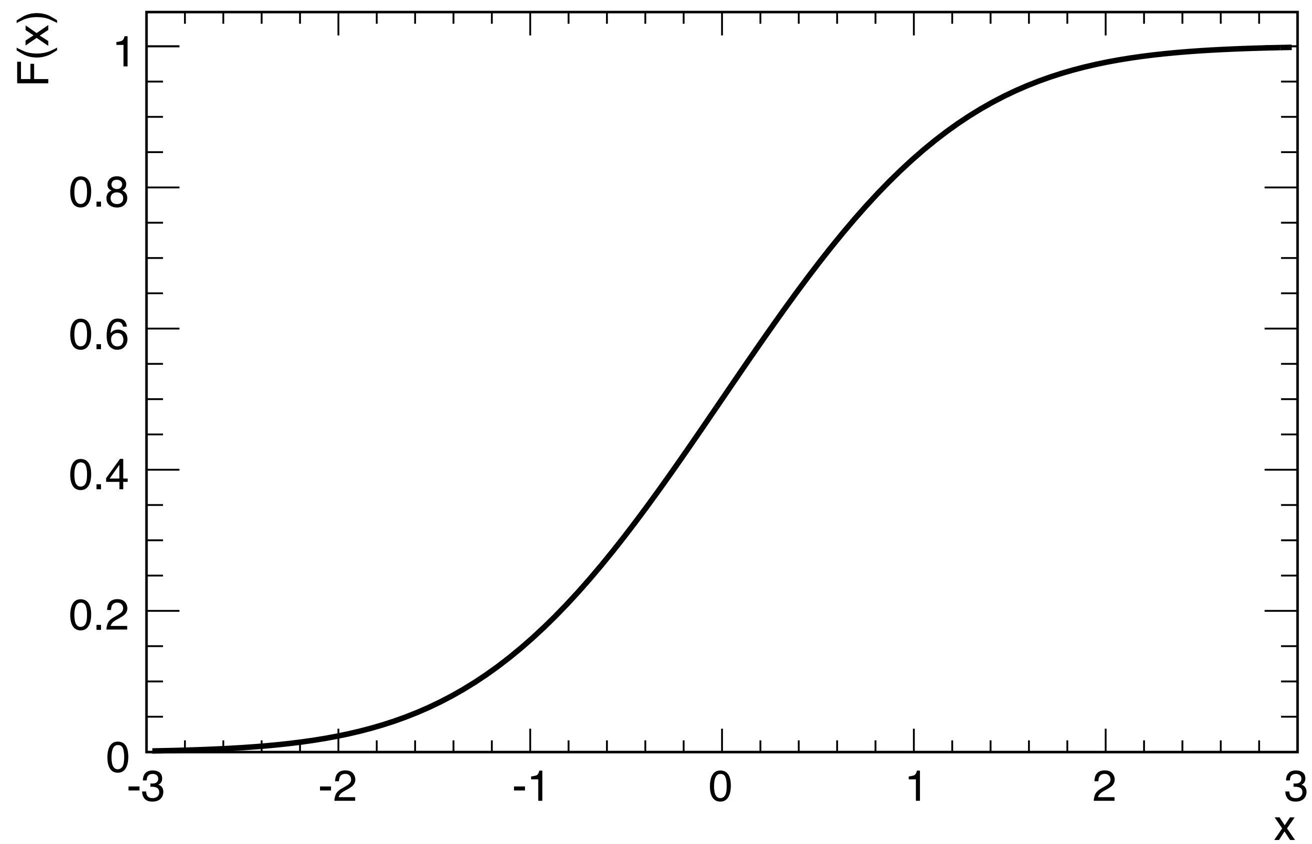
CUMULATIVE DENSITY FUNCTIONS

Often useful to use a cumulative distribution:

- in 1-dimension:

$$\int_{-\infty}^{x} f(x')dx' = F(x)$$





- alternatively, define density as partial of cumulative:

$$f(x) = \frac{\partial F(x)}{\partial x}$$

- same relationship as total and differential cross section:

$$f(E) = \frac{1}{\sigma}\frac{\partial \sigma}{\partial E}$$

HISTOGRAM $\{x_i\} \rightarrow f(x)$

Given a set of observations $\{x_i\}$ we can approximate the pdf with a histogram.

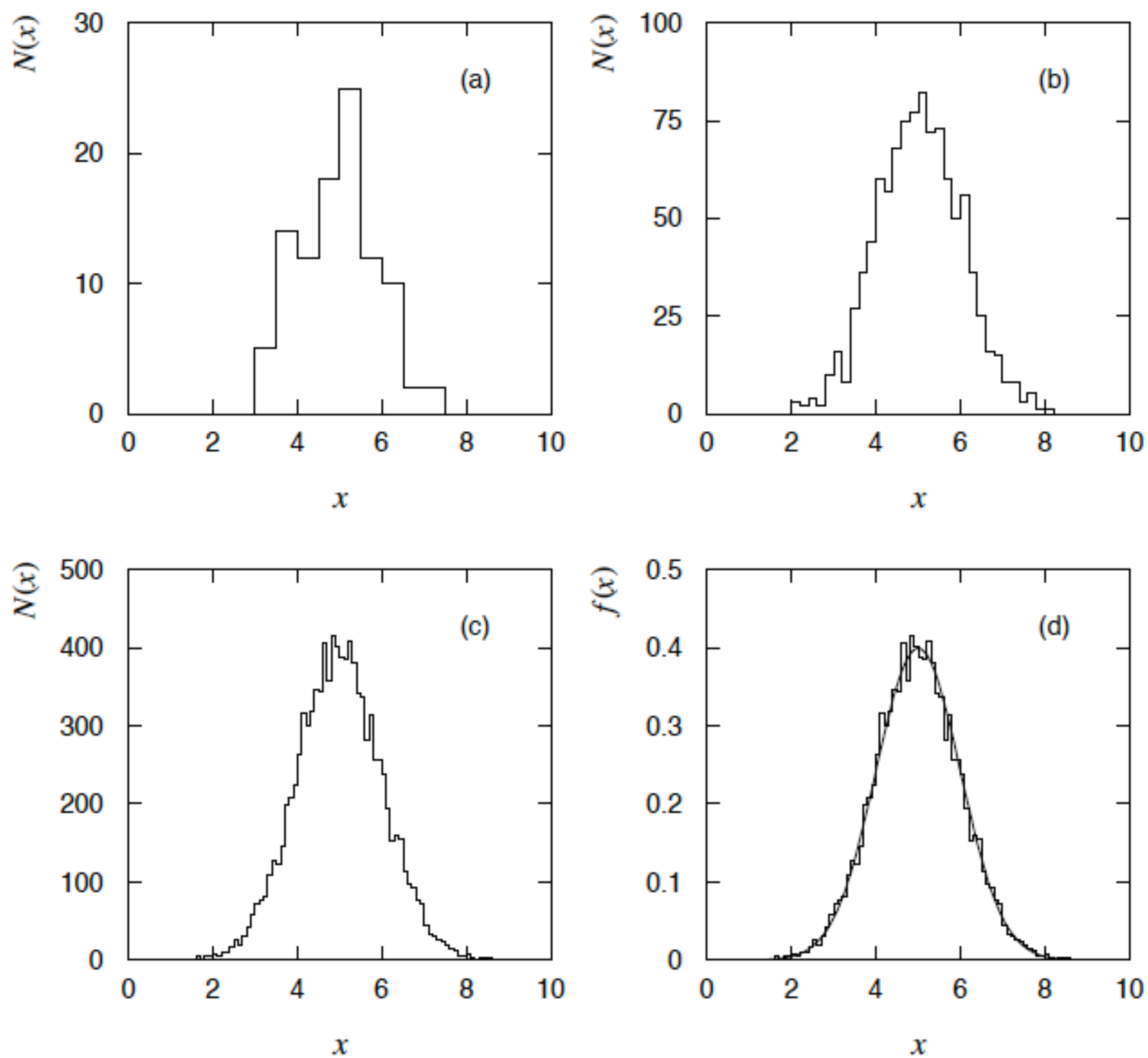
Think of a pdf as a histogram with:

infinite data sample,
zero bin width,
normalized to unit area.

$$f(x) = \frac{N(x)}{n\Delta x}$$

$n =$ number of entries

$\Delta x =$ bin width



[G. Cowan]

PARAMETRIZED FAMILIES / MODELS

Often we are interested in a parametried family of pdfs

- We will write these as: $f(x|\alpha)$ said "f of x given α"
  - where α are the parameters of the "model" (written in greek characters)

A discrete example:

- The Poisson distribution is a probability mass function for n, the number of events one observes, when one expects μ events

$$Pois(n|\mu) = \mu^n \frac{e^{-\mu}}{n!}$$

A continuous example

- The Gaussian distribution is a probability density function for a continuous variable x characterized by a mean μ and standard deviation σ

$$G(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The Likelihood Function

Consider the Poisson distribution describes a discrete event count n for a real-valued mean μ.

$$Pois(n|\mu) = \mu^n \frac{e^{-\mu}}{n!}$$

The **likelihood** of μ given n is the same equation evaluated as a function of μ

- Now it's a continuous function
- But it is not a pdf!

$$L(\mu) = Pois(n|\mu)$$

Common to plot the -ln L (or -2 ln L)

- helps avoid thinking of it as a PDF
- connection to χ^2 distribution

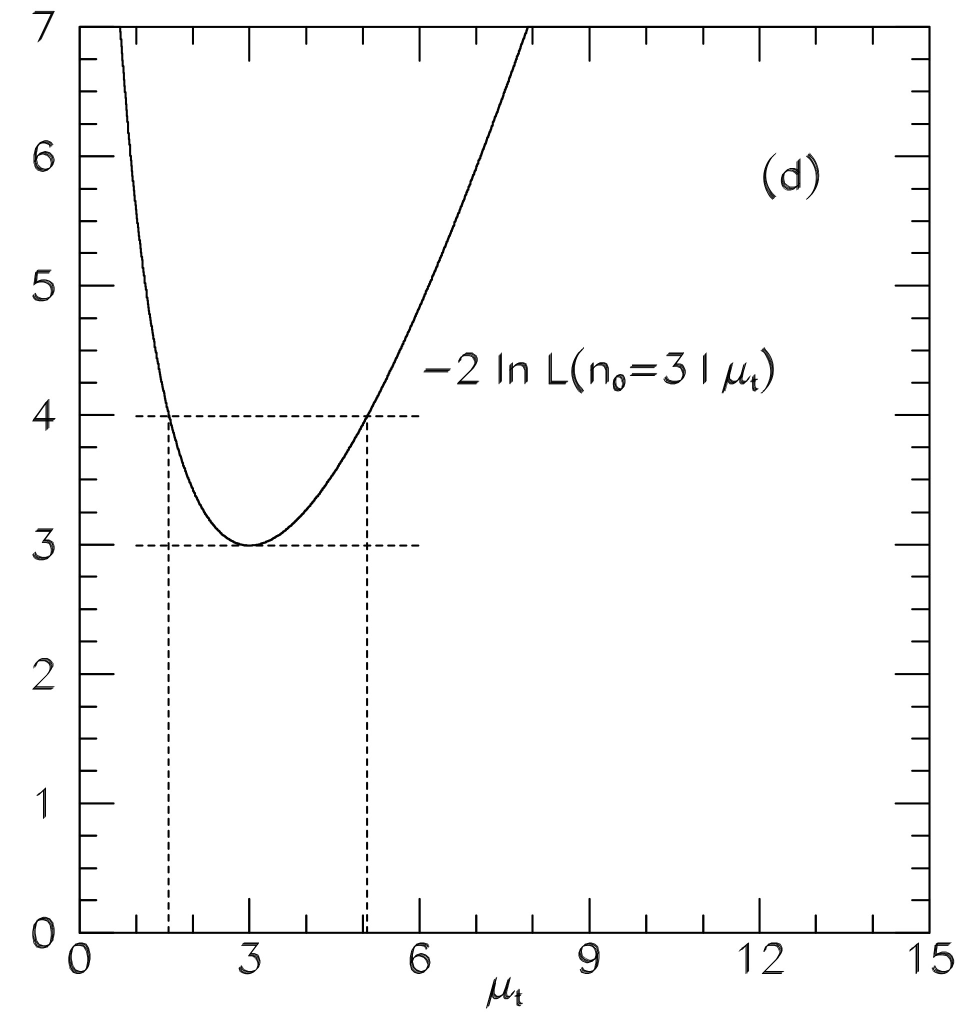


Figure from R. Cousins, Am. J. Phys. 63 398 (1995)

Repeated observations

In particle physics we are usually able to perform repeated observations of x that are **independent & identically distributed**

- These repeated observations are written $\{x_i\}$
- and the likelihood in that case is

$$L(\alpha) = \prod_i f(x_i|\alpha)$$

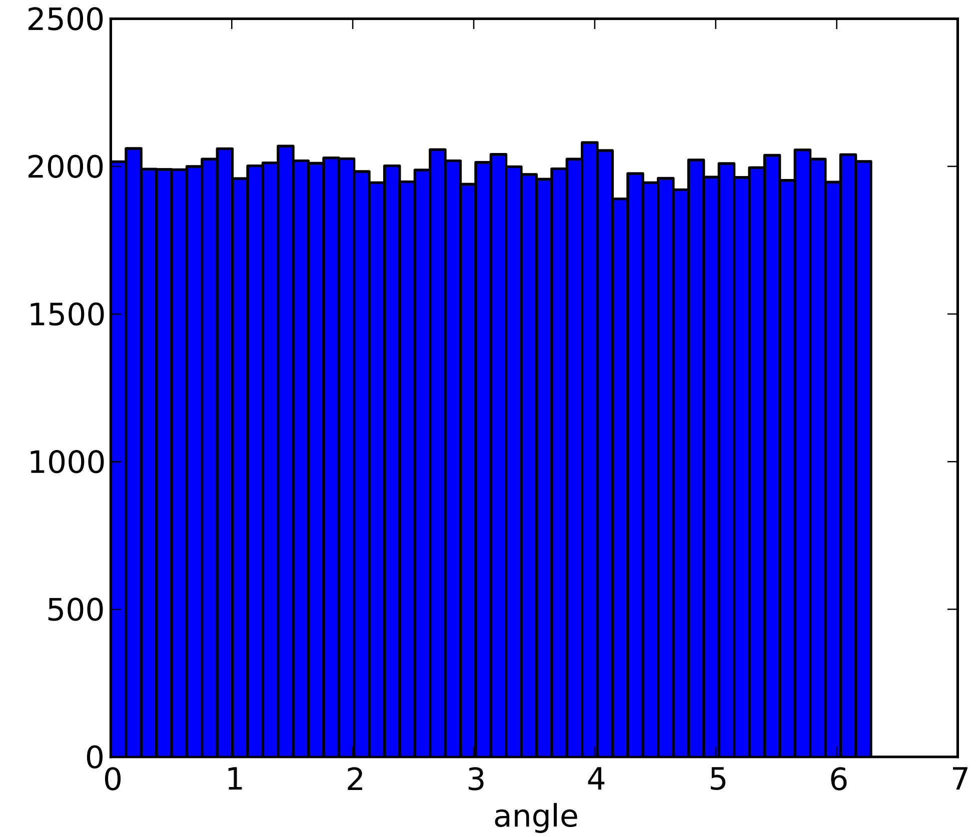
- and the log-likelihood is

$$\log L(\alpha) = \sum_i \log f(x_i|\alpha)$$

TRANSFORMATION PROPERTIES:
PDF VS. LIKELIHOOD

What happens with x⟶ cos(x)

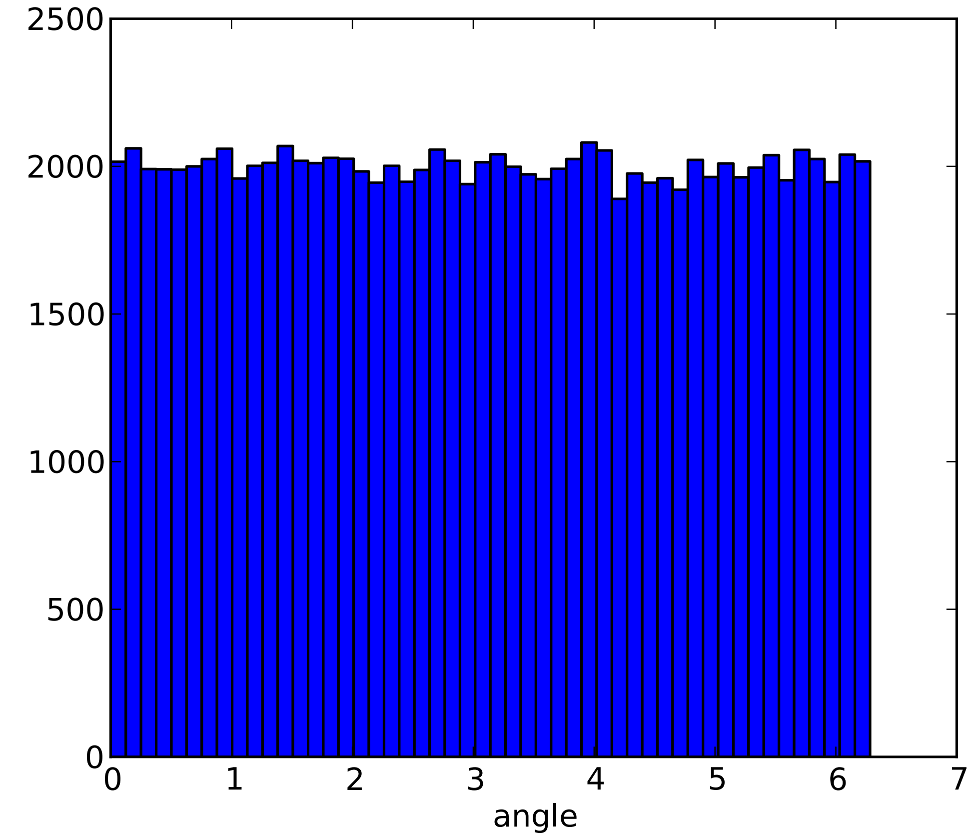
```
import numpy as np
import matplotlib.pyplot as plt

N_MC=100000  # number of Monte Carlo Experiments
nBins = 50 # number of bins for Histograms

data_x, data_y = [],[]  #lists that will hold x and y

# do experiments
for i in range(N_MC):
    # generate observation for x
    x = np.random.uniform(0,2*np.pi)

    y = np.cos(x)
    data_x.append(x)
    data_y.append(y)

#setup figures
fig = plt.figure(figsize=(13,5))
fig_x = fig.add_subplot(1,2,1)
fig_y = fig.add_subplot(1,2,2)

fig_x.hist(data_x,nBins)
fig_x.set_xlabel('angle')

fig_y.hist(data_y,nBins)
fig_y.set_xlabel('cos(angle)')

plt.show()
```

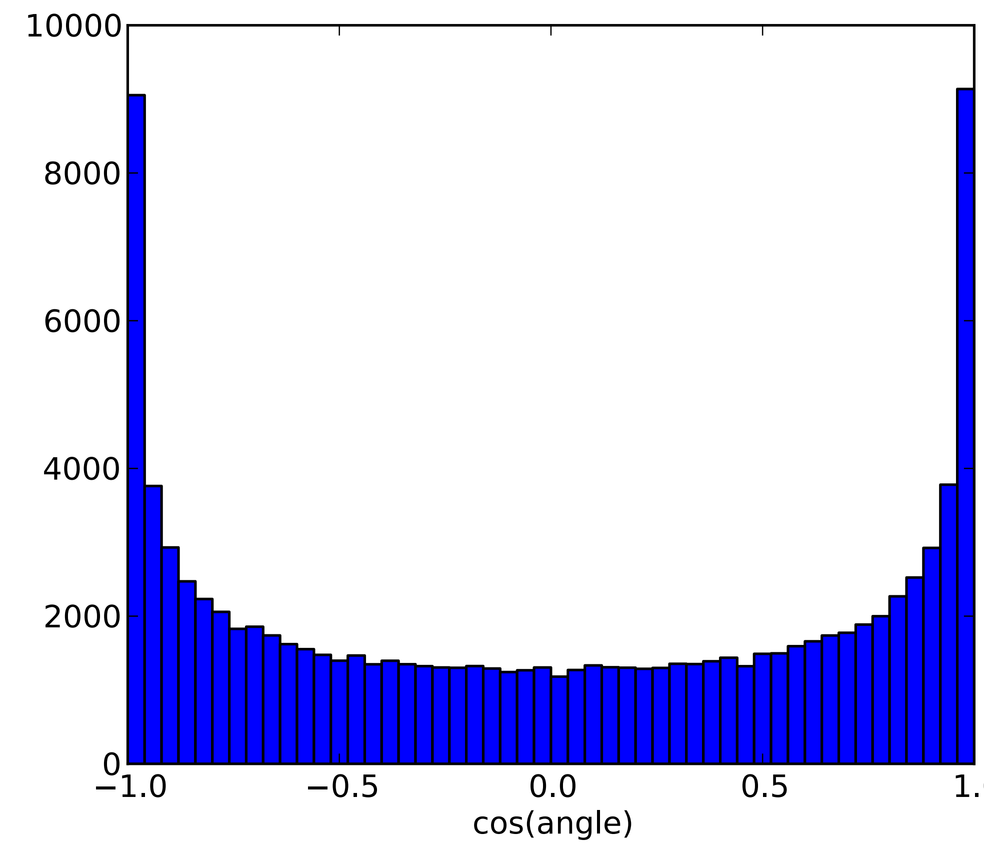


CHANGE OF VARIABLES

What happens with x⟶ cos(x)

```
import numpy as np
import matplotlib.pyplot as plt

N_MC=100000  # number of Monte Carlo Experiments
nBins = 50 # number of bins for Histograms

data_x, data_y = [],[]  #lists that will hold x and y

# do experiments
for i in range(N_MC):
    # generate observation for x
    x = np.random.uniform(0,2*np.pi)

    y = np.cos(x)
    data_x.append(x)
    data_y.append(y)

#setup figures
fig = plt.figure(figsize=(13,5))
fig_x = fig.add_subplot(1,2,1)
fig_y = fig.add_subplot(1,2,2)

fig_x.hist(data_x,nBins)
fig_x.set_xlabel('angle')

fig_y.hist(data_y,nBins)
fig_y.set_xlabel('cos(angle)')

plt.show()
```





If *f(x)* is the pdf for *x* and *y(x)* is a change of variables, then the pdf *g(y)* must satisfy

$$P(x_a < x < x_b) \equiv \int_{x_a}^{x_b} f(x)dx = \int_{y(x_a)}^{y(x_b)} g(y)dy \equiv P(y(x_a) < y < y(x_b))$$

We can rewrite the integral on the right

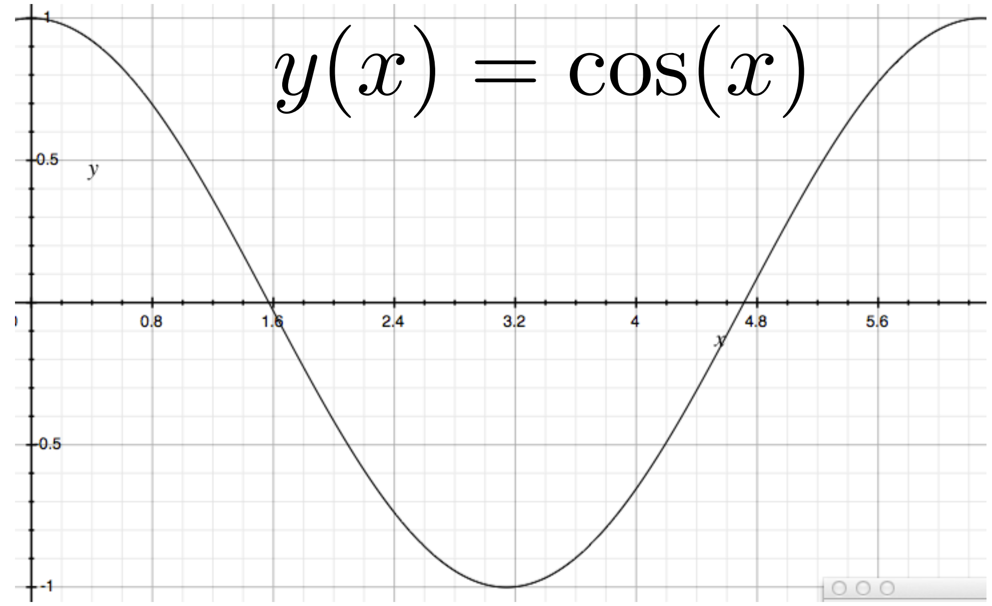
$$\int_{y(x_a)}^{y(x_b)} g(y)dy = \int_{x_a}^{x_b} g(y(x)) \left|\frac{dy}{dx}\right| dx$$

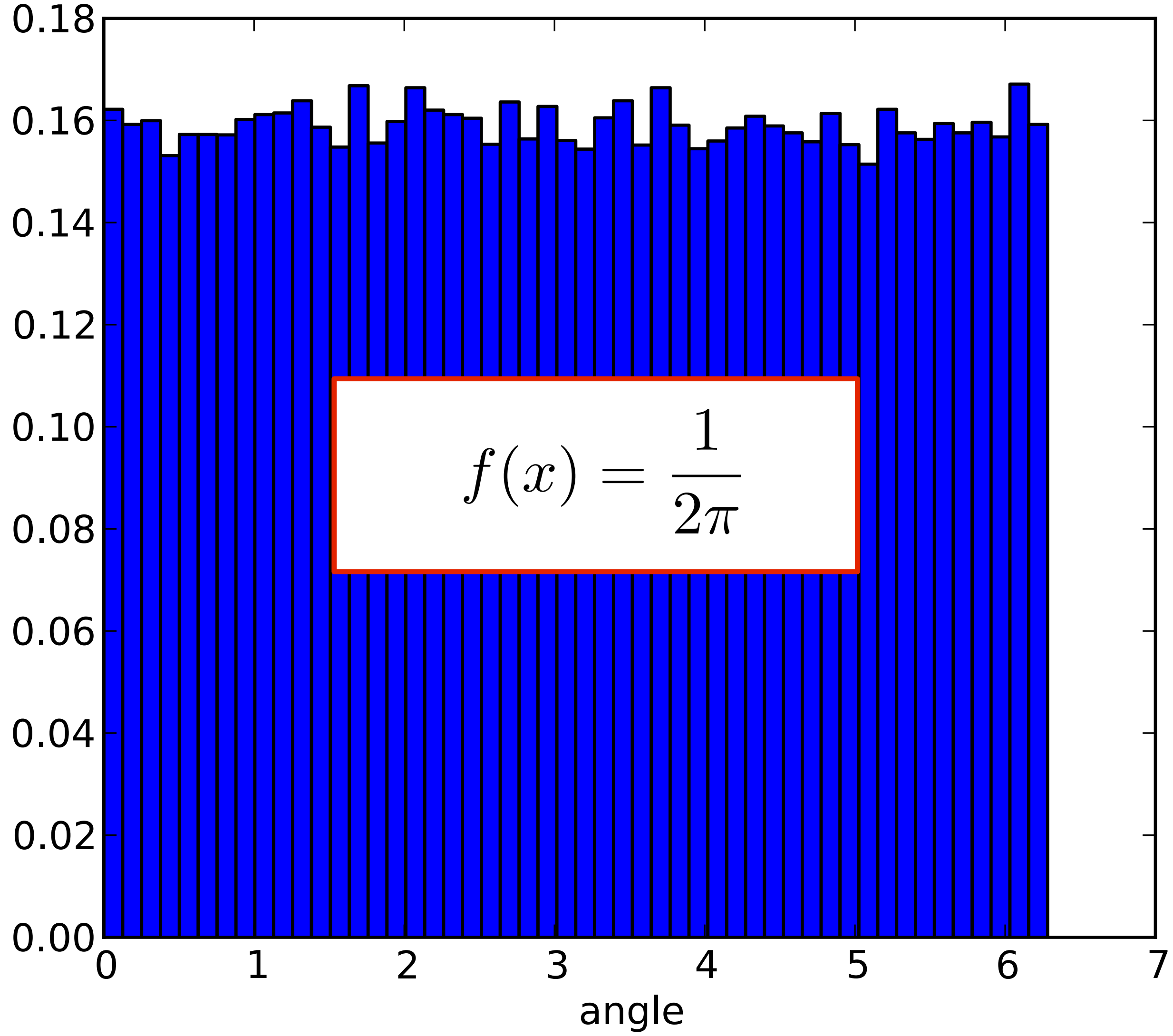
therefore, the two pdfs are related by a Jacobian factor

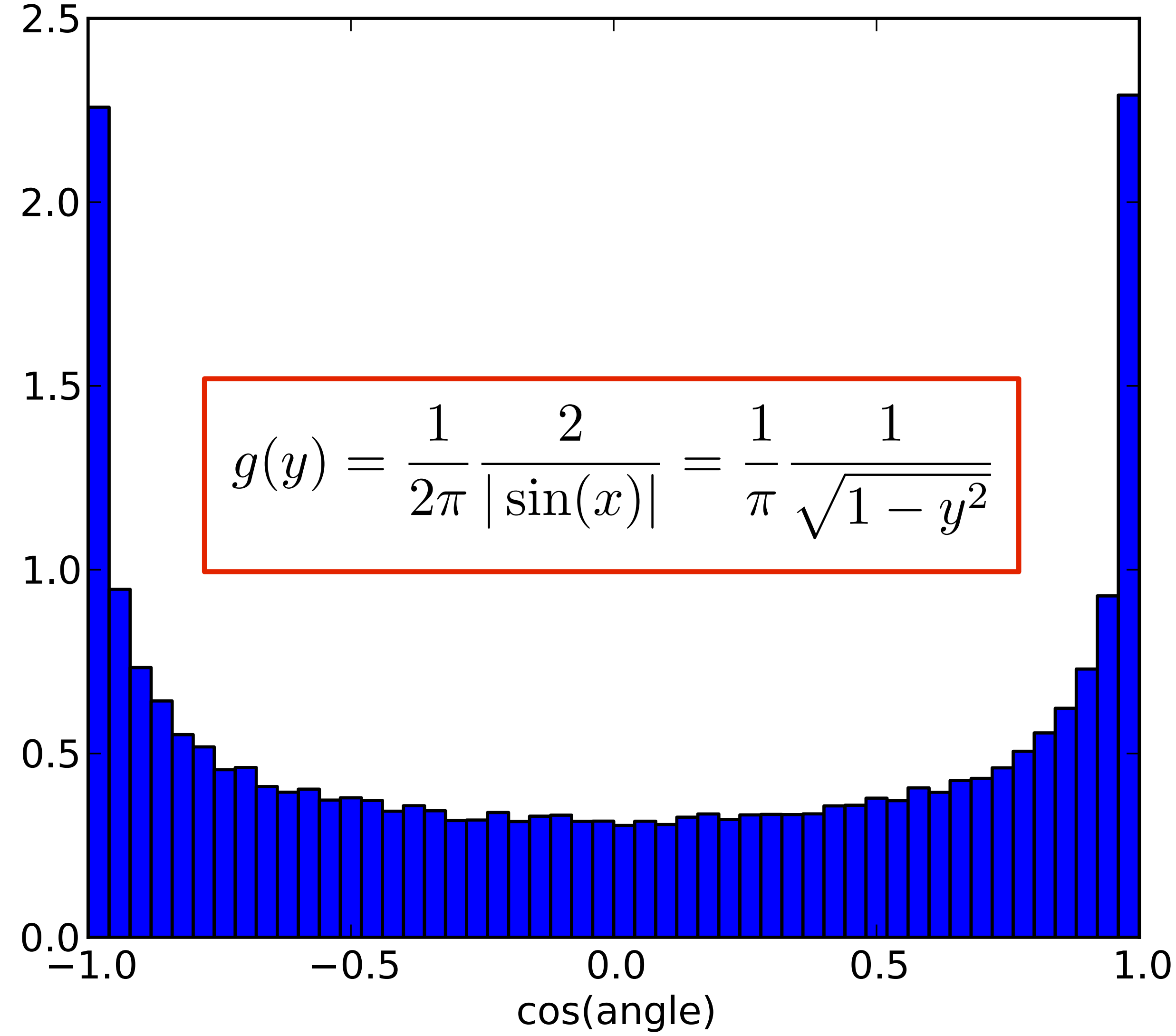
$$f(x) = g(y) \left|\frac{dy}{dx}\right|$$

AN EXAMPLE

$$f(x) = g(y) \left| \frac{dy}{dx} \right|$$

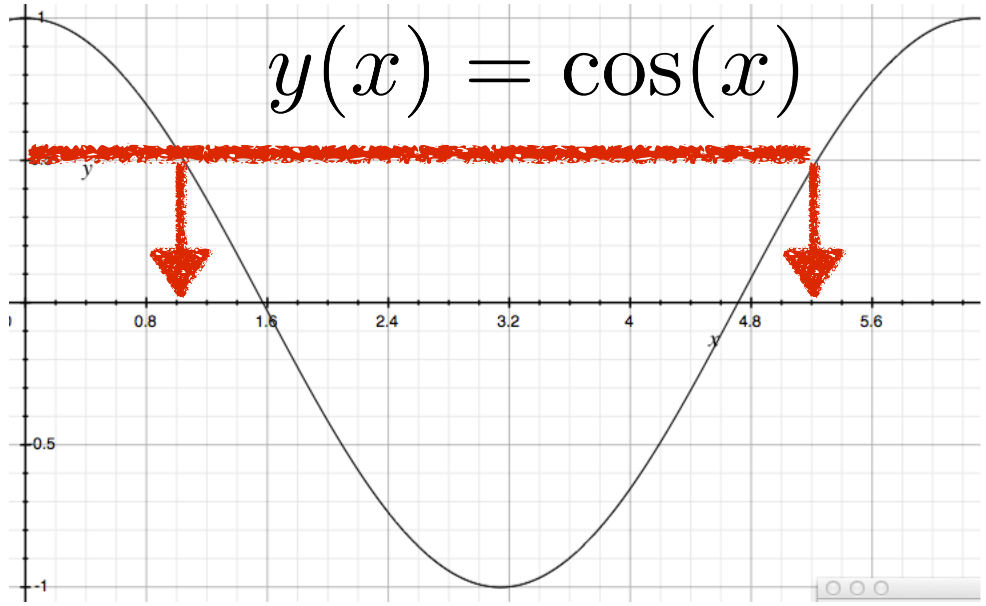


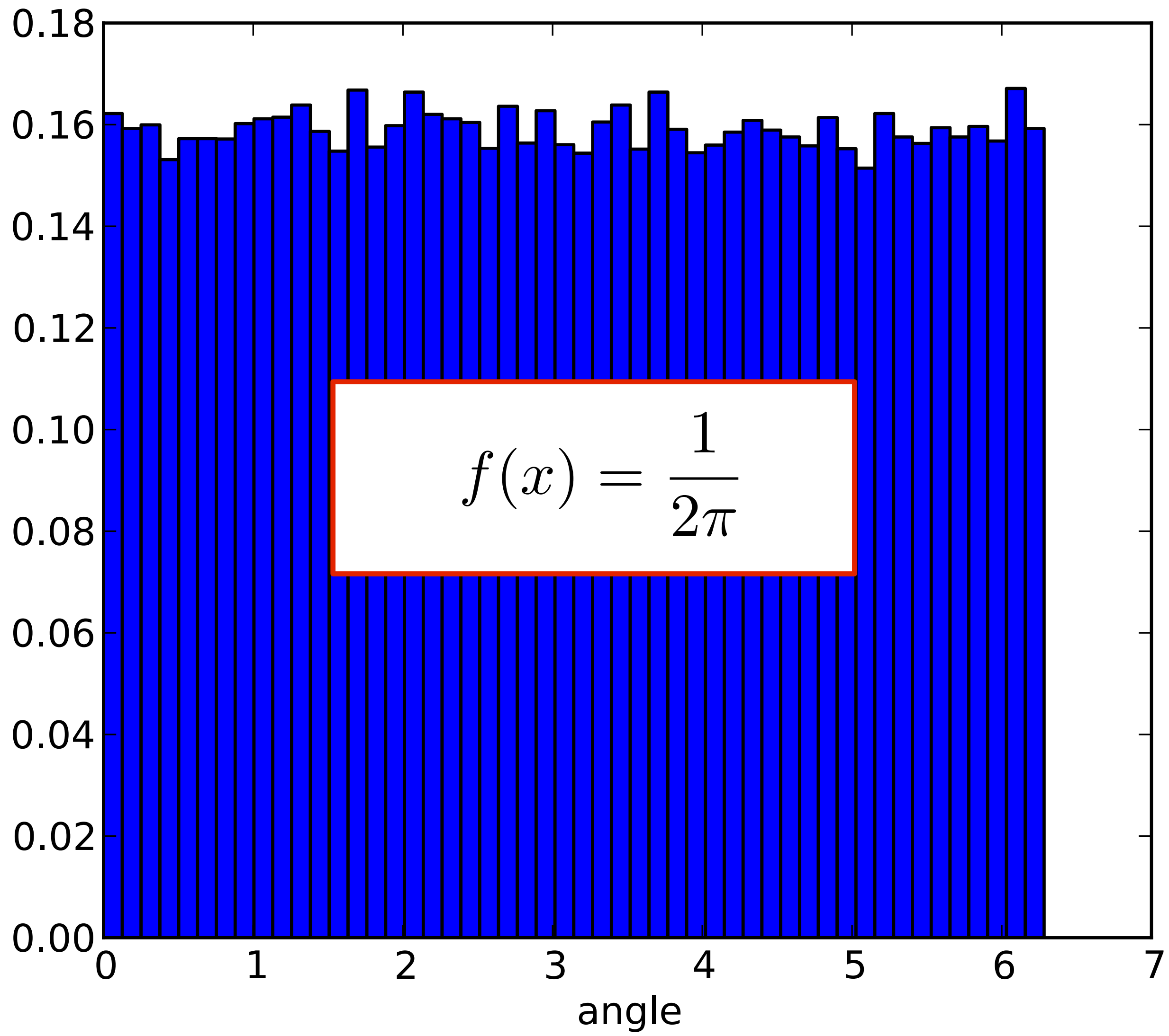


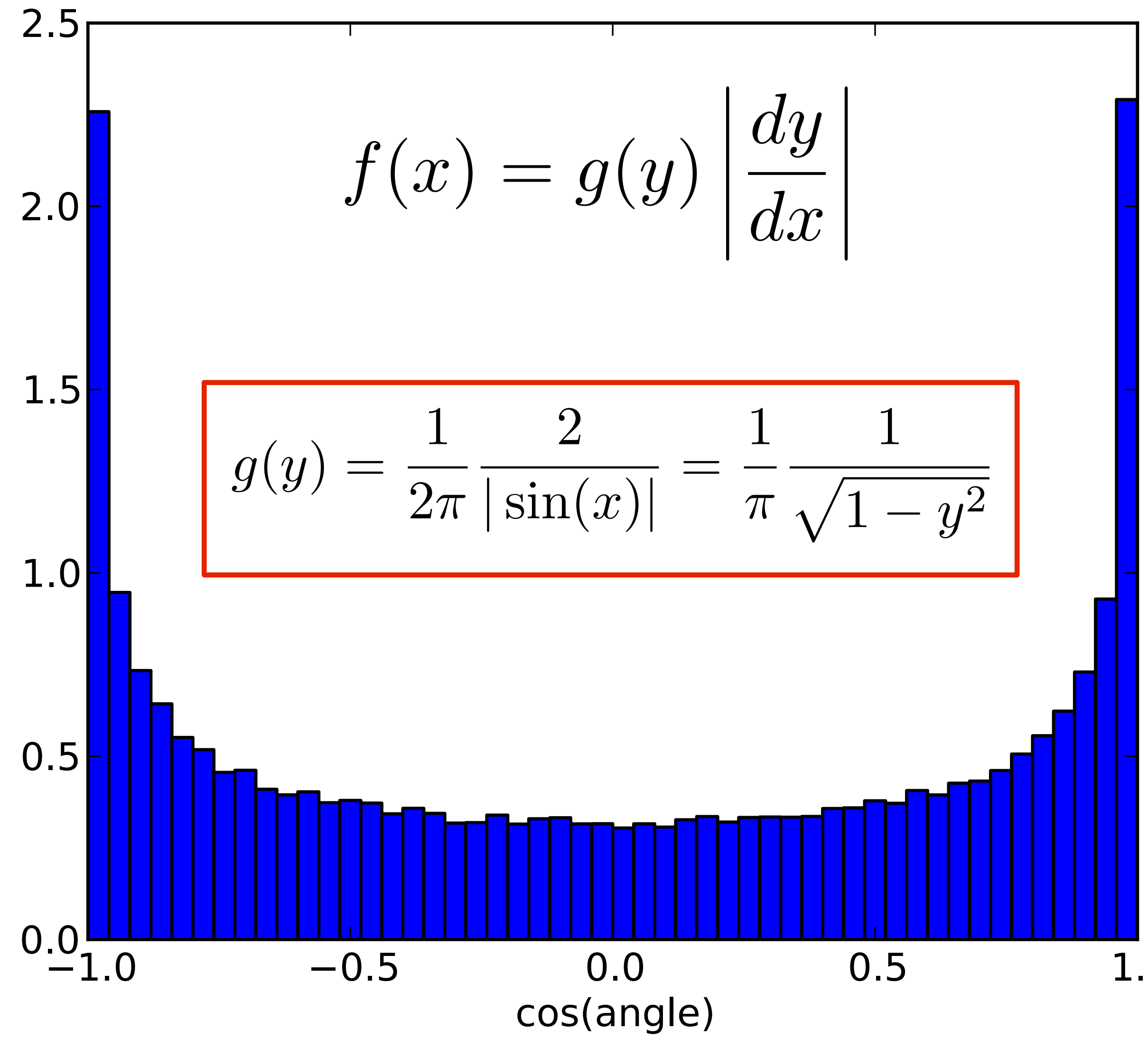


An example

I am glossing over the fact that the map is not 1-to-1. Different values of x, map into same value of y. We will need to sum/integrate over them. Here it is easy, but in general this may become intractable… need inverse map







Change of variable x, change of parameter θ

- **For pdf $p(x|\theta)$ and change of variable from x to y(x):**

  $$p(y(x)|\theta) = p(x|\theta) \,/\, |dy/dx|.$$

  Jacobian modifies probability *density*, guaranties that

  $$P(\, y(x_1) < y < y(x_2)\,) \;=\; P(x_1 < x < x_2\,),$$ **i.e., that**

  ***Probabilities* are invariant under change of variable x.**

  – **Mode of probability *density* is *not* invariant (so, e.g., criterion of maximum probability density is ill-defined).**

  – **Likelihood *ratio* is invariant under change of variable x. (Jacobian in denominator cancels that in numerator).**

- **For likelihood $\mathcal{L}(\theta)$ and reparametrization from θ to u(θ):**

  $$\mathcal{L}(\theta) \;=\; \mathcal{L}(u(\theta)) \quad (!).$$

  – **Likelihood $\mathcal{L}(\theta)$ is invariant under reparametrization of parameter θ (reinforcing fact that $\mathcal{L}$ is *not* a pdf in θ).**

Bob Cousins, CMS, 2008

The Likelihood Function

Consider the Poisson distribution describes a discrete event count n for a real-valued mean μ.

$$Pois(n|\mu) = \mu^n \frac{e^{-\mu}}{n!}$$

The **likelihood** of μ given n is the same equation evaluated as a function of μ

- Now it's a continuous function
- But it is not a pdf!

$$L(\mu) = Pois(n|\mu)$$

Common to plot the -ln L (or -2 ln L)

- helps avoid thinking of it as a PDF
- connection to χ^2 distribution

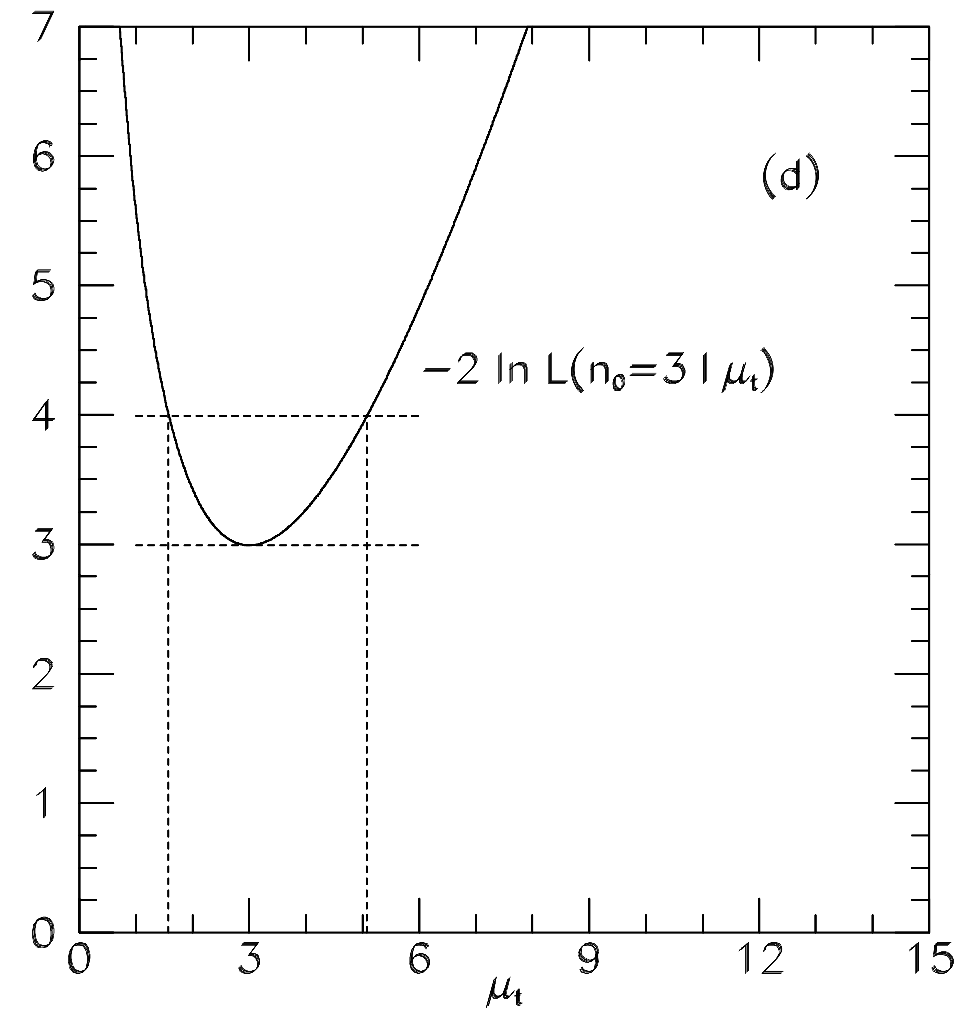


Figure from R. Cousins,
Am. J. Phys. 63 398 (1995)

Consider a specific change of variables related to the cumulative for some arbitrary *f(x)*

$$y(x) = \int_{-\infty}^{x} f(x')dx'$$

Using our general change of variables formula:

$$f(x) = g(y) \left| \frac{dy}{dx} \right|$$

We find for this case the Jacobian factor is

$$\left| \frac{dy}{dx} \right| = f(x)$$

Thus $g(y) = 1$

Probability Integral Transform

"…seems likely to be one of the most fruitful conceptions introduced into statistical theory in the last few years"
– Egon Pearson (1938)

Given continuous $x \in (a,b)$, and its pdf $p(x)$, let

$$y(x) = \int_a^x p(x')\,dx' .$$

Then $y \in (0,1)$ and $p(y) = 1$ (uniform) for all y. (!)

So there always exists a metric in which the pdf is uniform.

Many issues become more clear (or trivial) after this transformation*. (If x is discrete, some complications.)

The specification of a Bayesian prior pdf $p(\mu)$ for parameter μ is equivalent to the choice of the metric $f(\mu)$ in which the pdf is uniform. This is a *deep* issue, not always recognized as such by users of flat prior pdf's in HEP!

*And the inverse transformation provides for efficient M.C. generation of p(x) starting from RAN().

Bob Cousins, CMS, 2008

BAYES THEOREM

Bayes' Theorem

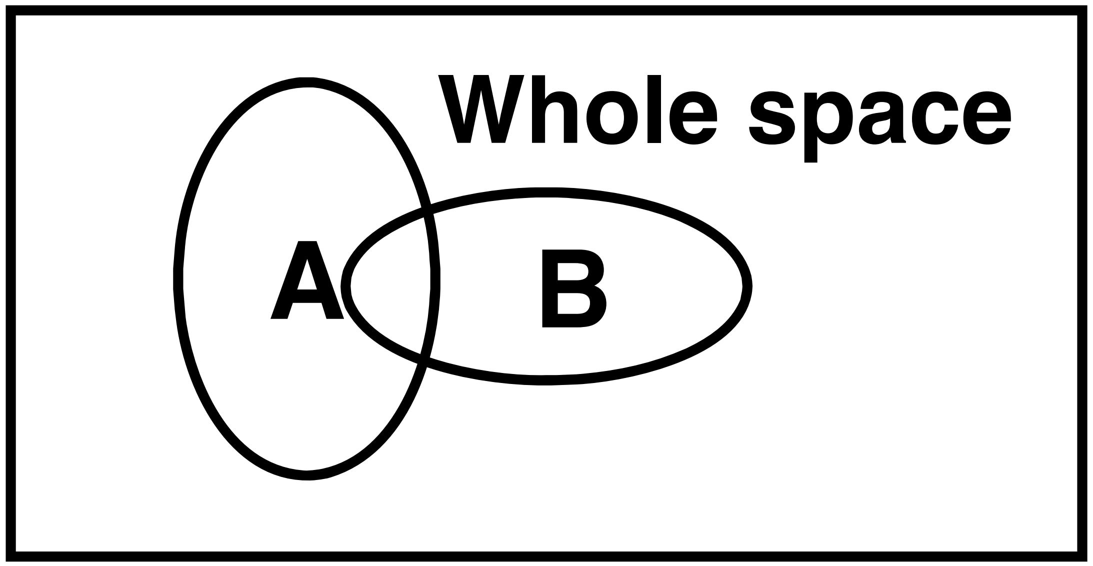
Bayes' theorem relates the conditional and marginal probabilities of events A & B

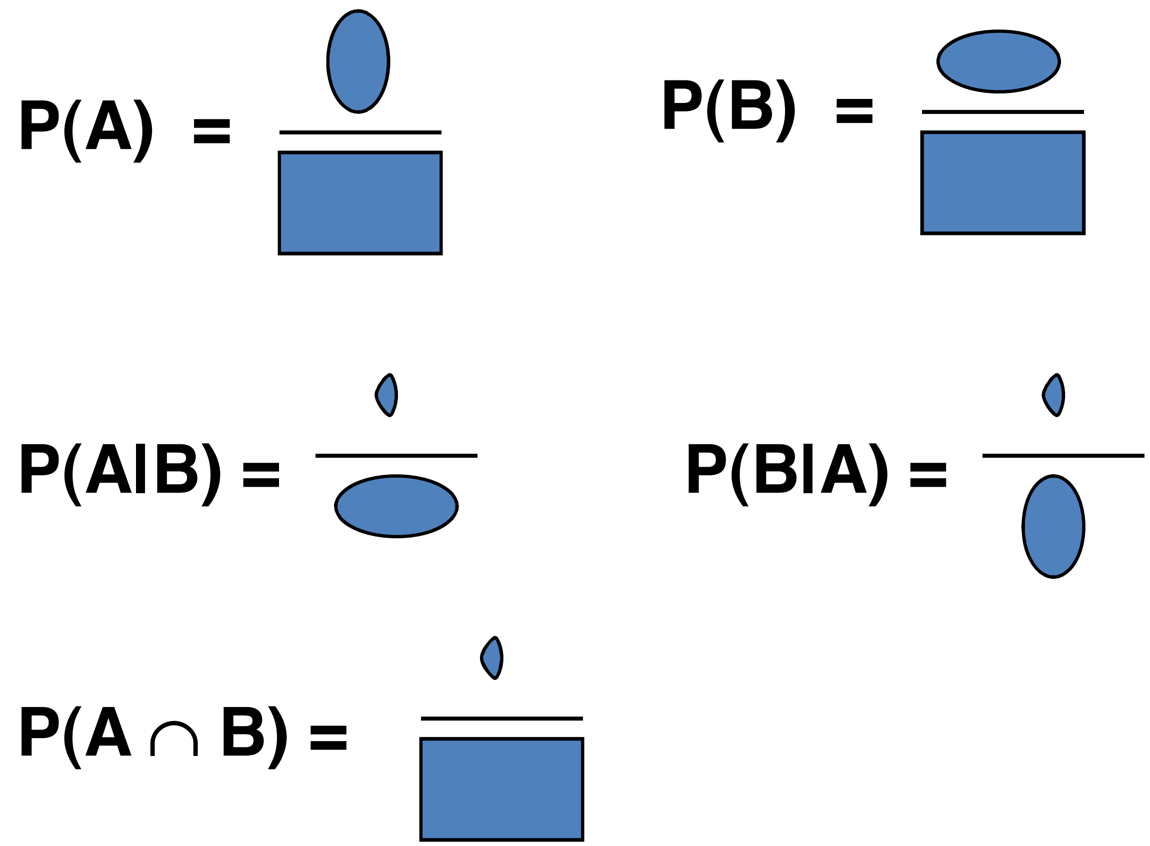
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

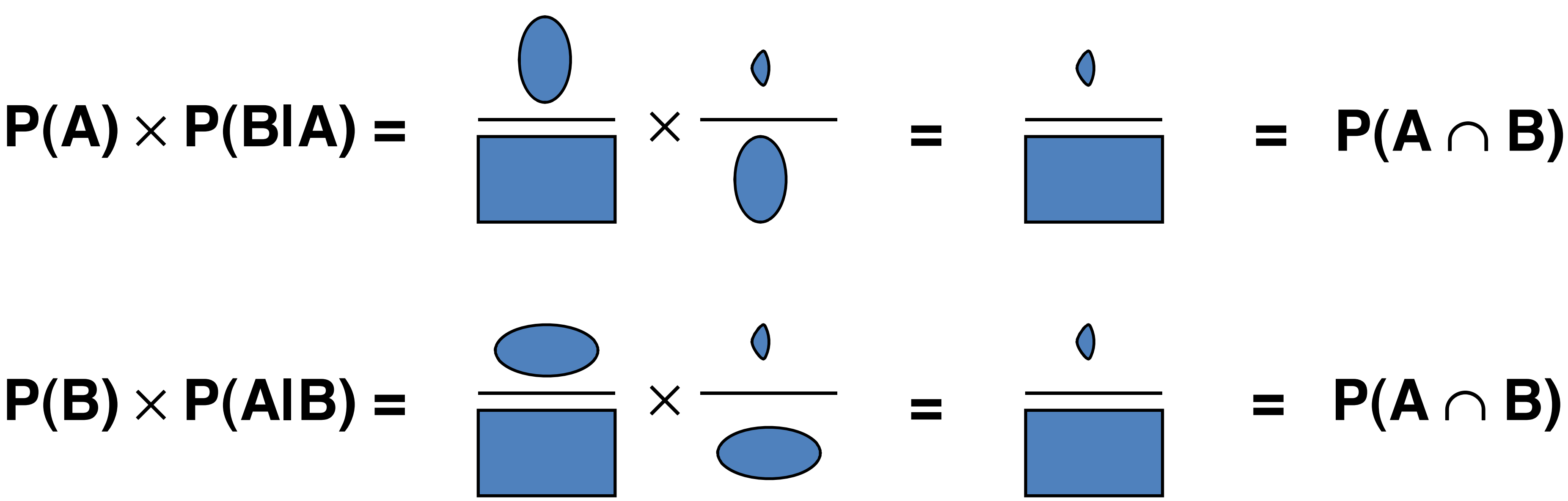
- **P(*A*)** is the prior probability. It is "prior" in the sense that it does not take into account any information about *B*.
- **P(*A*|*B*)** is the conditional probability of *A*, given *B*. It is also called the posterior probability because it is derived from or depends upon the specified value of *B*.
- **P(*B*|*A*)** is the conditional probability of *B* given *A*.
- **P(*B*)** is the prior or marginal probability of *B*, and acts as a normalizing constant.

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\mathcal{N}} \propto L(\theta)\pi(\theta)$$

P, Conditional P, and Derivation of Bayes' Theorem in Pictures





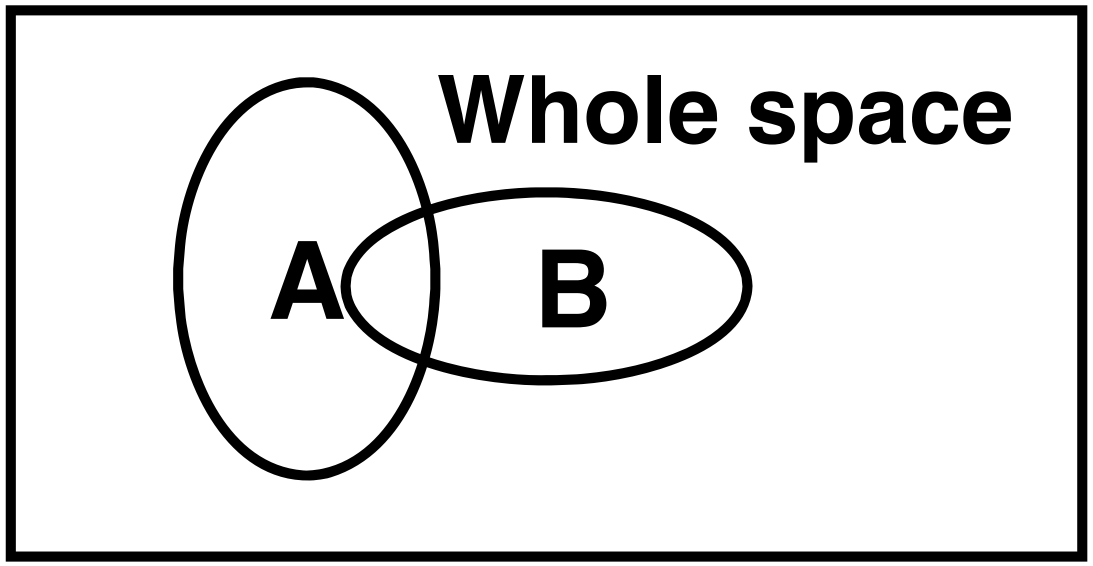


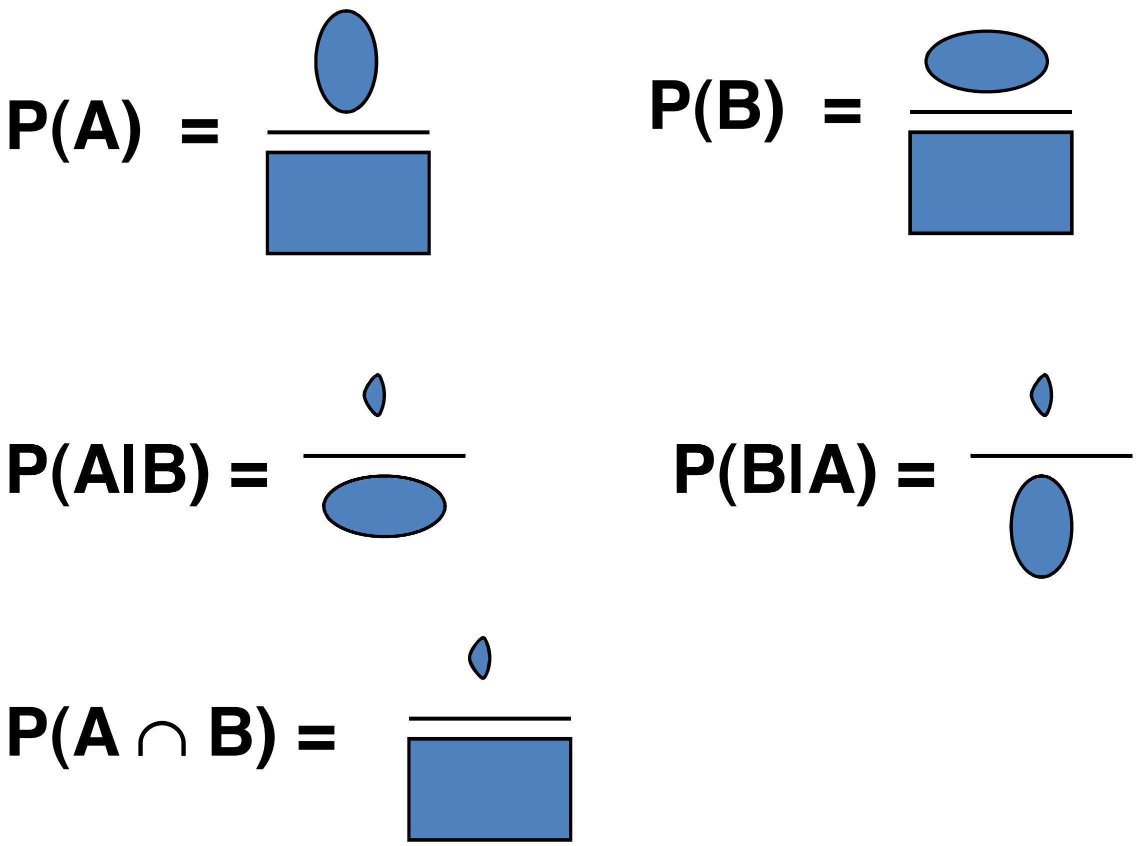
$\Rightarrow$ **P(B|A) = P(A|B) × P(B) / P(A)**

Bob Cousins, CMS, 2008

… IN PICTURES (FROM BOB COUSINS)

P, Conditional P, and Derivation of Bayes' Theorem in Pictures





Don't forget about "Whole space" Ω I will drop it from the notation typically, but occasionally it is important.

Bob Cousins, CMS, 2008

$\Rightarrow$ **P(B|A) = P(A|B) $\times$ P(B) / P(A)**

Louis's Example

P (Data;Theory) $\neq$ P (Theory;Data)

Theory = male or female

Data = pregnant or not pregnant

P (pregnant ; female) ~ 3%

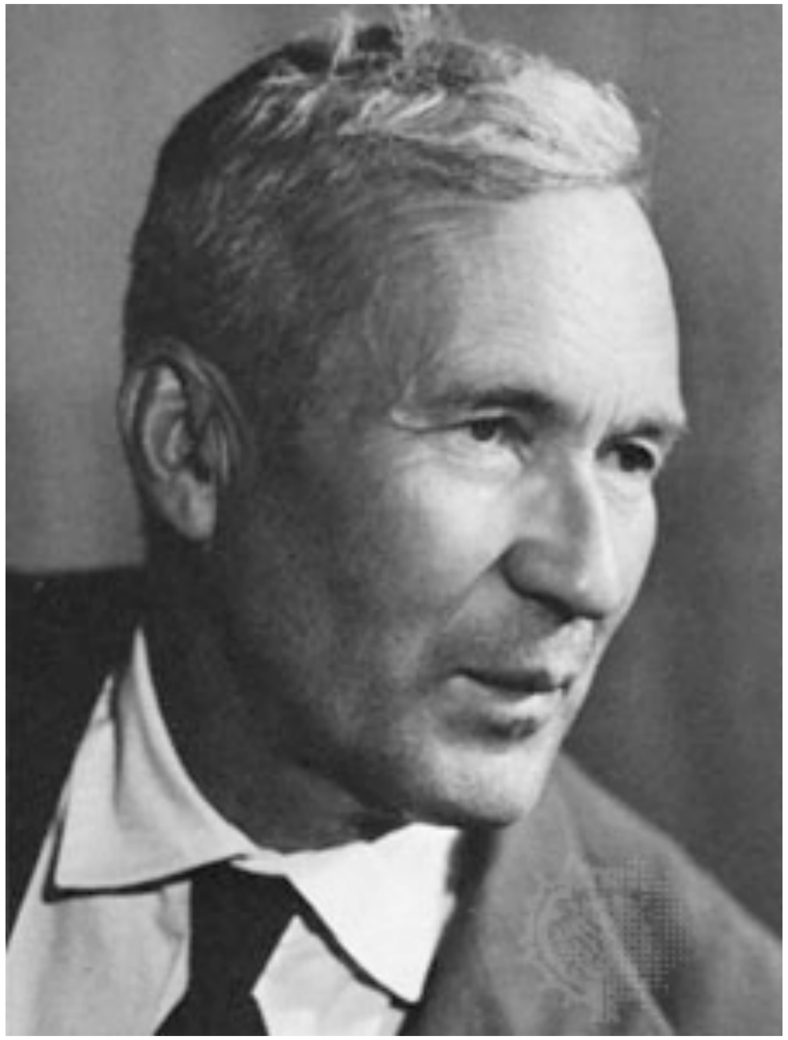
but

P (female ; pregnant) >>>3%

Axioms of Probability

These Axioms are a mathematical starting point for probability and statistics

1. probability for every element, E, is non-negative $P(E) \geq 0 \qquad \forall E \subseteq \mathcal{F} = 2^{\Omega}$
2. probability for the entire space of possibilities is 1 $P(\Omega) = 1.$
3. if elements E_i are disjoint, probability is additive $P(E_1 \cup E_2 \cup \cdots) = \sum_i P(E_i).$



Kolmogorov axioms (1933)

Consequences:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$$P(\Omega \setminus E) = 1 - P(E)$$

DIFFERENT DEFINITIONS OF PROBABILITY

Frequentist

- defined as limit of long term frequency
- probability of rolling a 3 := limit of (# rolls with 3 / # trials)
  - you don't need an infinite sample for definition to be useful
  - sometimes ensemble doesn't exist
    - eg. P(Higgs mass = 125 GeV), P(it will snow tomorrow)
- Intuitive if you are familiar with Monte Carlo methods
- compatible with orthodox interpretation of probability in Quantum Mechanics. Probability to measure spin projected on x-axis if spin of beam is polarized along +z

$$|\langle\rightarrow|\uparrow\rangle|^2 = \frac{1}{2}$$

Subjective Bayesian

- Probability is a degree of belief (personal, subjective)
  - can be made quantitative based on betting odds
  - most people's subjective probabilities are not ***coherent*** and do not obey laws of probability

http://plato.stanford.edu/archives/sum2003/entries/probability-interpret/#3.1

MEASUREMENT / ESTIMATORS

Given some model $f(x|\alpha)$ and a set of observations $\{x_i\}$ often one wants to estimate the true value of α (assuming the model is true).

An **estimator** is function of the data written $\hat{\alpha}(x_1, \ldots x_n)$

- Since the data are random, so is the resulting estimate
- often it is just written $\hat{\alpha}$, where the x-dependence is implicit
- one can compute expectation of the estimator

$$E[\hat{\alpha}(x)|\alpha] = \int \hat{\alpha}(x) f(x|\alpha) dx$$

Properties of estimators:

- **bias** $E[\hat{\alpha}(x)|\alpha] - \alpha$ (unbiased means bias=0)
- **variance** $E[(\hat{\alpha}(x) - \bar{\alpha})^2|\alpha] = \int (\hat{\alpha}(x) - \bar{\alpha})^2 f(x|\alpha) dx$
- **asymptotic bias** limit of bias with infinite observations

Maximum likelihood estimators

There are many different possible estimators, but the most well-known and well-studied is the maximum likelihood estimator (MLE)

$$\hat{\alpha}(x) = \text{argmax}_\alpha L(\alpha) = \text{argmax}_\alpha f(x|\alpha)$$

This is just the value of α that maximizes the likelihood

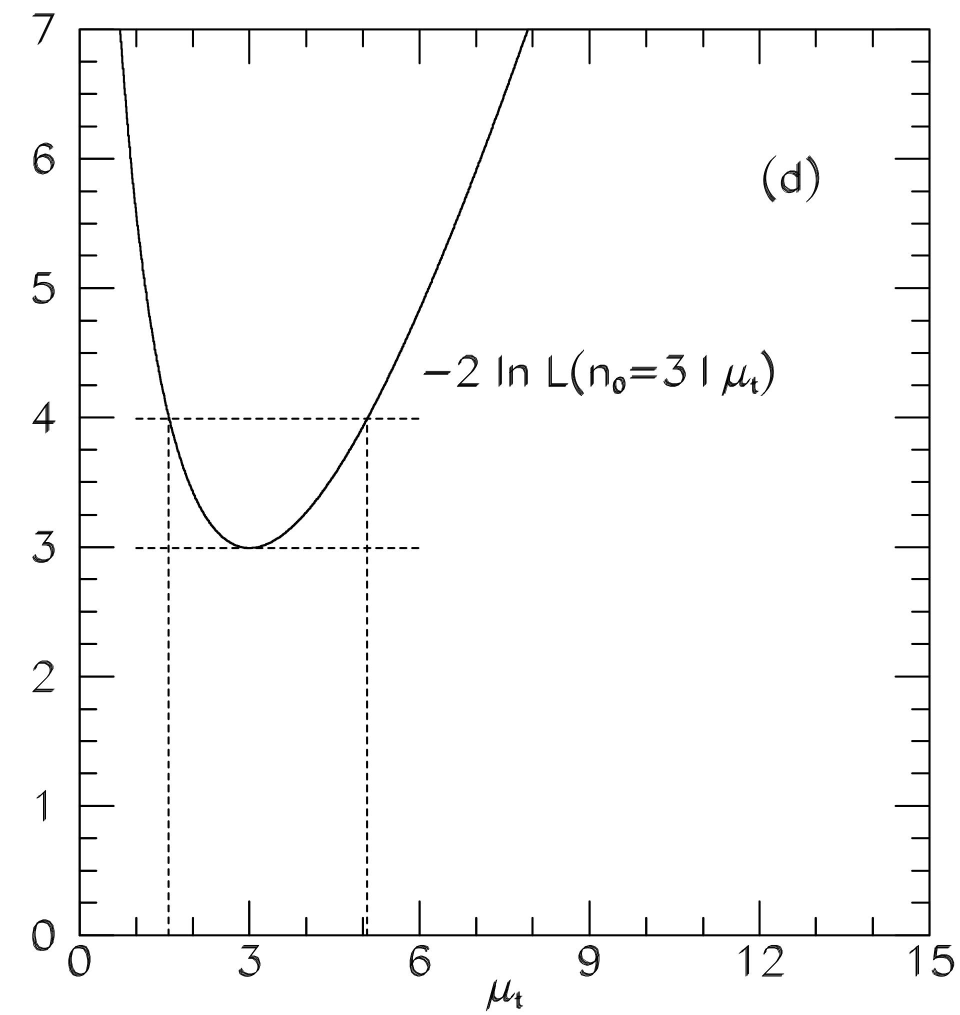


Figure from R. Cousins, Am. J. Phys. 63 398 (1995)

Example: the Poisson distribution

$$Pois(n|\mu) = \mu^n \frac{e^{-\mu}}{n!}$$

Maximizing $L(\mu)$ is the same as minimizing -ln $L(\mu)$

$$-\frac{d}{d\mu} \ln L(\mu)\big|_{\hat{\mu}} = 0 = \frac{d}{d\mu}\left(\mu - n\ln\mu + \underbrace{\ln n!}_{\text{const}}\right) = 1 - \frac{n}{\mu}$$

$$\Rightarrow \hat{\mu} = n$$

In this case, the MLE is unbiased b/c E[n]=μ

A SECOND EXAMPLE

Consider a set of observations $\{x_i\}$ and we want to estimate the mean of a Gaussian with known σ

which gives
$$G(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$-\frac{d}{d\mu} \ln L(\mu)\Big|_{\hat{\mu}} = 0 = \frac{d}{d\mu}\left(\sum_i \frac{(x_i-\mu)^2}{2\sigma^2} + \underbrace{\ln\sqrt{2\pi}\sigma}_{\text{const}}\right) = \sum_i \frac{(x_i-\mu)}{\sigma^2}$$

$$\Rightarrow \hat{\mu} = \frac{1}{N}\sum_i x_i$$ (an unbiased estimator) .

However, the MLE $\hat{\sigma}^2 = \frac{1}{N}\sum_i (x_i-\mu)^2$ is biased

It can be shown that $\hat{\sigma}^2 = \frac{1}{N-1}\sum_i (x_i-\mu)^2$ is unbiased

Thus, the MLE is **asymptotially unbiased .**

Note: if $\hat{\sigma}^2$ is an unbiased estimate of σ^2, then $\sqrt{\{\hat{\sigma}^2\}}$ is a biased estimate of σ.

COVARIANCE AND CORRELATION

Define covariance cov[x,y] (also use matrix notation V_{xy}) as

$$\mathrm{cov}[x,y] = E[xy] - \mu_x\mu_y = E[(x-\mu_x)(y-\mu_y)]$$

Correlation coefficient (dimensionless) defined as

$$\rho_{xy} = \frac{\mathrm{cov}[x,y]}{\sigma_x\sigma_y}$$

If x, y, independent, i.e., $f(x,y) = f_x(x)f_y(y)$, then

$$E[xy] = \int\int xy\, f(x,y)\, dx dy = \mu_x\mu_y$$

$\rightarrow$ $\mathrm{cov}[x,y] = 0$ $\quad$ x and y, ‘uncorrelated’

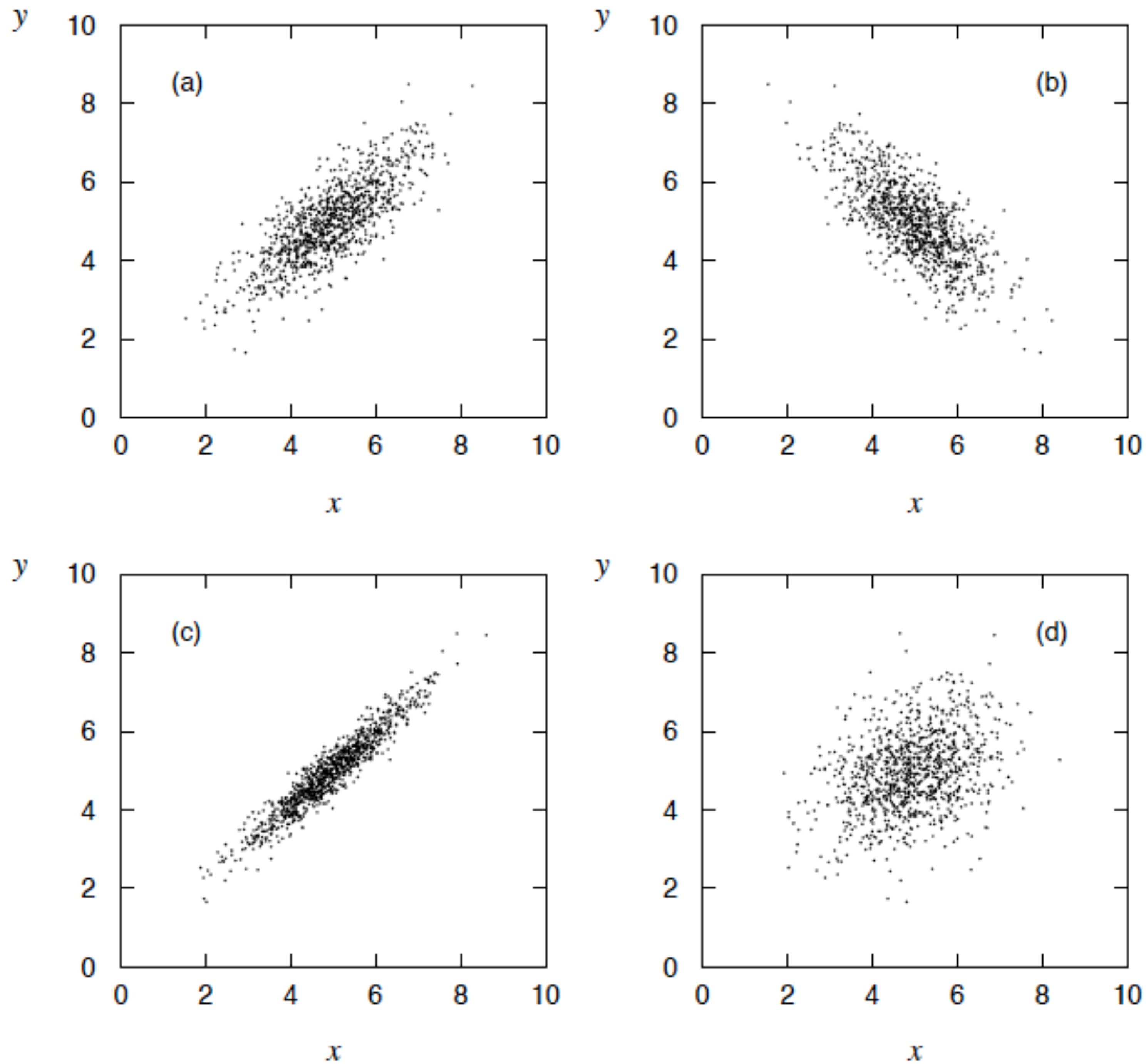
N.B. converse not always true.

[G. Cowan]

CORRELATION (CONT.)

$\rho = 0.75$

$\rho = -0.75$

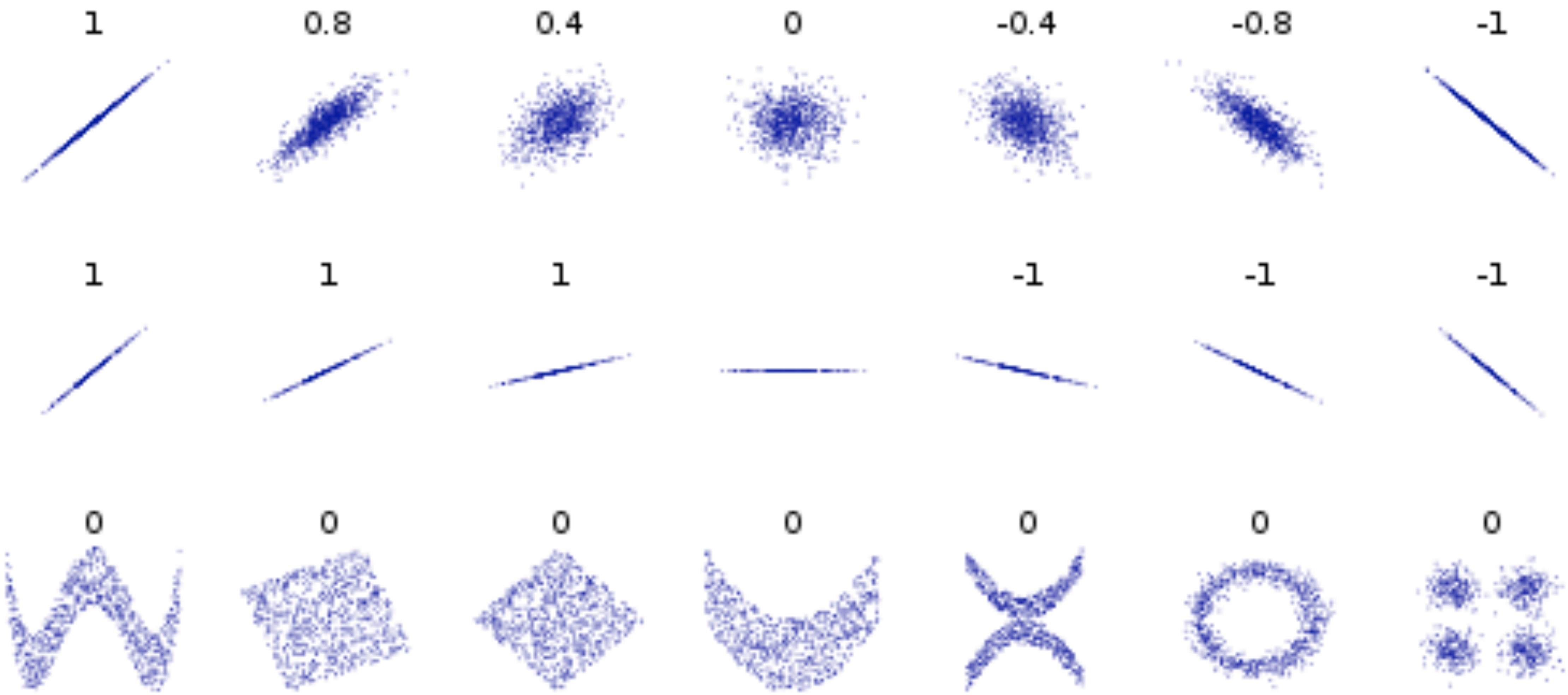


$\rho = 0.95$

$\rho = 0.25$

[G. Cowan]

CORRELATION (CONT.)



http://en.wikipedia.org/wiki/Correlation_and_dependence

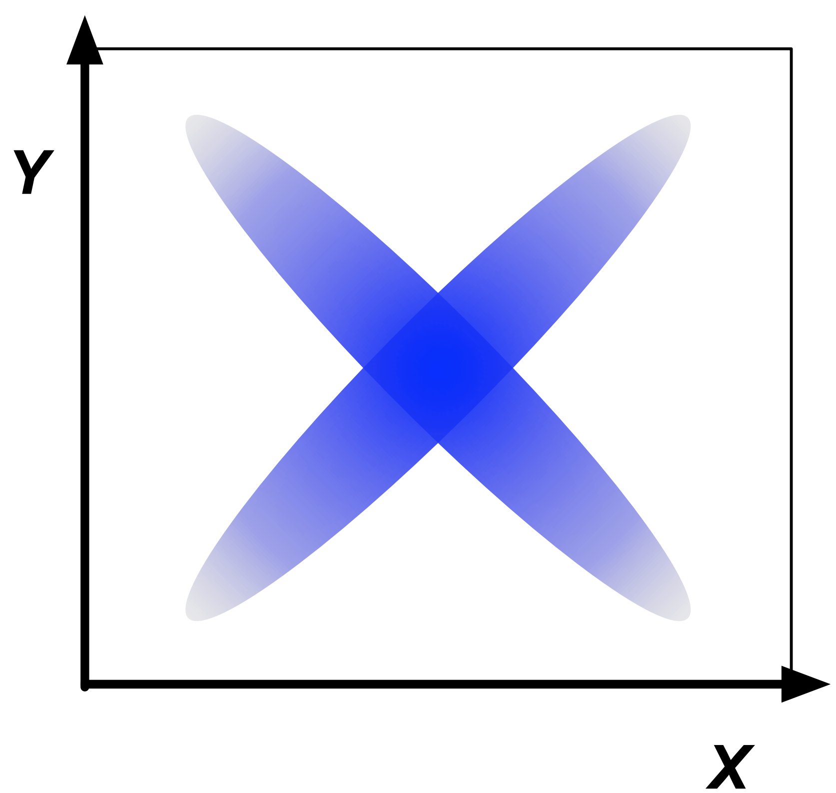
Mutual Information

Mutual Information is a more general notion of 'correlation'

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p_1(x)\,p_2(y)}\right),$$

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

- it is symmetric: I(X;Y) = I(Y;X)
- if and only if X,Y totally independent: I(X;Y)=0
- possible for X,Y to be uncorrelated, but not independent



Mutual Information doesn't seem to be used much within HEP, but it seems quite useful

We introduced Bias and Variance of estimators

$$\mathrm{Var}[\hat{\mu}|\mu] = E[(\hat{\mu} - E[\mu|\mu])^2]\,|\mu]$$

Most physicist are allergic to the idea of a biased estimator

- try to find unbiased estimator with smallest variance
- hence importance of Cramér-Rao bound

But what if we just want to minimize the mean-squared error?

$$MSE[\hat{\mu}|\mu] = E[(\hat{\mu} - \mu)^2]\,|\mu]$$

it decomposes like this

$$MSE[\hat{\mu}|\mu] = \mathrm{Var}[\hat{\mu}|\mu] + (\mathrm{Bias}[\hat{\mu}|\mu])^2$$

So it encodes some relative weight to bias and variance. Think harder!

Cramér-Rao Bound

The minimum variance bound on an estimator is given by the Cramér-Rao inequality:

- simple univariate case:

$$\mathrm{Var}[\hat{\theta}|\theta] = E[(\hat{\theta} - E[\theta|\theta])^2] \, |\theta]$$

- For an unbiased estimator the Cramér-Rao bound states

$$\mathrm{Var}[\hat{\theta}|\theta] \geq \frac{1}{I(\theta)}$$

- where $I(\theta)$ is the Fisher information

$$(\mathcal{I}(\theta))_{i,j} = \mathrm{E}\left[\frac{\partial}{\partial\theta_i} \ln f(X;\theta) \frac{\partial}{\partial\theta_j} \ln f(X;\theta) \middle| \theta \right].$$

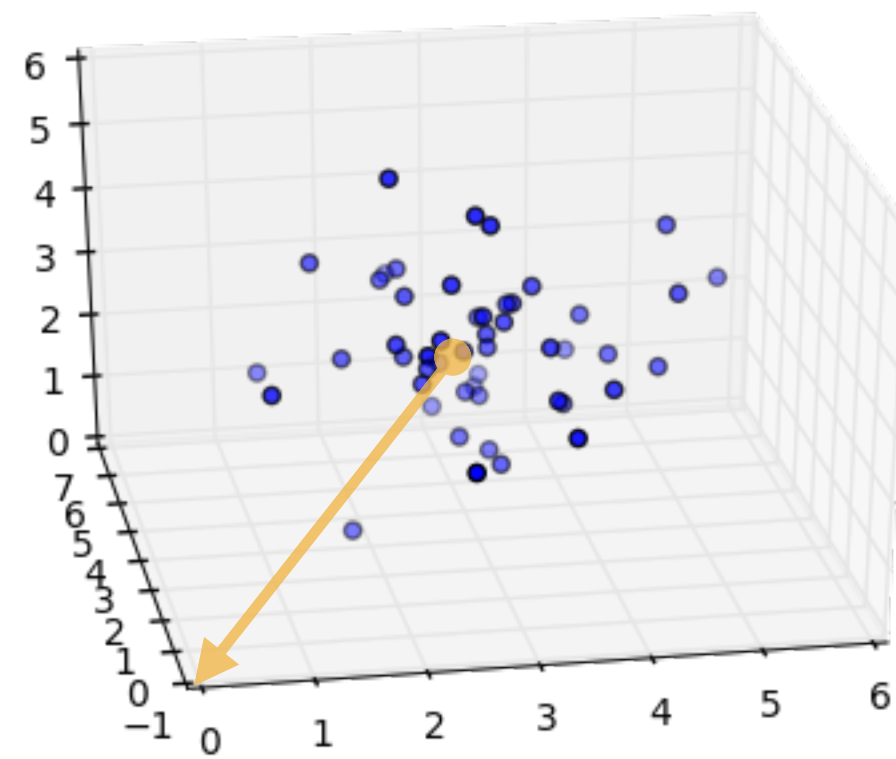
- General form for multiple parameters:

$$\mathrm{cov}[\hat{\theta}|\theta]_{ij} \geq I^{-1}_{ij}(\theta)$$

Maximum Likelihood Estimators *asymptotically* reach this bound

Consider a standard multivariate Gaussian distribution for $\vec{x}$ in n dimensions centered around $\vec{\mu}$

$$f(\vec{x}|\vec{\mu}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_i)^2}{2}\right) .$$



Goal: minimize mean-squared error

$$MSE[\hat{\vec{\mu}}] = E[||\hat{\vec{\mu}} - \vec{\mu}||^2])$$

MLE (unbiased)

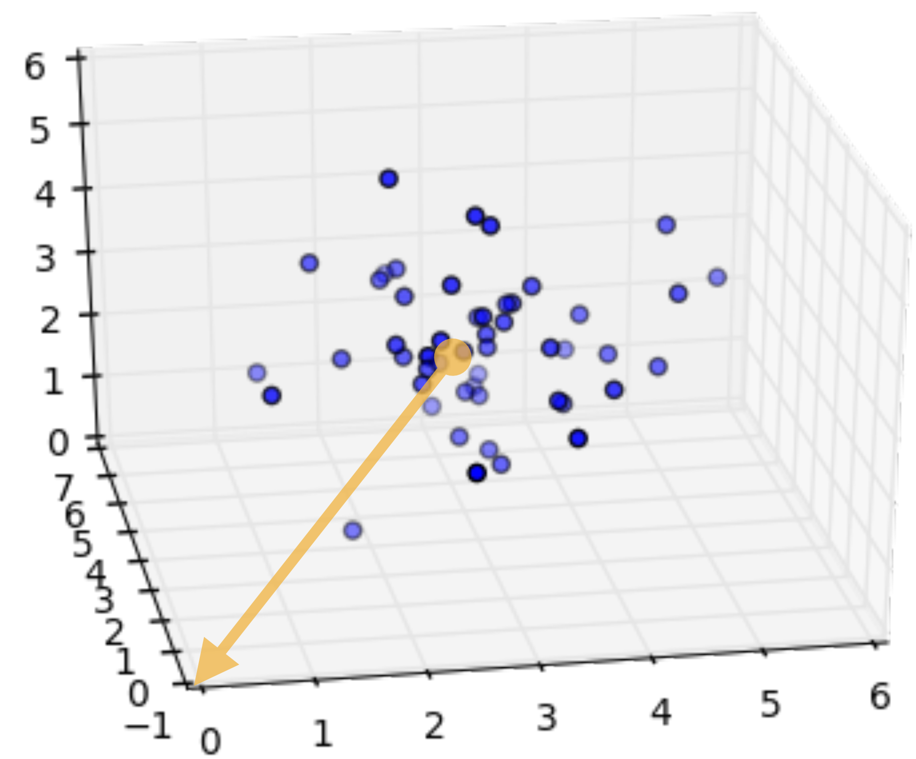
$$\hat{\vec{\mu}}_{MLE} = \overline{x} = \frac{1}{m}\sum_{j=1}^{m} \vec{x}_j$$

James-Stein (weird)

$$\hat{\mu}_{JS} = \left(1 - \frac{n-2}{||\bar{x}||^2}\right)\bar{x}$$

The James-Stein estimator seems like a horrible suggestion

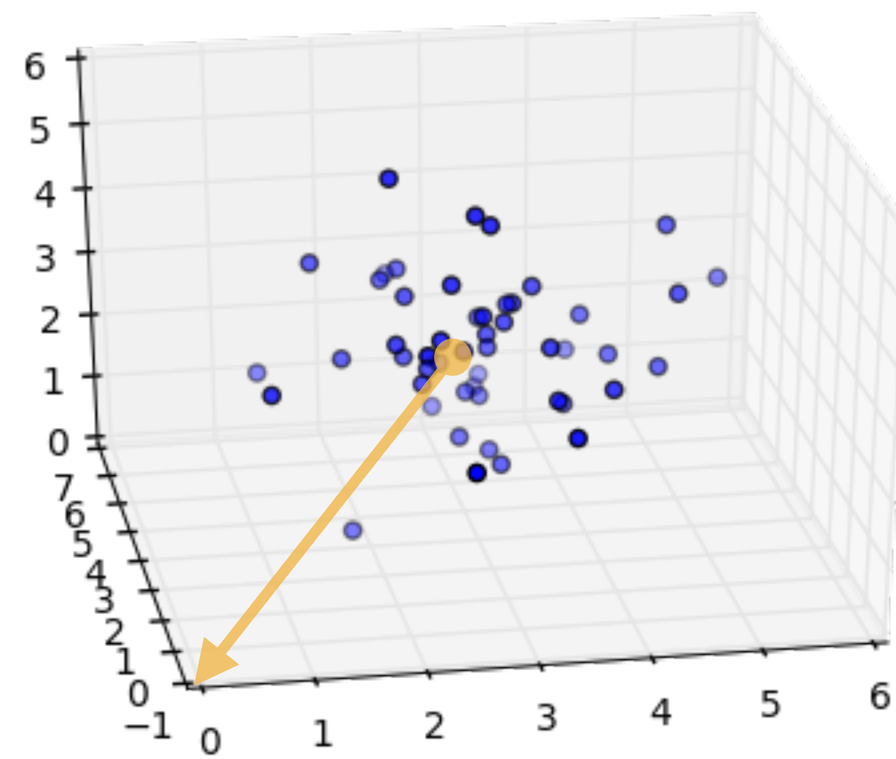
$$\hat{\mu}_{JS} = \left(1 - \frac{n-2}{||\bar{x}||^2}\right)\bar{x}$$



- clearly biased (MLE is not)
- shifts towards origin is not translationally invariant
  $x \rightarrow x' = x+\Delta$

The James-Stein estimator seems like a horrible suggestion

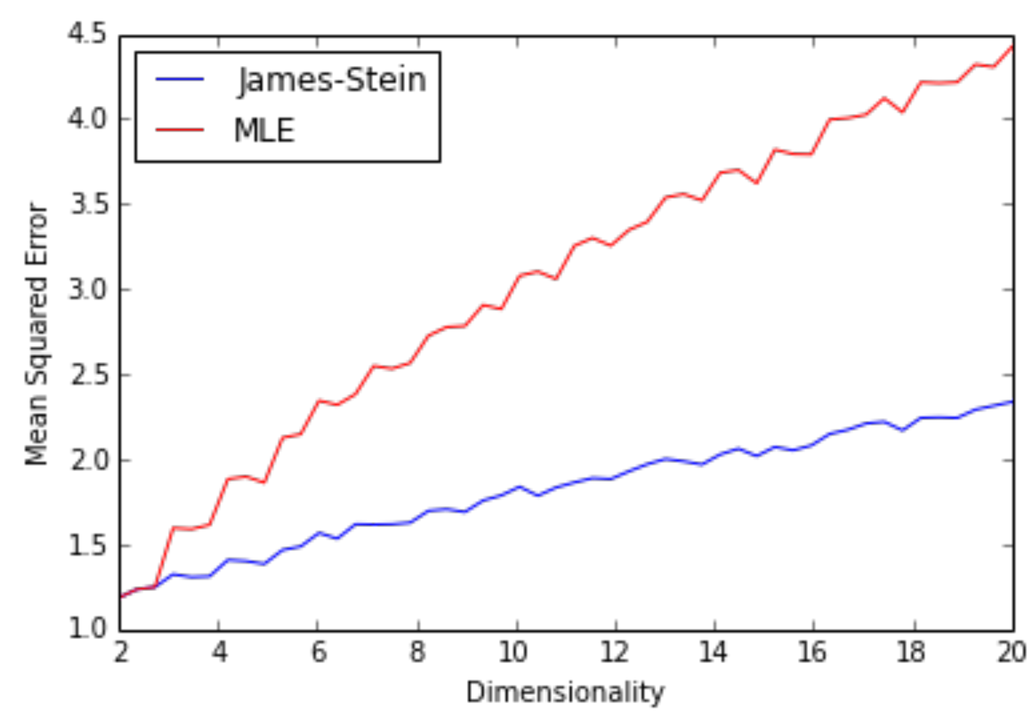
$$\hat{\mu}_{JS} = \left(1 - \frac{n-2}{||\bar{x}||^2}\right)\bar{x}$$



- clearly biased (MLE is not)
- shifts towards origin is not translationally invariant
  x ➞ x' = x+Δ

Yet, it has smaller mean squared error than MLE for n>2 !

- it "dominates" the MLE



We introduced Bias and Variance of estimators

$$\mathrm{Var}[\hat{\mu}|\mu] = E[(\hat{\mu} - E[\mu|\mu])^2]\,|\mu]$$

Most physicist are allergic to the idea of a biased estimator

- try to find unbiased estimator with smallest variance
- hence importance of Cramér-Rao bound

But what if we just want to minimize the mean-squared error?

$$MSE[\hat{\mu}|\mu] = E[(\hat{\mu} - \mu)^2]\,|\mu]$$

it decomposes like this

$$MSE[\hat{\mu}|\mu] = \mathrm{Var}[\hat{\mu}|\mu] + (\mathrm{Bias}[\hat{\mu}|\mu])^2$$

So it encodes some relative weight to bias and variance. Think harder!

STATISTICAL DECISION THEORY IN 1 SLIDE

Θ - States of nature; X - possible observations; A - action to be taken

$f(x|\theta)$ - statistical model; $\pi(\theta)$ - prior

$\delta: X \rightarrow A$ - **decision rule** (take some action based on observation)

$L: \Theta \times A \rightarrow \mathbb{R}$ - **loss function**, real-valued function true parameter and action

$R(\theta,\delta) = E_{f(x|\theta)}[L(\theta, \delta)]$ - **risk**

- A decision δ^* rule **dominates** a decision rule δ if and only if $R(\theta,\delta^*) \leq R(\theta,\delta)$ for all θ, and the inequality is strict for some θ.
- A decision rule is **admissible** if and only if no other rule dominates it; otherwise it is inadmissible

$r(\pi, \delta) = E_{\pi(\theta)}[R(\theta,\delta)]$ - **Bayes risk** (expectation over θ w.r.t. prior and possible observations)

$\rho(\pi, \delta \mid x) = E_{\pi(\theta|x)}[L(\theta,\delta(x))]$ - **expected loss** (expectation over θ w.r.t. posterior $\pi(\theta|x)$)

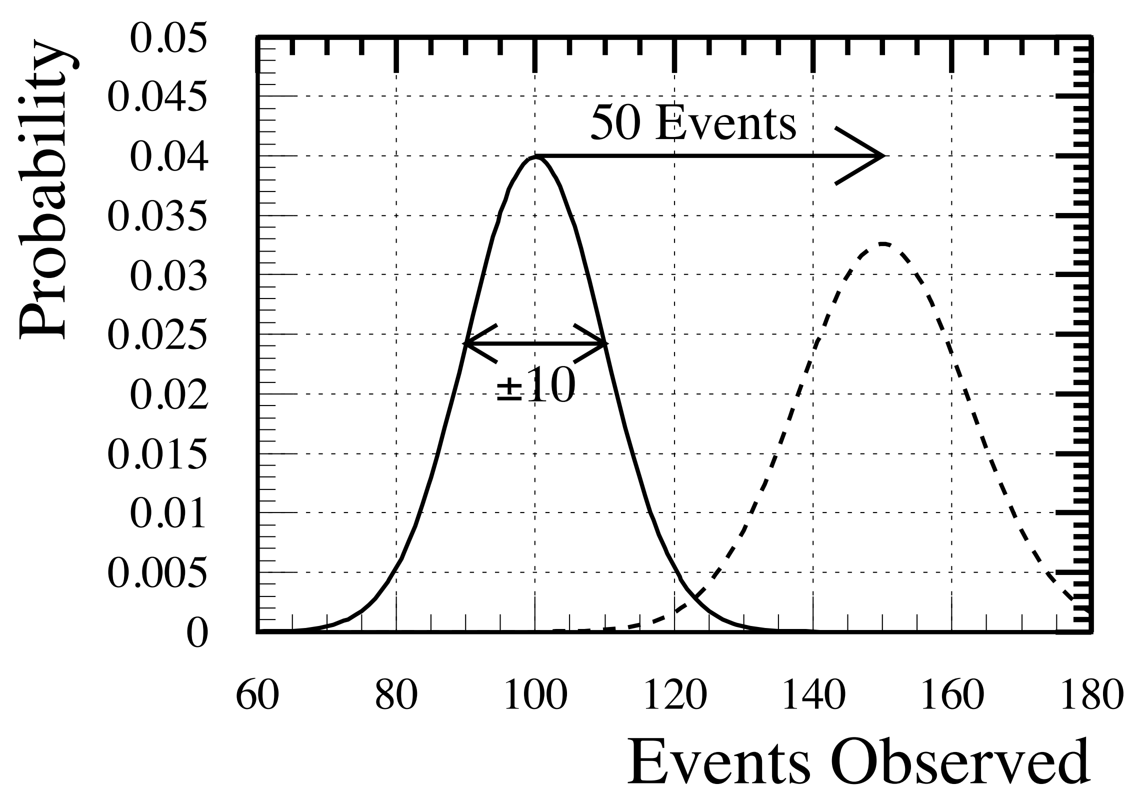
- δ' is a (generalized) Bayes rule if it minimizes the expected loss
- under mild conditions every admissible rule is a (generalized) Bayes rule (**with respect to some prior** —possibly an improper one—that favors distributions where that rule achieves low risk). Thus, in frequentist decision theory it is sufficient to consider only (generalized) Bayes rules.
- Conversely, while Bayes rules with respect to proper priors are virtually always admissible, generalized Bayes rules corresponding to improper priors need not yield admissible procedures. Stein's example is one such famous situation.

HYPOTHESIS TESTING

HYPOTHESIS TESTING

One of the most common uses of statistics in particle physics is Hypothesis Testing (e.g. for discovery of a new particle)

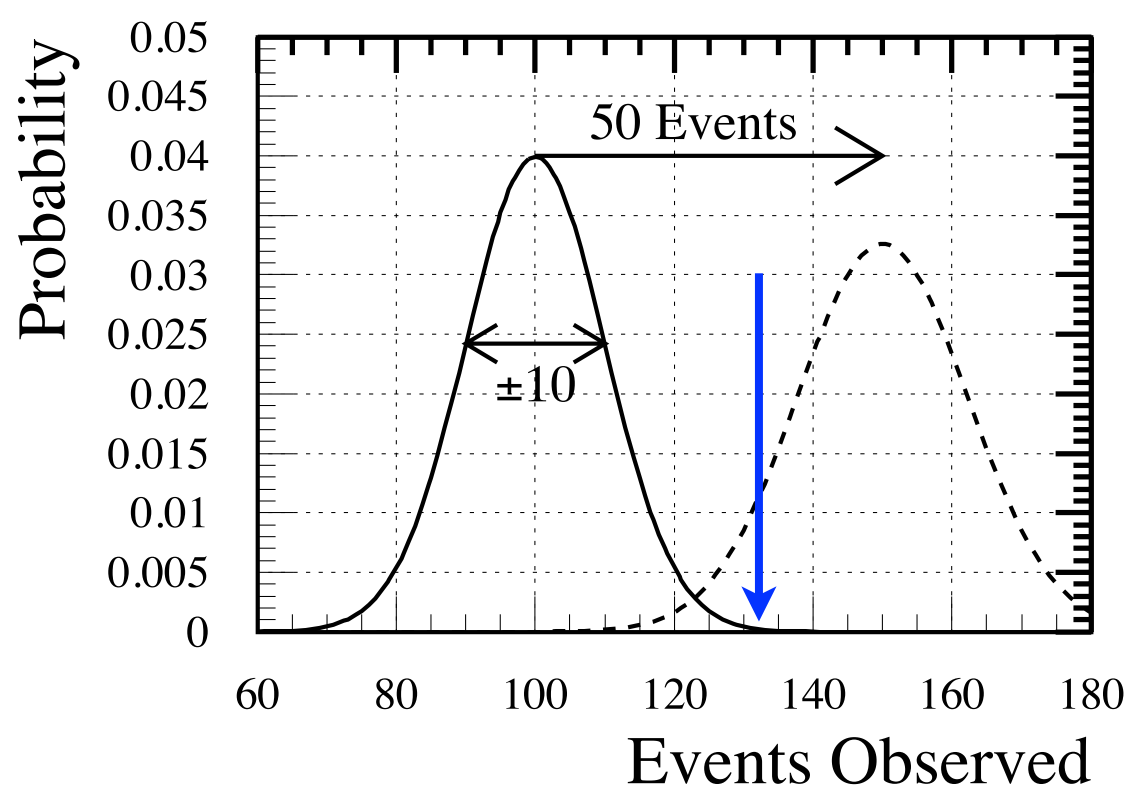
- assume one has pdf for data under two hypotheses:
  - Null-Hypothesis, H_0: eg. background-only
  - Alternate-Hypothesis H_1: eg. signal-plus-background
- one makes a measurement and then needs to decide whether to **reject** or **accept** H_0



Hypothesis testing

One of the most common uses of statistics in particle physics is Hypothesis Testing (e.g. for discovery of a new particle)

- assume one has pdf for data under two hypotheses:
  - Null-Hypothesis, H_0: eg. background-only
  - Alternate-Hypothesis H_1: eg. signal-plus-background
- one makes a measurement and then needs to decide whether to **reject** or **accept** H_0



Before we can make much progress with statistics, we need to decide what it is that we want to do.

- first let us define a few terms:
  - Rate of Type I error α
  - Rate of Type II β
  - Power = $1 - \beta$

		Actual condition	
		Guilty	Not guilty
Decision	Verdict of 'guilty'	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

Before we can make much progress with statistics, we need to decide what it is that we want to do.

- first let us define a few terms:
  - Rate of Type I error α
  - Rate of Type II β
  - Power = $1 - \beta$

		Actual condition	
		Guilty	Not guilty
Decision	Verdict of 'guilty'	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

Treat the two hypotheses asymmetrically

- the Null is special.
  - Fix rate of Type I error, call it “the size of the test”

Before we can make much progress with statistics, we need to decide what it is that we want to do.

- first let us define a few terms:
  - Rate of Type I error α
  - Rate of Type II β
  - Power = $1 - \beta$

		Actual condition	
		Guilty	Not guilty
Decision	Verdict of 'guilty'	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

Treat the two hypotheses asymmetrically

- the Null is special.
  - Fix rate of Type I error, call it "the size of the test"

Now one can state "a well-defined goal"

- Maximize power for a fixed rate of Type I error

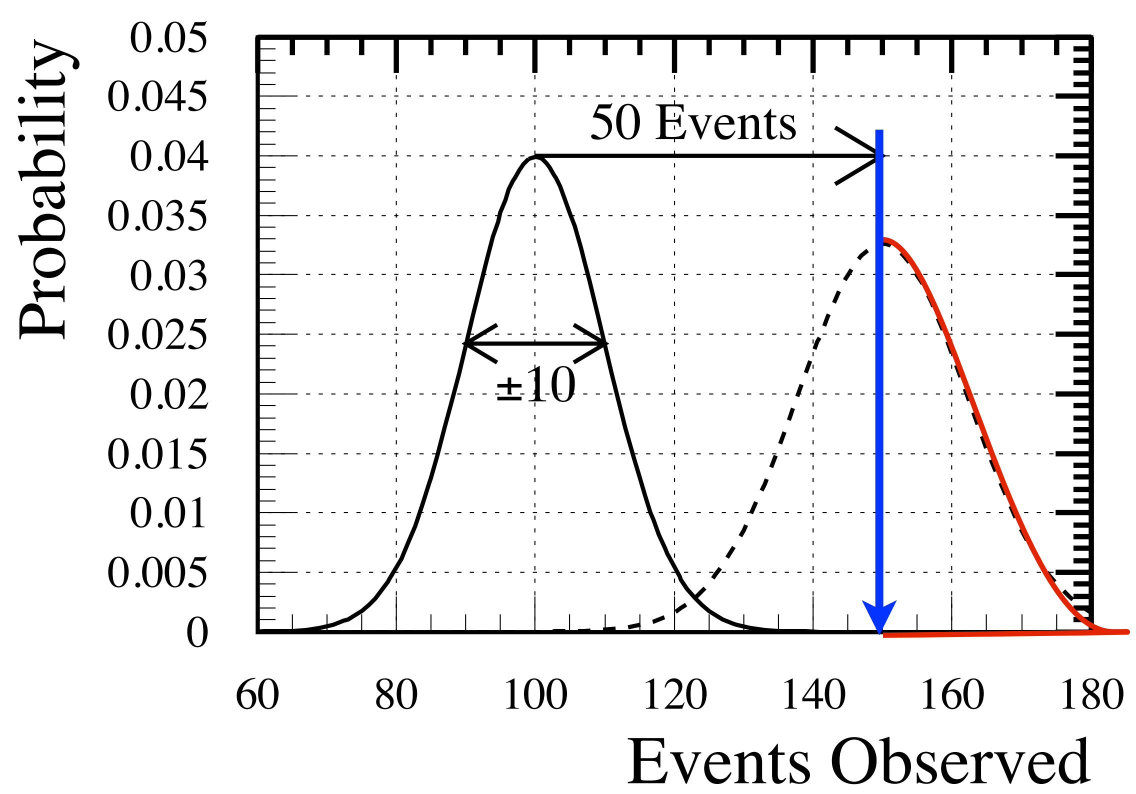
HYPOTHESIS TESTING

The idea of a "5σ" discovery criteria for particle physics is really a conventional way to specify the size of the test

- usually 5σ corresponds to $\alpha = 2.87 \cdot 10^{-7}$
  - eg. a very small chance we reject the standard model

In the simple case of number counting it is obvious what region is sensitive to the presence of a new signal

- but in higher dimensions it is not so easy

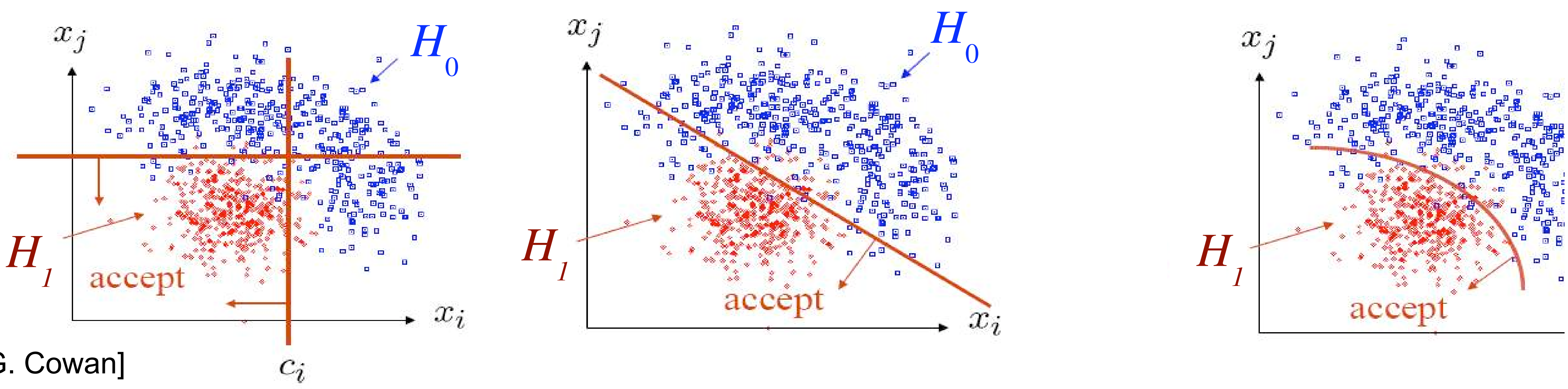


The idea of a "5σ" discovery criteria for particle physics is really a conventional way to specify the size of the test

- usually 5σ corresponds to $\alpha = 2.87 \cdot 10^{-7}$
  - eg. a very small chance we reject the standard model

In the simple case of number counting it is obvious what region is sensitive to the presence of a new signal

- but in higher dimensions it is not so easy



[G. Cowan]

The Neyman-Pearson Lemma

In 1928-1938 Neyman & Pearson developed a theory in which one must consider competing Hypotheses:

- the Null Hypothesis H_0 (background only)
- the Alternate Hypothesis H_1 (signal-plus-background)

Given some probability that we wrongly reject the Null Hypothesis

$$\alpha = P(x \notin W | H_0)$$

(Convention: if data falls in W then we accept H_0)

Find the region W such that we minimize the probability of wrongly accepting the H_0 (when H_1 is true)

$$\beta = P(x \in W | H_1)$$

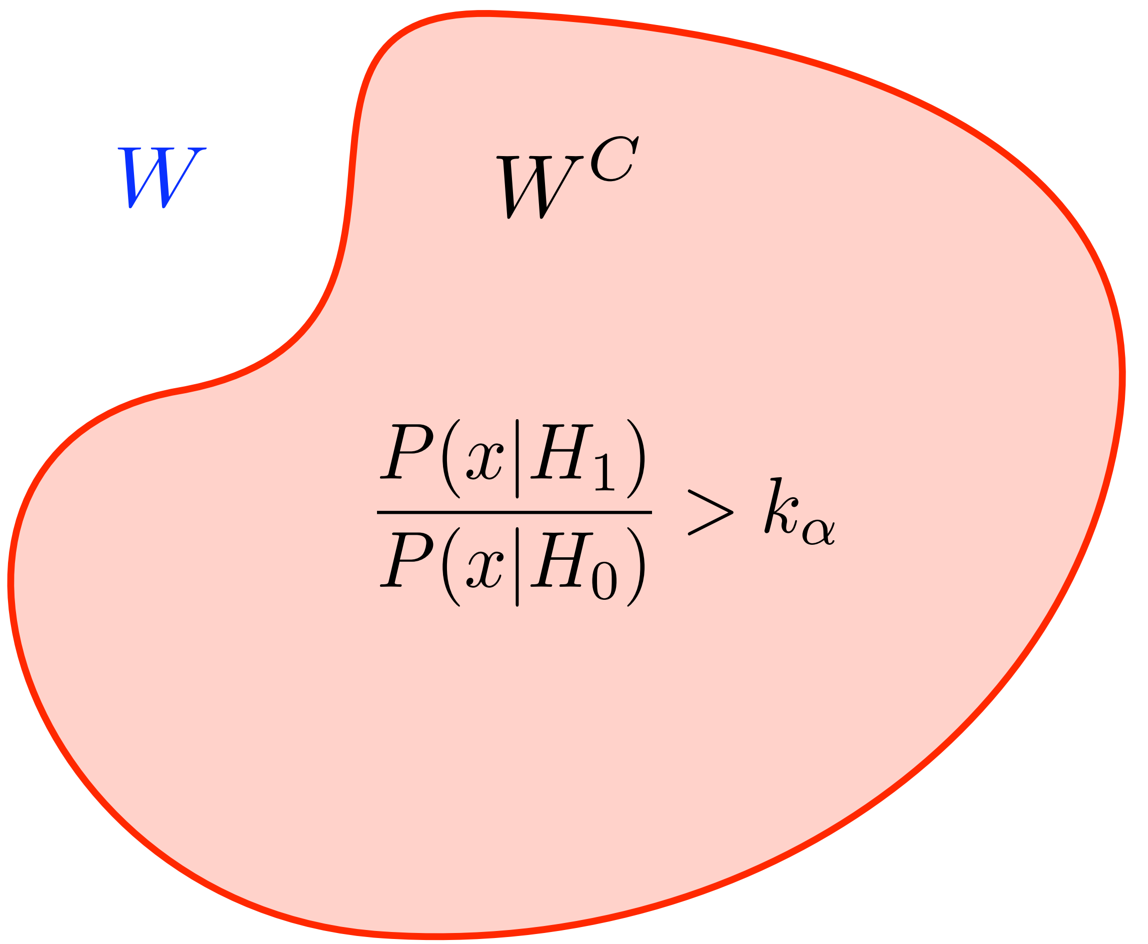
The region W that minimizes the probability of wrongly accepting H_0 is just a contour of the Likelihood Ratio

$$\frac{P(x|H_1)}{P(x|H_0)} > k_\alpha$$

Any other region of the same size will have less power

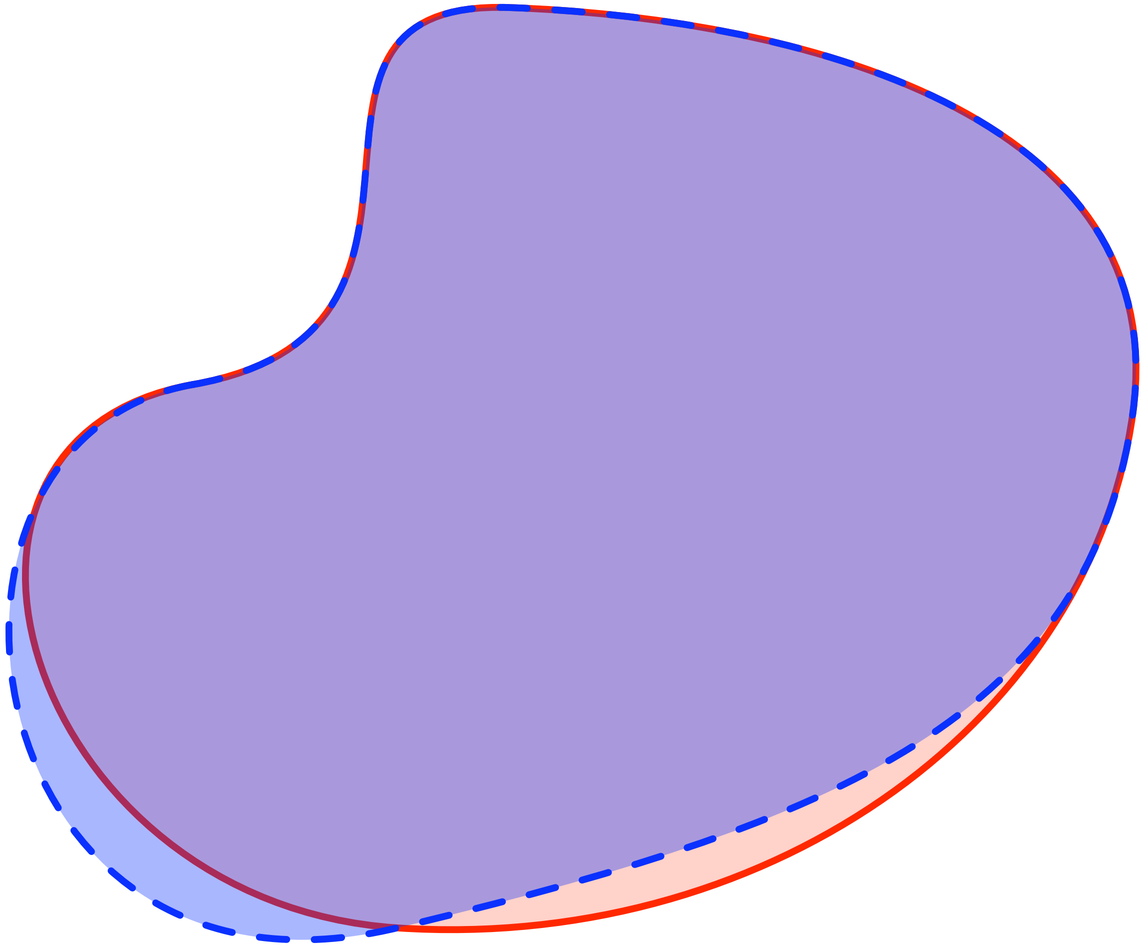
The likelihood ratio is an example of a **Test Statistic**, eg. a real-valued function that summarizes the data in a way relevant to the hypotheses that are being tested

A SHORT PROOF OF NEYMAN-PEARSON



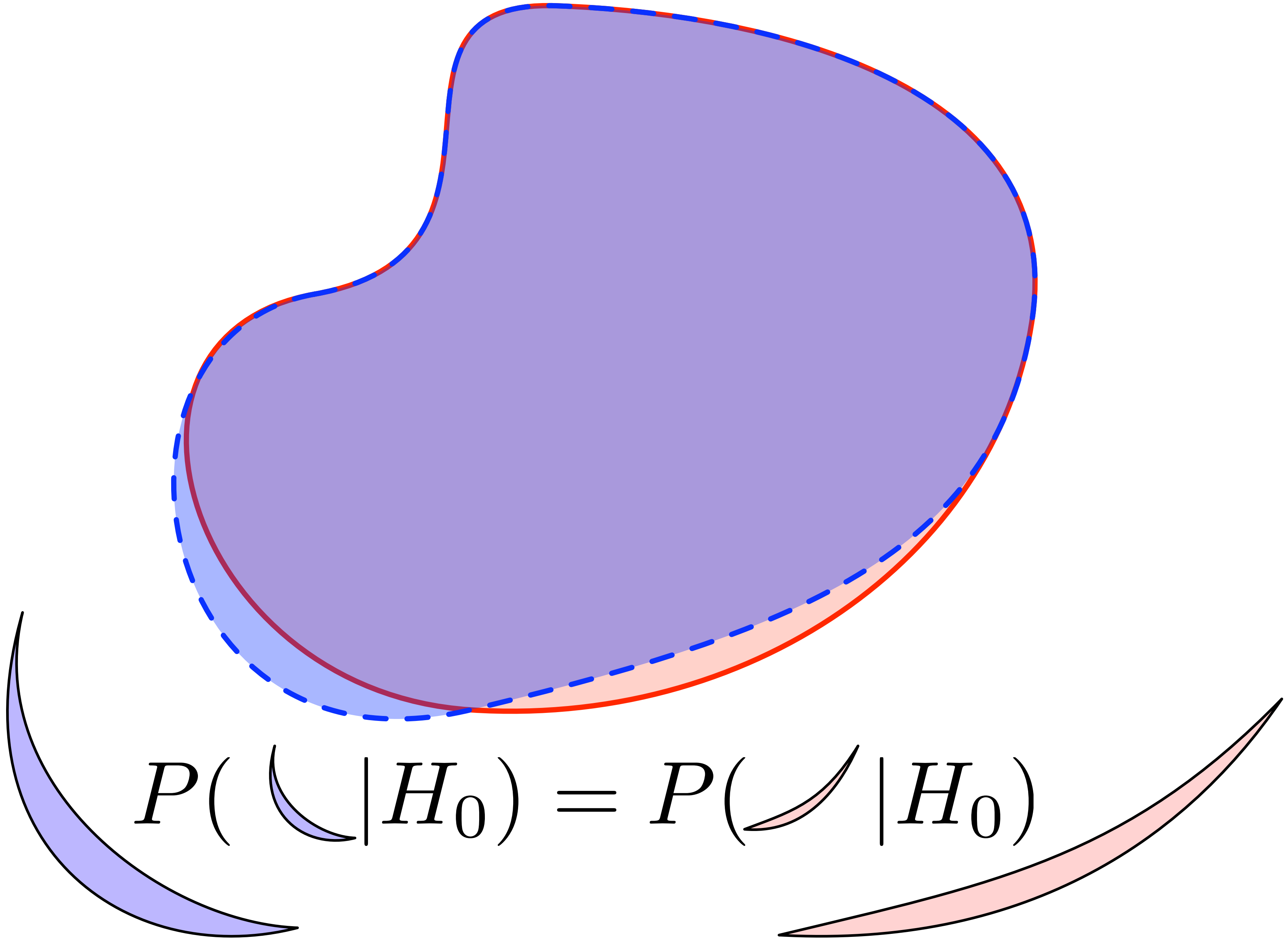
Consider the contour of the likelihood ratio that has size a given size (eg. probability under H_0 is 1-α)

A short proof of Neyman-Pearson



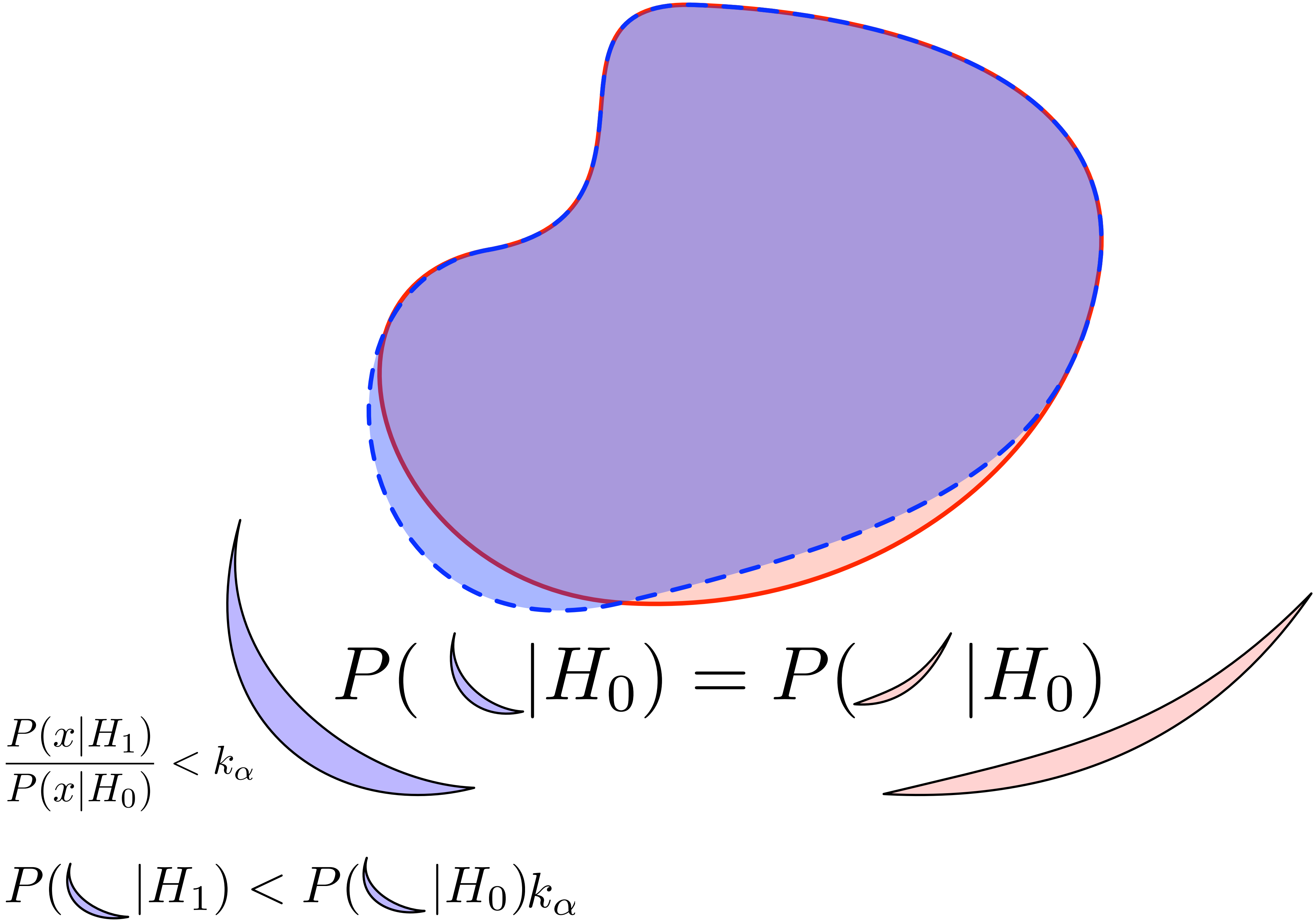
Now consider a variation on the contour that has the same size

A SHORT PROOF OF NEYMAN-PEARSON



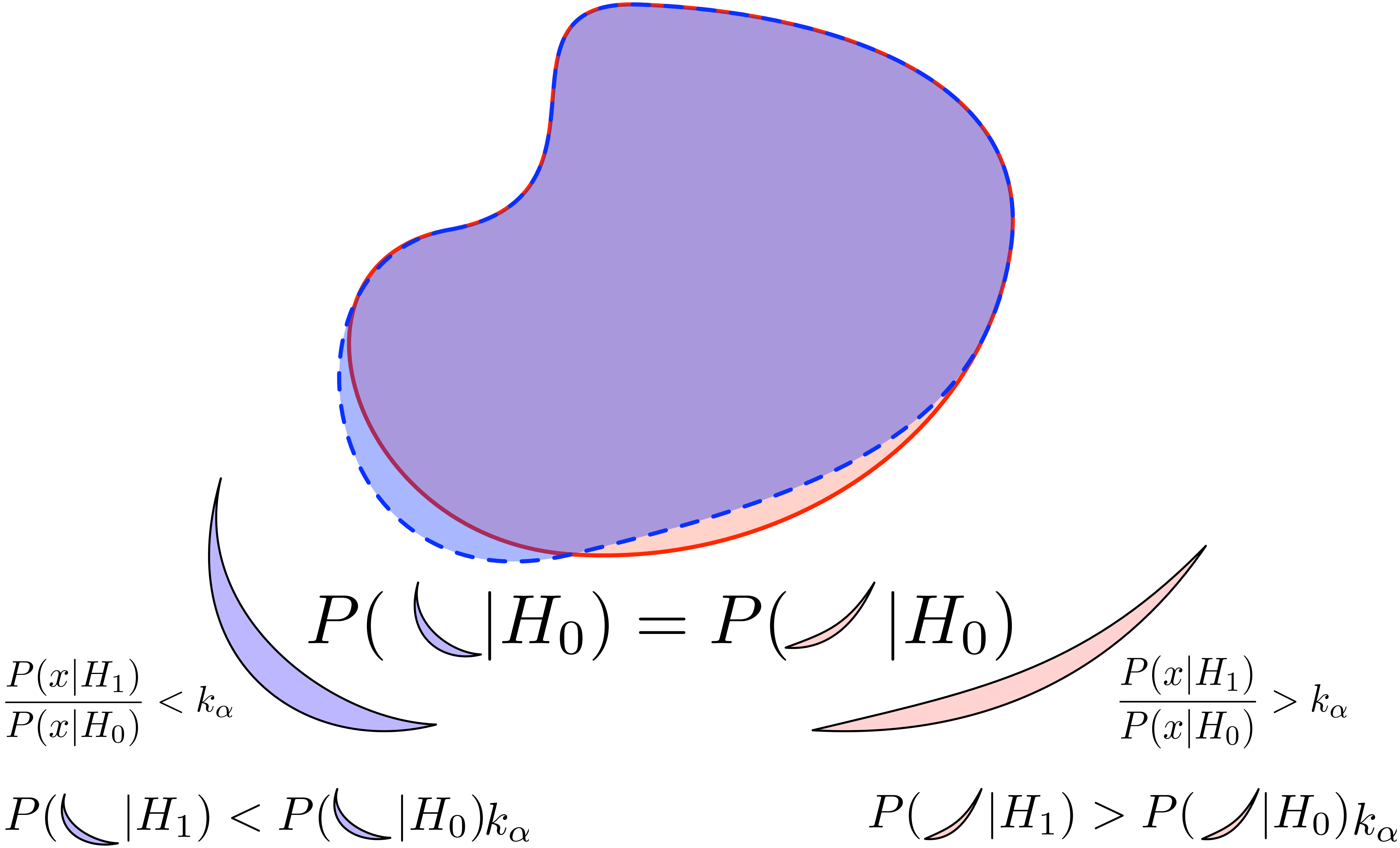
Now consider a variation on the contour that has the same size (eg. same probability under H_0)

A SHORT PROOF OF NEYMAN-PEARSON



Because the new area is outside the contour of the likelihood ratio, we have an inequality

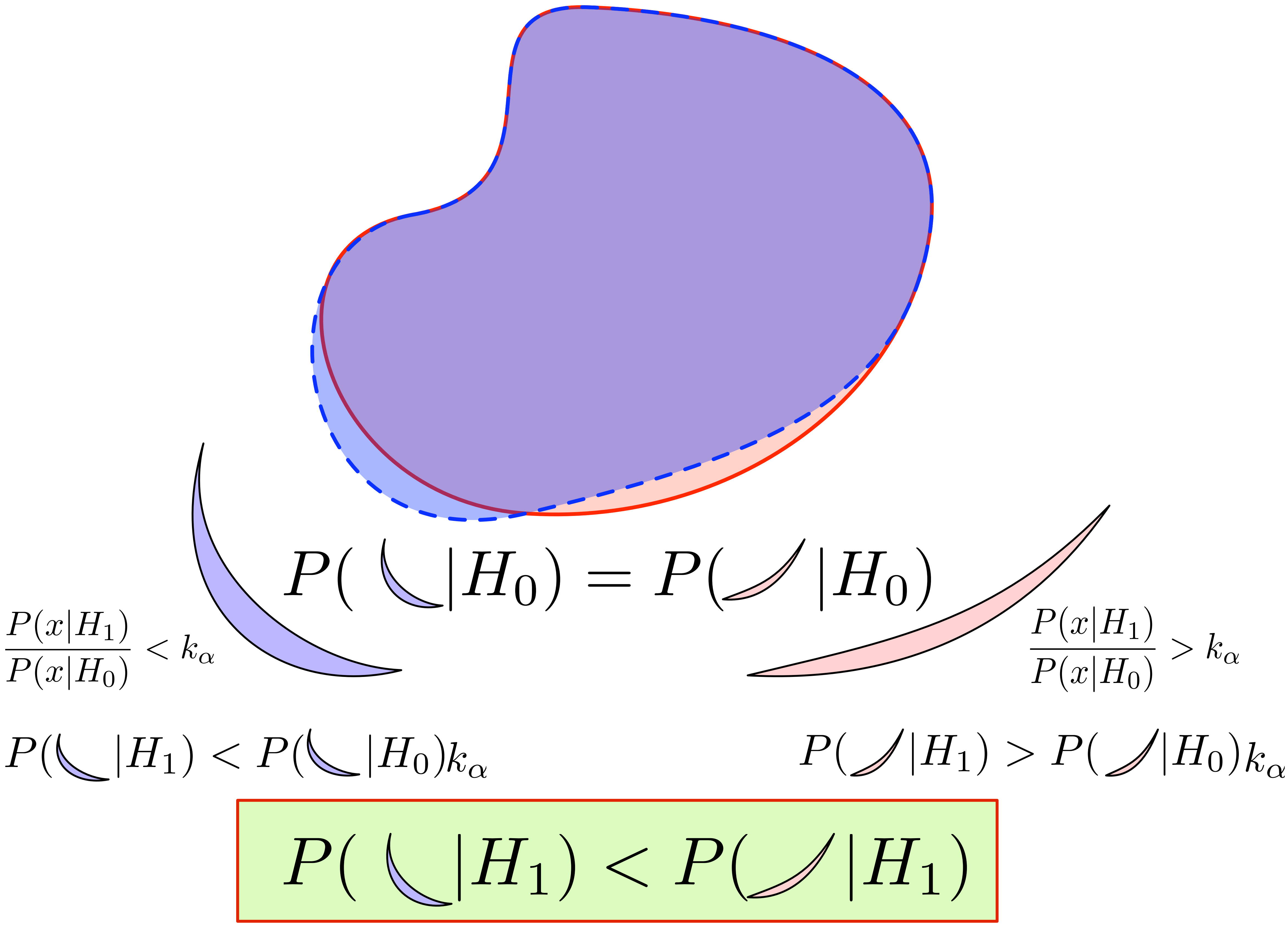
A SHORT PROOF OF NEYMAN-PEARSON



And for the region we lost, we also have an inequality

Together they give...

A short proof of Neyman-Pearson



The new region region has less power.

STATISTICAL DECISION THEORY IN 1 SLIDE

Θ - States of nature; X - possible observations; A - action to be taken

$f(x|\theta)$ - statistical model; $\pi(\theta)$ - prior

δ: X → A - **decision rule** (take some action based on observation)

L: Θ x A → $\mathbb{R}$ - **loss function**, real-valued function true parameter and action

$R(\theta,\delta) = E_{f(x|\theta)}[L(\theta, \delta)]$ - **risk**

- A decision δ^* rule **dominates** a decision rule δ if and only if $R(\theta,\delta^*) \leq R(\theta,\delta)$ for all θ, and the inequality is strict for some θ.

- A decision rule is **admissible** if and only if no other rule dominates it; otherwise it is inadmissible

$r(\pi, \delta) = E_{\pi(\theta)}[\, R(\theta,\delta)]$ - **Bayes risk** (expectation over θ w.r.t. prior and possible observations)

$\rho(\pi, \delta \mid x) = E_{\pi(\theta|x)}[\, L(\theta,\delta(x))]$ - **expected loss** (expectation over θ w.r.t. posterior $\pi(\theta|x)$)

- δ' is a (generalized) Bayes rule if it minimizes the expected loss

- under mild conditions every admissible rule is a (generalized) Bayes rule (**with respect to some prior** —possibly an improper one—that favors distributions where that rule achieves low risk). Thus, in frequentist decision theory it is sufficient to consider only (generalized) Bayes rules.

- Conversely, while Bayes rules with respect to proper priors are virtually always admissible, generalized Bayes rules corresponding to improper priors need not yield admissible procedures. Stein's example is one such famous situation.

Statistical Issues in Searches
Richard Lockhart
General
Optimality
Exclusion
Meta-analysis
Estimating Equations
Multimodal L
Bayes Power
Separate hypotheses
Stuff I won't get to

- Optimality theory: Data X. Model $f(x|\theta), \theta \in \Theta$.
- Decision problem: observe X, make decision $d(X)$.
- Lose $L(d(X), \theta)$ – real valued.
- Judge quality of $d(X)$ by long run average risk:

$$R(d, \theta) = \langle L(d(X), \theta \rangle_\theta = \mathrm{E}\left[L(d(X), \theta | \theta\right].$$

- Key idea: admissibility.
- Procedure d_1 is better than d_2 if, for *all* θ,

$$R(d_1, \theta) < R(d_2, \theta).$$

- We call d_2 *inadmissible*.

Admissibility and Bayes

Theorem

Every admissible procedure is Bayes.

Theorem

Every Bayes procedure is admissible

Written separately because neither is quite right.
But meaning is – sensible procedures need to be Bayes.
Not always an easy restriction to impose – but wise, in my view, to remember.

Famous example – Neyman-Pearson lemma

- Data X with density f_0 or f_1.
- Decision: observe X guess which density. Hypothesis testing.
- Loss: 1 if wrong, 0 if right.
- Risk is

$$(P_0(\text{Reject}), P_1(\text{Accept}))$$

- Neyman Pearson say minimize second component subject to constraint on first.

Implied Priors

- Langrange multipliers. Minimize

$$P_1(\text{Accept}) + \lambda P_0(\text{Reject}) = \beta + \lambda\alpha.$$

- Same as Bayes for prior $P(f_1 \text{ true}) = 1/(1+\lambda)$.
- Then adjust prior (λ) to find Bayes procedure which satisfies constraint.
- Notice that $\lambda/(1+\lambda) = P(H_o)$.
- Procedure implies (at least one) prior.

BAYES THEOREM

Bayes' theorem relates the conditional and marginal probabilities of events A & B

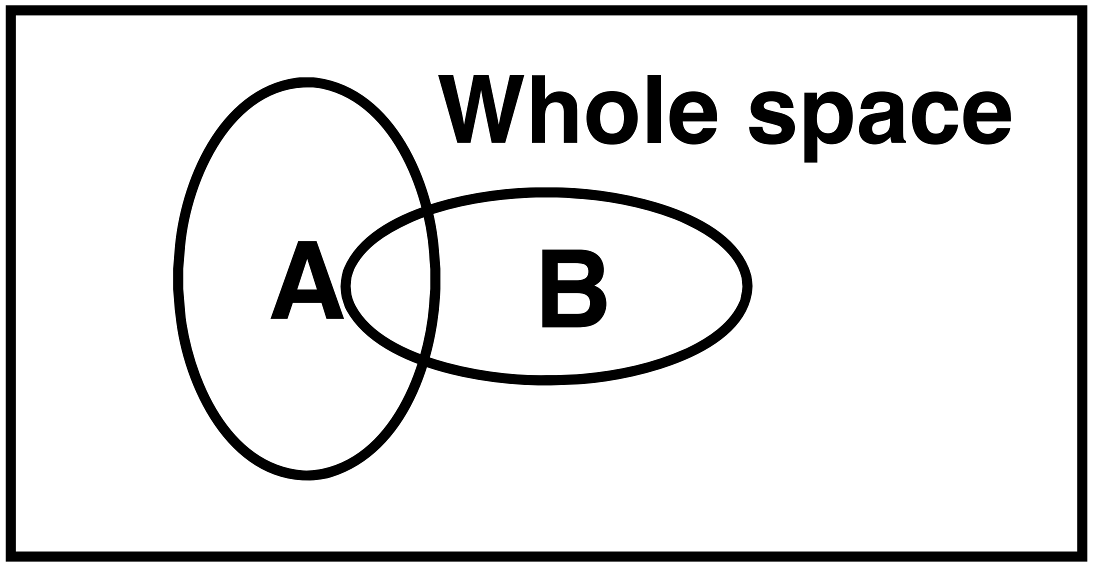
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

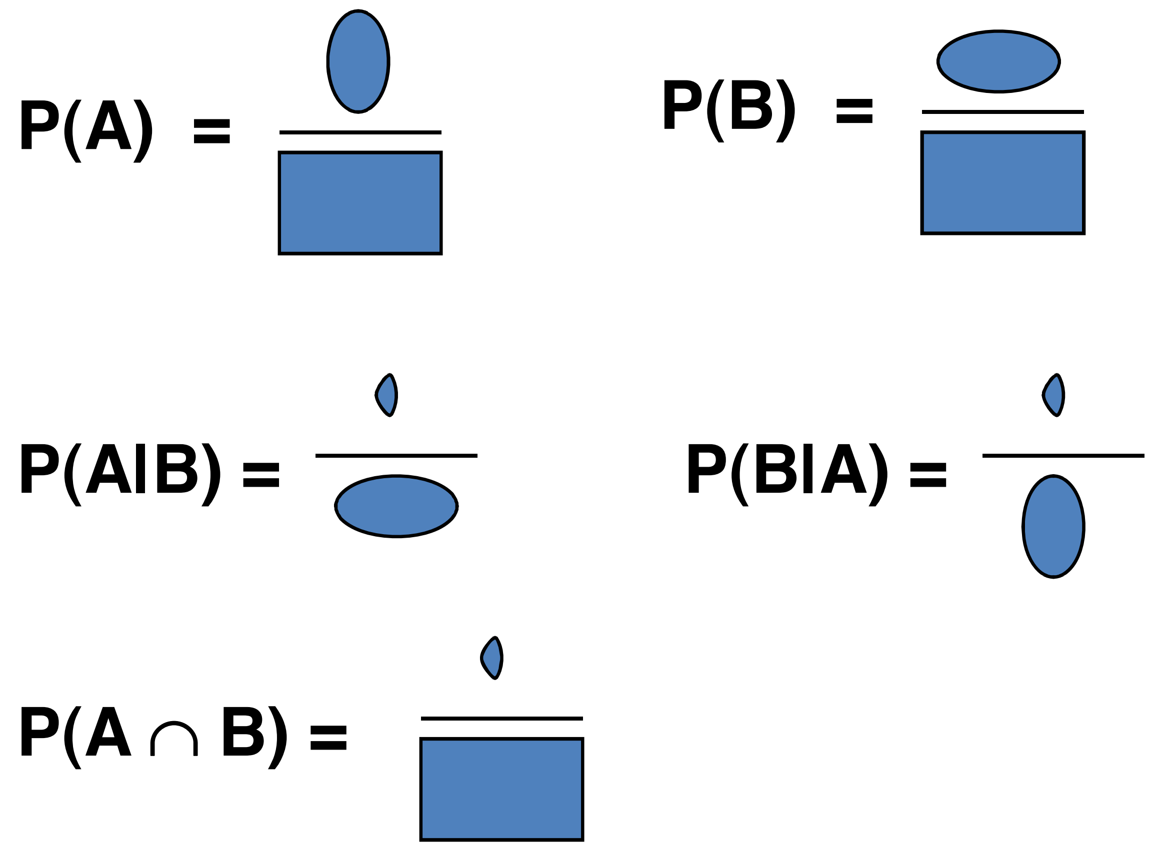
- **P(*A*)** is the prior probability. It is "prior" in the sense that it does not take into account any information about *B*.
- **P(*A*|*B*)** is the conditional probability of *A*, given *B*. It is also called the posterior probability because it is derived from or depends upon the specified value of *B*.
- **P(*B*|*A*)** is the conditional probability of *B* given *A*.
- **P(*B*)** is the prior or marginal probability of *B*, and acts as a normalizing constant.

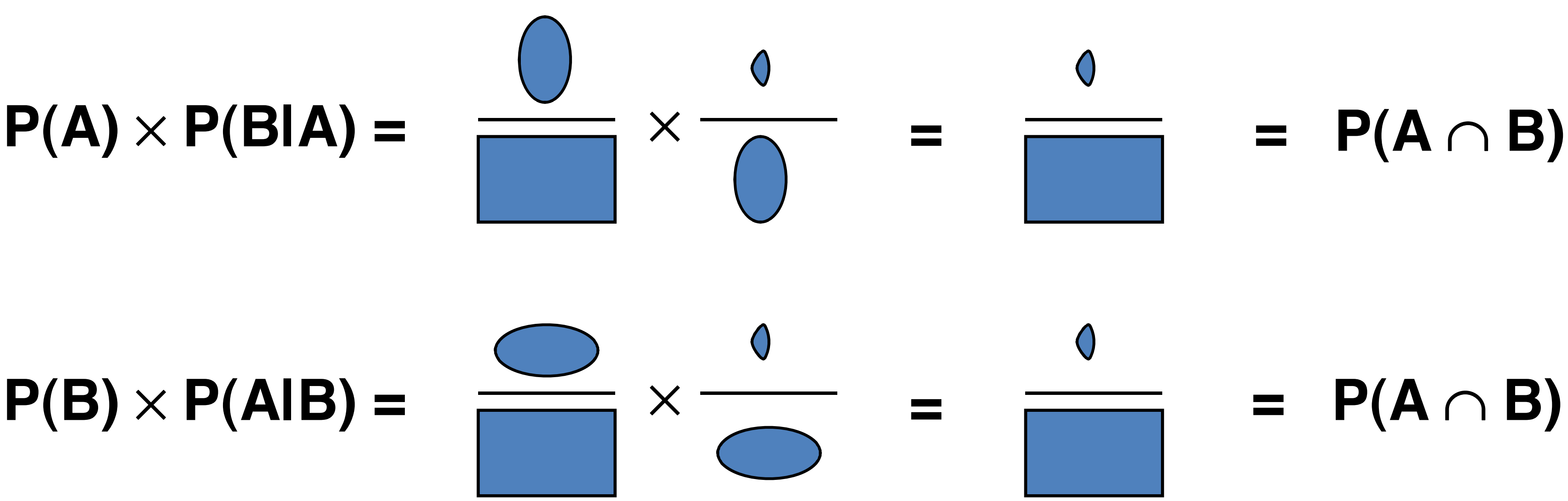
$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\mathcal{N}} \propto L(\theta)\pi(\theta)$$

… IN PICTURES (FROM BOB COUSINS)

P, Conditional P, and Derivation of Bayes' Theorem in Pictures



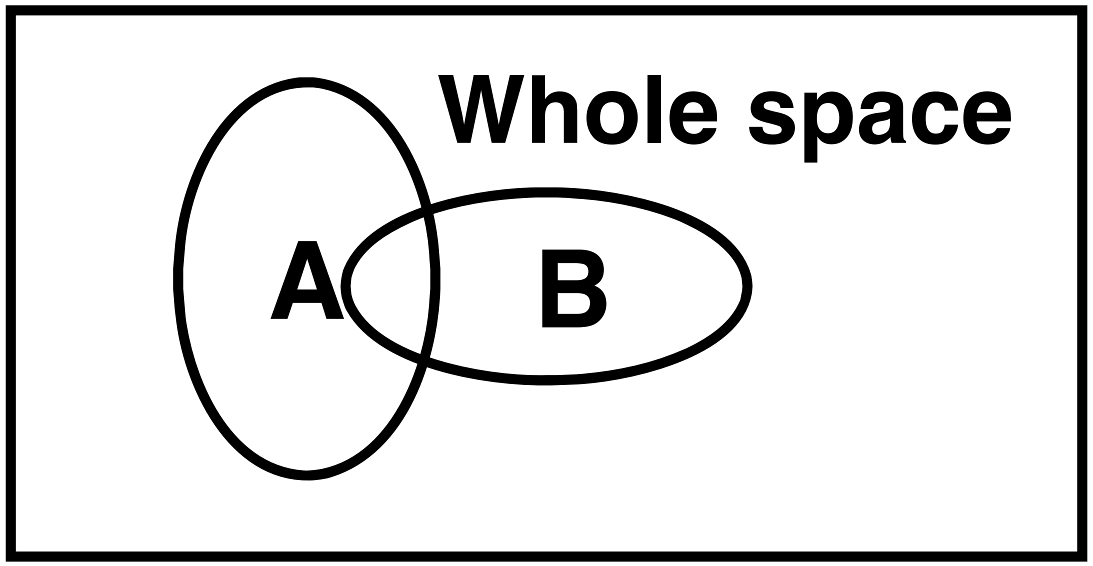


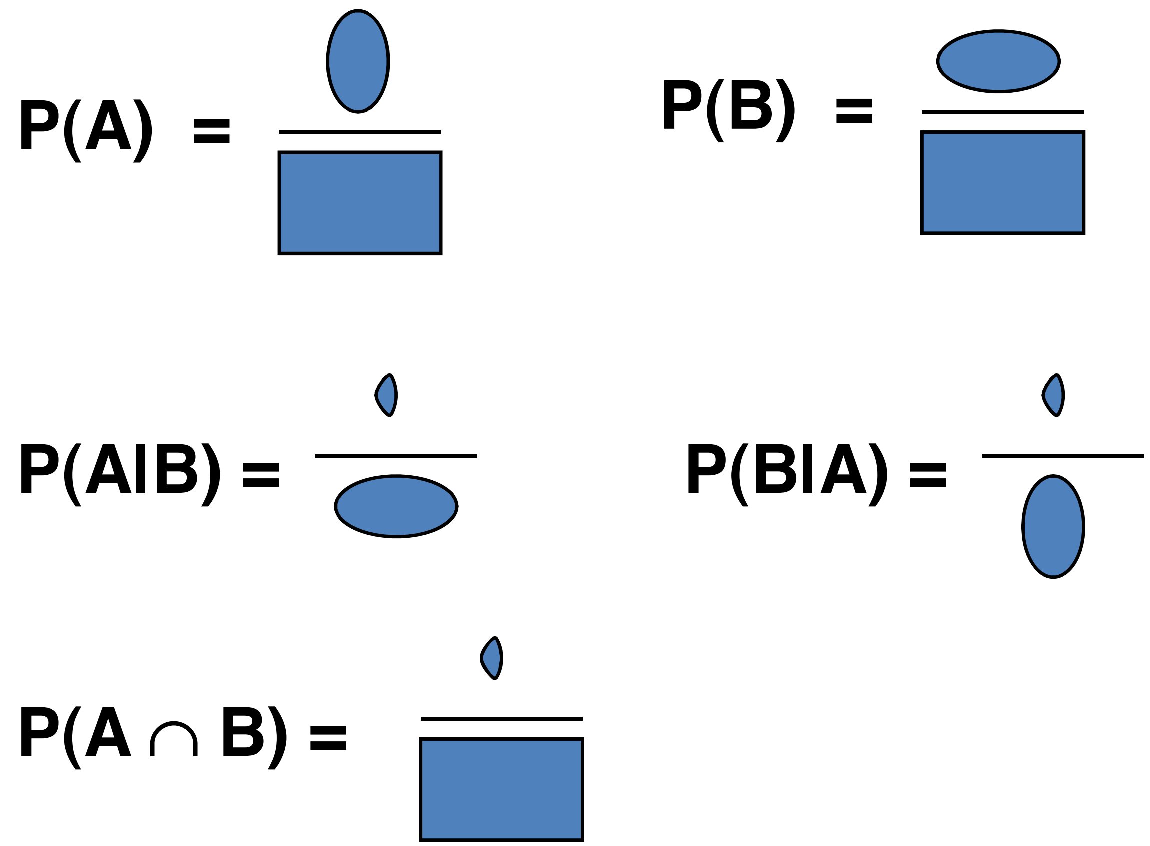


Bob Cousins, CMS, 2008

$\Rightarrow$ **P(B|A) = P(A|B) × P(B) / P(A)**

P, Conditional P, and Derivation of Bayes' Theorem in Pictures





Don't forget about "Whole space" Ω I will drop it from the notation typically, but occasionally it is important.

$\Rightarrow$ **P(B|A) = P(A|B) × P(B) / P(A)**

Bob Cousins, CMS, 2008

Louis's Example

P (Data;Theory) $\neq$ P (Theory;Data)

Theory = male or female

Data = pregnant or not pregnant

P (pregnant ; female) ~ 3%

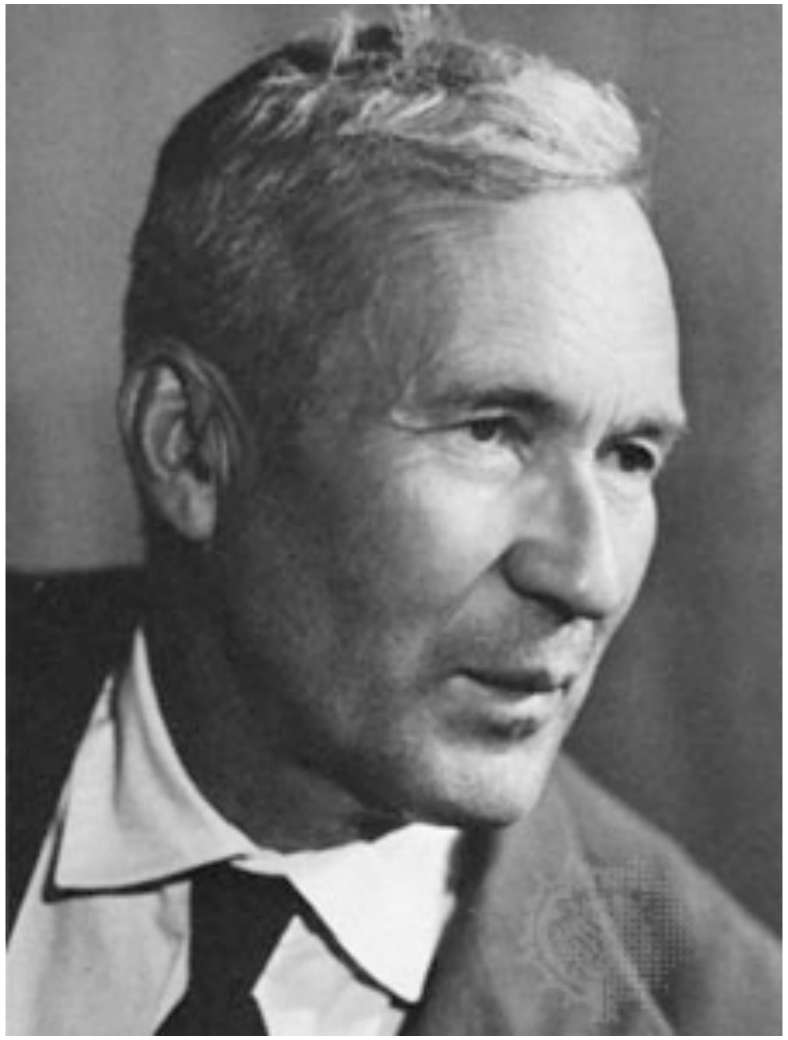
but

P (female ; pregnant) >>>3%

Axioms of Probability

These Axioms are a mathematical starting point for probability and statistics

1. probability for every element, E, is non-negative $P(E) \geq 0 \qquad \forall E \subseteq \mathcal{F} = 2^{\Omega}$
2. probability for the entire space of possibilities is 1 $P(\Omega) = 1.$
3. if elements E_i are disjoint, probability is additive $P(E_1 \cup E_2 \cup \cdots) = \sum_i P(E_i).$



Kolmogorov axioms (1933)

Consequences:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$$P(\Omega \setminus E) = 1 - P(E)$$

Different definitions of Probability

Frequentist

- defined as limit of long term frequency
- probability of rolling a 3 := limit of (# rolls with 3 / # trials)
  - you don't need an infinite sample for definition to be useful
  - sometimes ensemble doesn't exist
    - eg. P(Higgs mass = 125 GeV), P(it will snow tomorrow)
- Intuitive if you are familiar with Monte Carlo methods
- compatible with orthodox interpretation of probability in Quantum Mechanics. Probability to measure spin projected on x-axis if spin of beam is polarized along +z

$$|\langle \rightarrow \mid \uparrow \rangle|^2 = \frac{1}{2}$$

Subjective Bayesian

- Probability is a degree of belief (personal, subjective)
  - can be made quantitative based on betting odds
  - most people's subjective probabilities are not ***coherent*** and do not obey laws of probability

http://plato.stanford.edu/archives/sum2003/entries/probability-interpret/#3.1

Measurement / Estimators

ESTIMATORS

Given some model $f(x|\alpha)$ and a set of observations $\{x_i\}$ often one wants to estimate the true value of α (assuming the model is true).

An **estimator** is function of the data written $\hat{\alpha}(x_1, \ldots x_n)$

- Since the data are random, so is the resulting estimate
- often it is just written $\hat{\alpha}$, where the x-dependence is implicit
- one can compute expectation of the estimator

$$E[\hat{\alpha}(x)|\alpha] = \int \hat{\alpha}(x) f(x|\alpha) dx$$

Properties of estimators:

- **bias** $E[\hat{\alpha}(x)|\alpha] - \alpha$ (unbiased means bias=0)
- **variance** $E[(\hat{\alpha}(x) - \bar{\alpha})^2|\alpha] = \int (\hat{\alpha}(x) - \bar{\alpha})^2 f(x|\alpha) dx$
- **asymptotic bias** limit of bias with infinite observations

Maximum likelihood estimators

There are many different possible estimators, but the most well-known and well-studied is the maximum likelihood estimator (MLE)

$$\hat{\alpha}(x) = \text{argmax}_\alpha L(\alpha) = \text{argmax}_\alpha f(x|\alpha)$$

This is just the value of α that maximizes the likelihood

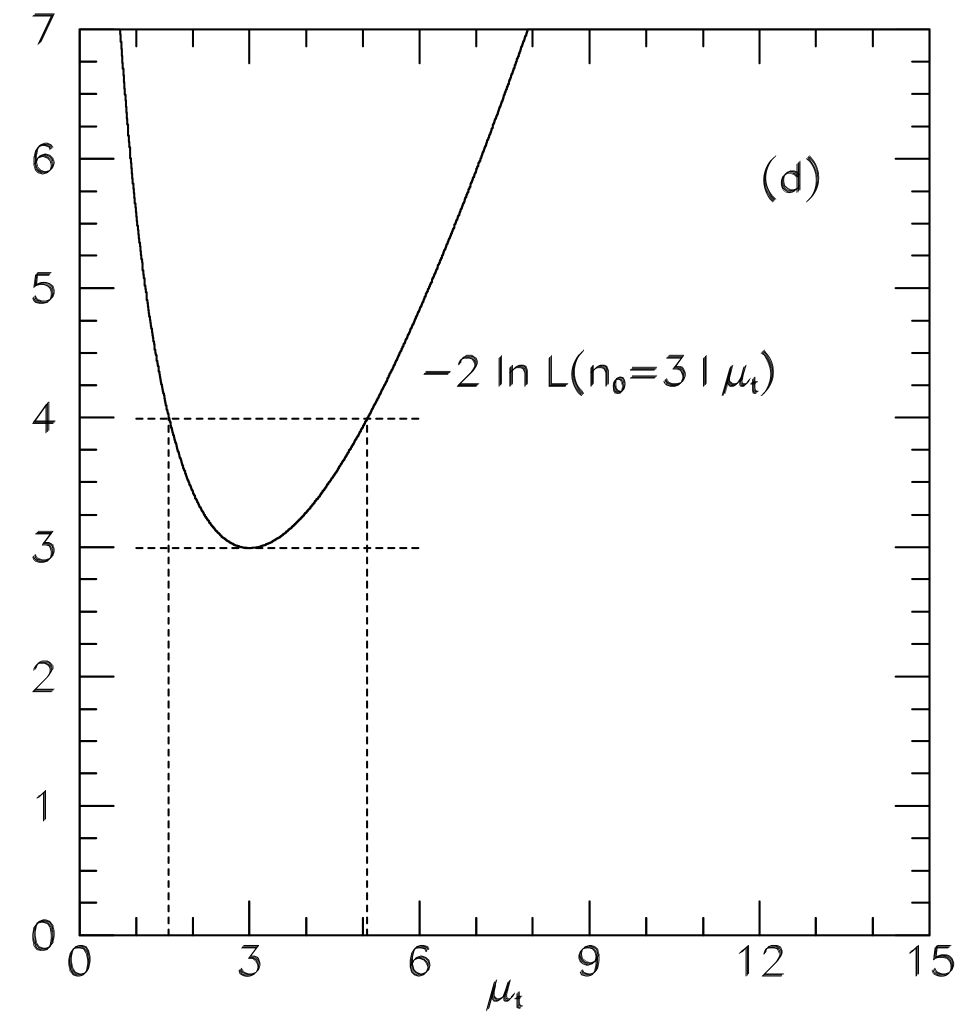


Figure from R. Cousins, Am. J. Phys. 63 398 (1995)

Example: the Poisson distribution

$$Pois(n|\mu) = \mu^n \frac{e^{-\mu}}{n!}$$

Maximizing $L(\mu)$ is the same as minimizing -ln $L(\mu)$

$$-\frac{d}{d\mu} \ln L(\mu)\Big|_{\hat{\mu}} = 0 = \frac{d}{d\mu}\left(\mu - n\ln\mu + \underbrace{\ln n!}_{\text{const}}\right) = 1 - \frac{n}{\mu}$$

$$\Rightarrow \hat{\mu} = n$$

In this case, the MLE is unbiased b/c E[n]=μ

A SECOND EXAMPLE

Consider a set of observations $\{x_i\}$ and we want to estimate the mean of a Gaussian with known σ

which gives

$$G(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$-\frac{d}{d\mu}\ln L(\mu)\Big|_{\hat{\mu}} = 0 = \frac{d}{d\mu}\left(\sum_i \frac{(x_i-\mu)^2}{2\sigma^2} + \underbrace{\ln\sqrt{2\pi}\sigma}_{\text{const}}\right) = \sum_i \frac{(x_i-\mu)}{\sigma^2}$$

$$\Rightarrow \hat{\mu} = \frac{1}{N}\sum_i x_i$$ (an unbiased estimator) .

However, the MLE $\hat{\sigma}^2 = \frac{1}{N}\sum_i (x_i-\mu)^2$ is biased

It can be shown that $\hat{\sigma}^2 = \frac{1}{N-1}\sum_i (x_i-\mu)^2$ is unbiased

Thus, the MLE is **asymptotially unbiased .**

Note: if $\hat{\sigma}^2$ is an unbiased estimate of σ^2, then $\sqrt{\{\hat{\sigma}^2\}}$ is a biased estimate of σ.

Covariance and Correlation

Define covariance cov[x,y] (also use matrix notation V_{xy}) as

$$\text{cov}[x,y] = E[xy] - \mu_x \mu_y = E[(x-\mu_x)(y-\mu_y)]$$

Correlation coefficient (dimensionless) defined as

$$\rho_{xy} = \frac{\text{cov}[x,y]}{\sigma_x \sigma_y}$$

If x, y, independent, i.e., $f(x,y) = f_x(x) f_y(y)$, then

$$E[xy] = \int\int xy\, f(x,y)\, dx dy = \mu_x \mu_y$$

→ $\text{cov}[x,y] = 0$ x and y, 'uncorrelated'

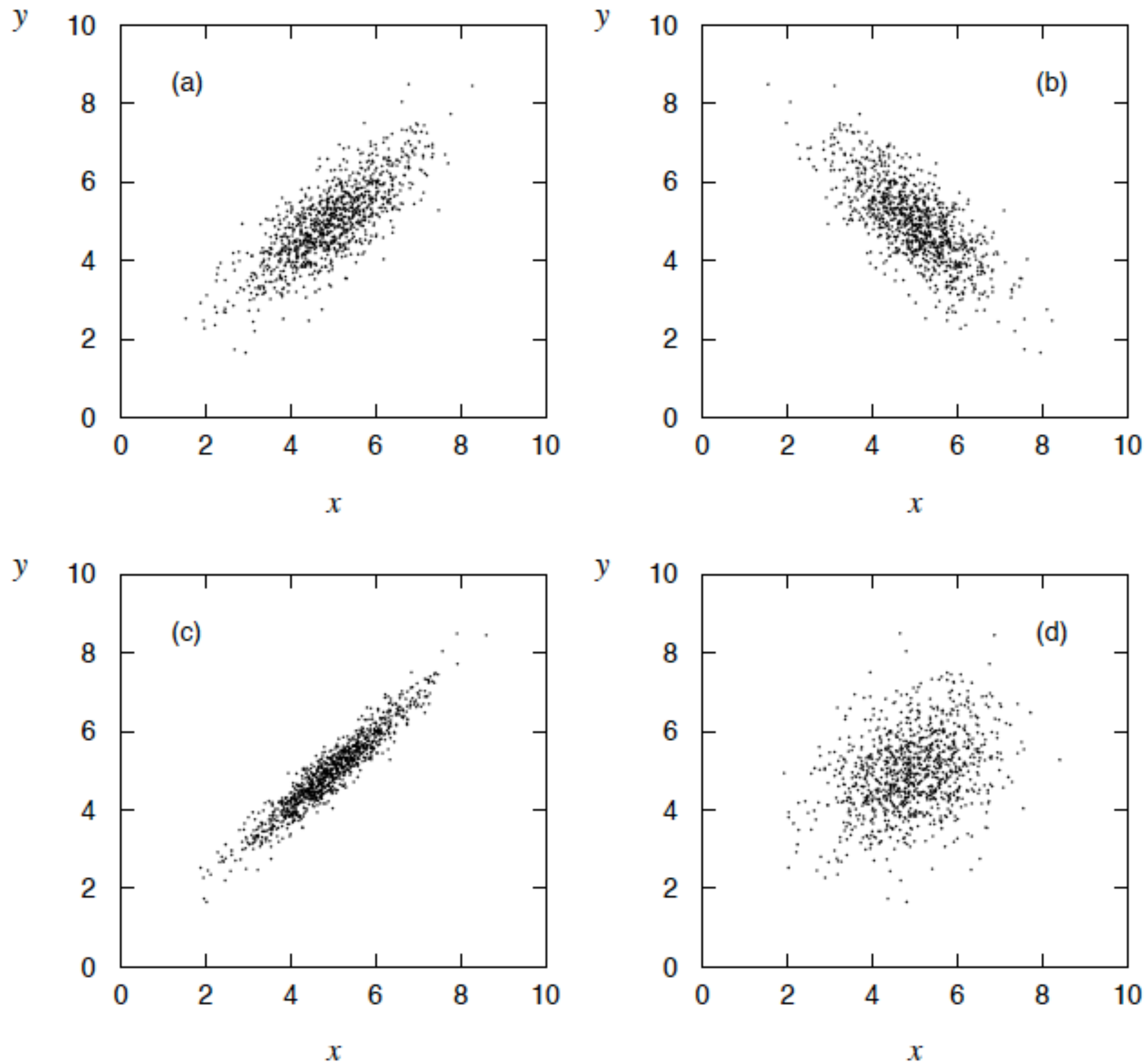
N.B. converse not always true.

[G. Cowan]

CORRELATION (CONT.)

$\rho = 0.75$

$\rho = -0.75$

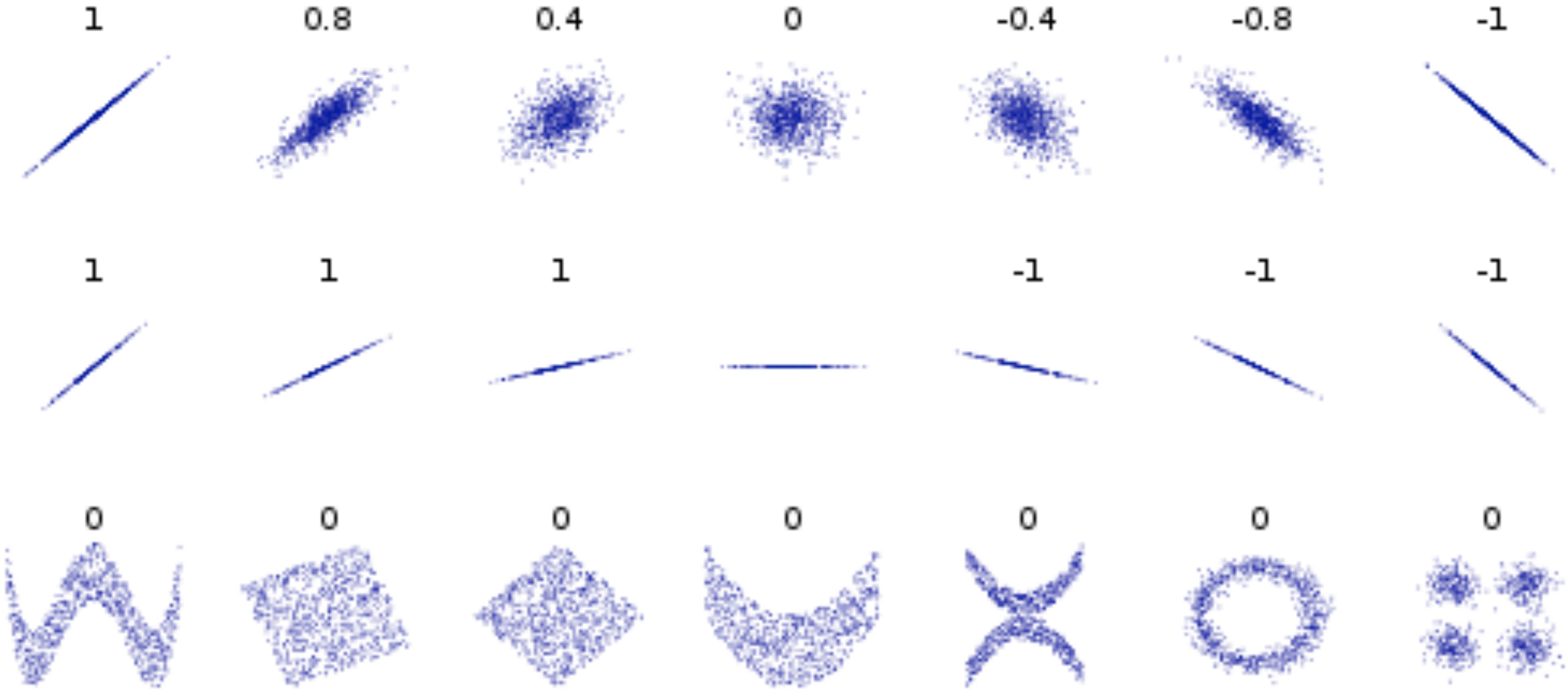


$\rho = 0.95$

$\rho = 0.25$

[G. Cowan]

CORRELATION (CONT.)



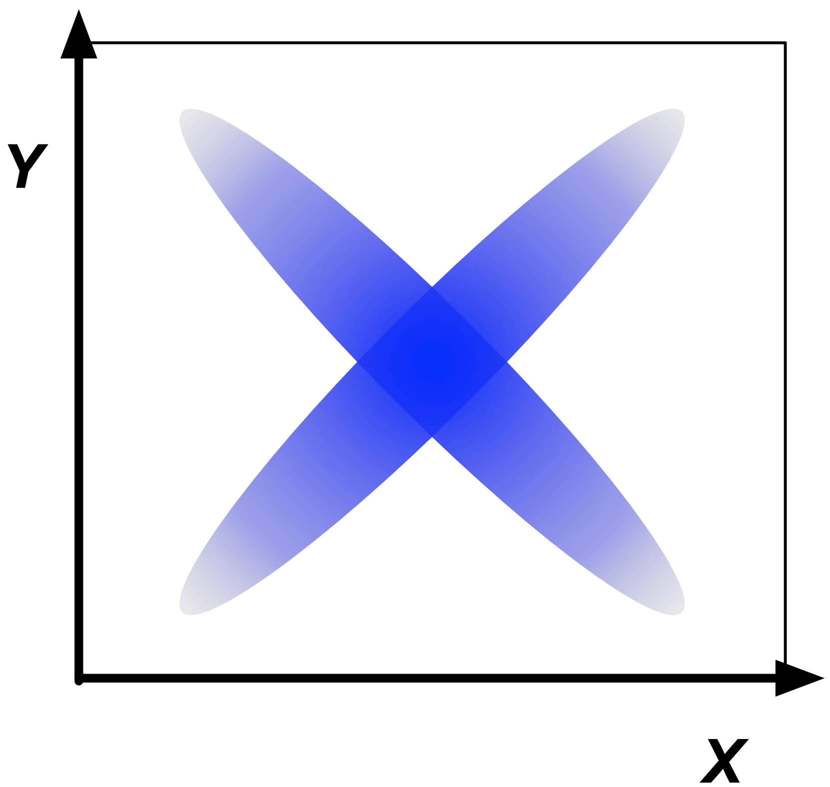
http://en.wikipedia.org/wiki/Correlation_and_dependence

Mutual Information is a more general notion of ‘correlation’

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p_1(x)\,p_2(y)}\right),$$

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

- it is symmetric: I(X;Y) = I(Y;X)
- if and only if X,Y totally independent: I(X;Y)=0
- possible for X,Y to be uncorrelated, but not independent



Mutual Information doesn’t seem to be used much within HEP, but it seems quite useful

We introduced Bias and Variance of estimators

$$\mathrm{Var}[\hat{\mu}|\mu] = E[(\hat{\mu} - E[\mu|\mu])^2] \,|\mu]$$

Most physicist are allergic to the idea of a biased estimator

- try to find unbiased estimator with smallest variance
- hence importance of Cramér-Rao bound

But what if we just want to minimize the mean-squared error?

$$MSE[\hat{\mu}|\mu] = E[(\hat{\mu} - \mu)^2] \,|\mu]$$

it decomposes like this

$$MSE[\hat{\mu}|\mu] = \mathrm{Var}[\hat{\mu}|\mu] + (\mathrm{Bias}[\hat{\mu}|\mu])^2$$

So it encodes some relative weight to bias and variance. Think harder!

Cramér-Rao Bound

The minimum variance bound on an estimator is given by the Cramér-Rao inequality:

- simple univariate case:

$$\mathrm{Var}[\hat{\theta}|\theta] = E[(\hat{\theta} - E[\theta|\theta])^2]\,|\theta]$$

- For an unbiased estimator the Cramér-Rao bound states

$$\mathrm{Var}[\hat{\theta}|\theta] \geq \frac{1}{I(\theta)}$$

- where $I(\theta)$ is the Fisher information

$$(\mathcal{I}(\theta))_{i,j} = \mathrm{E}\left[\frac{\partial}{\partial\theta_i} \ln f(X;\theta) \frac{\partial}{\partial\theta_j} \ln f(X;\theta) \middle| \theta \right].$$

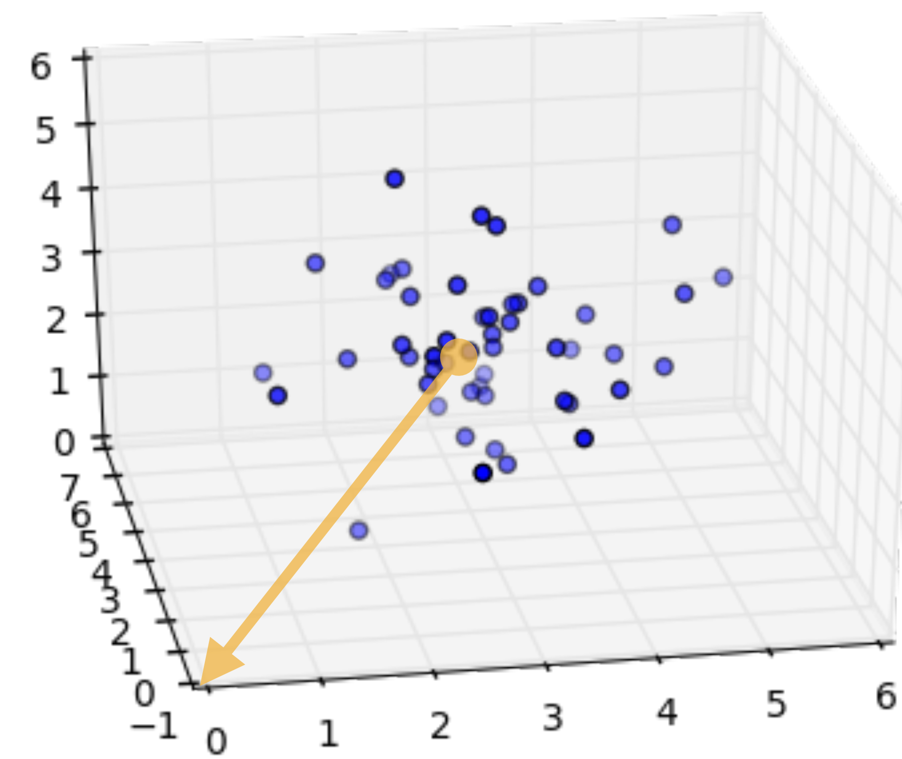
- General form for multiple parameters:

$$\mathrm{cov}[\hat{\theta}|\theta]_{ij} \geq I^{-1}_{ij}(\theta)$$

Maximum Likelihood Estimators *asymptotically* reach this bound

Consider a standard multivariate Gaussian distribution for $\vec{x}$ in n dimensions centered around $\vec{\mu}$

$$f(\vec{x}|\vec{\mu}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_i)^2}{2}\right) .$$



Goal: minimize mean-squared error

$$MSE[\hat{\vec{\mu}}] = E[||\hat{\vec{\mu}} - \vec{\mu}||^2])$$

MLE (unbiased)

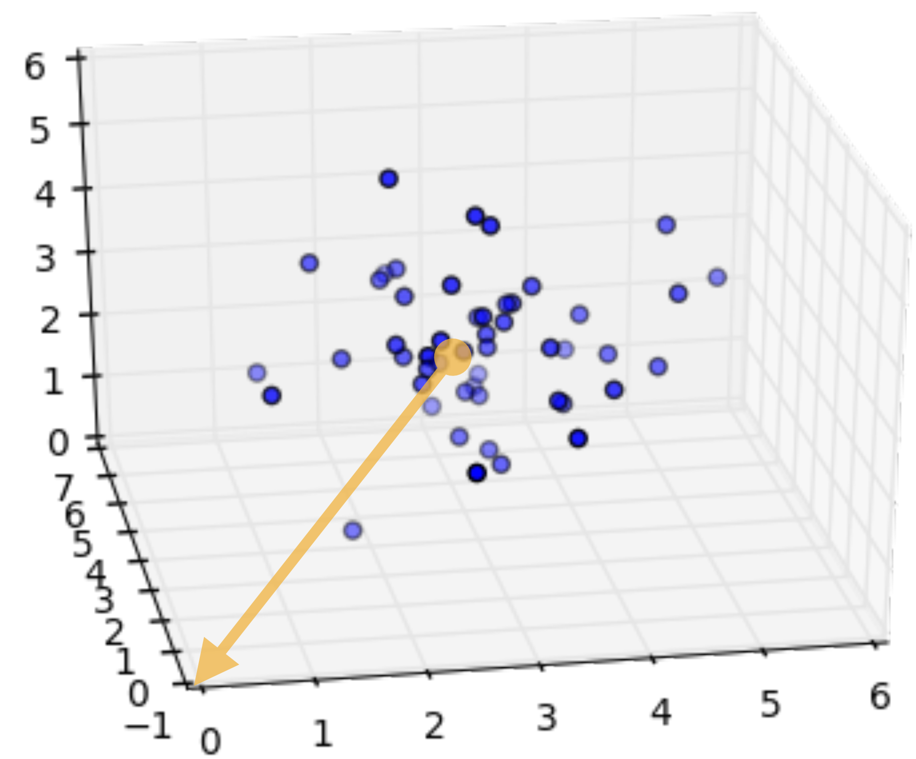
$$\hat{\vec{\mu}}_{MLE} = \overline{x} = \frac{1}{m}\sum_{j=1}^{m} \vec{x}_j$$

James-Stein (weird)

$$\hat{\mu}_{JS} = \left(1 - \frac{n-2}{||\bar{x}||^2}\right)\bar{x}$$

The James-Stein estimator seems like a horrible suggestion

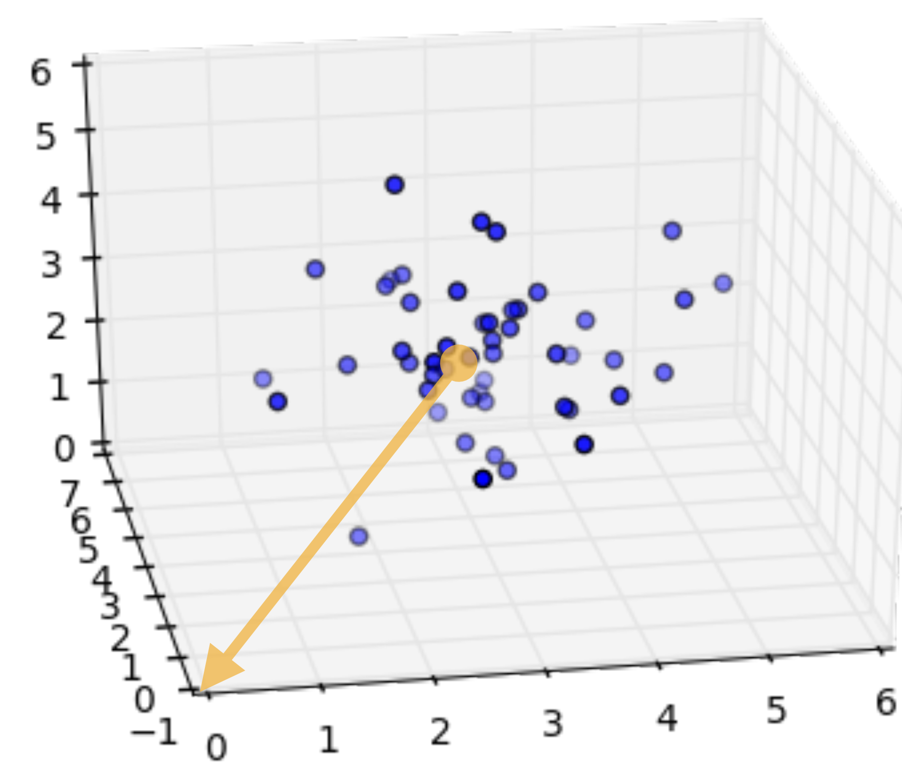
$$\hat{\mu}_{JS} = \left(1 - \frac{n-2}{||\bar{x}||^2}\right)\bar{x}$$



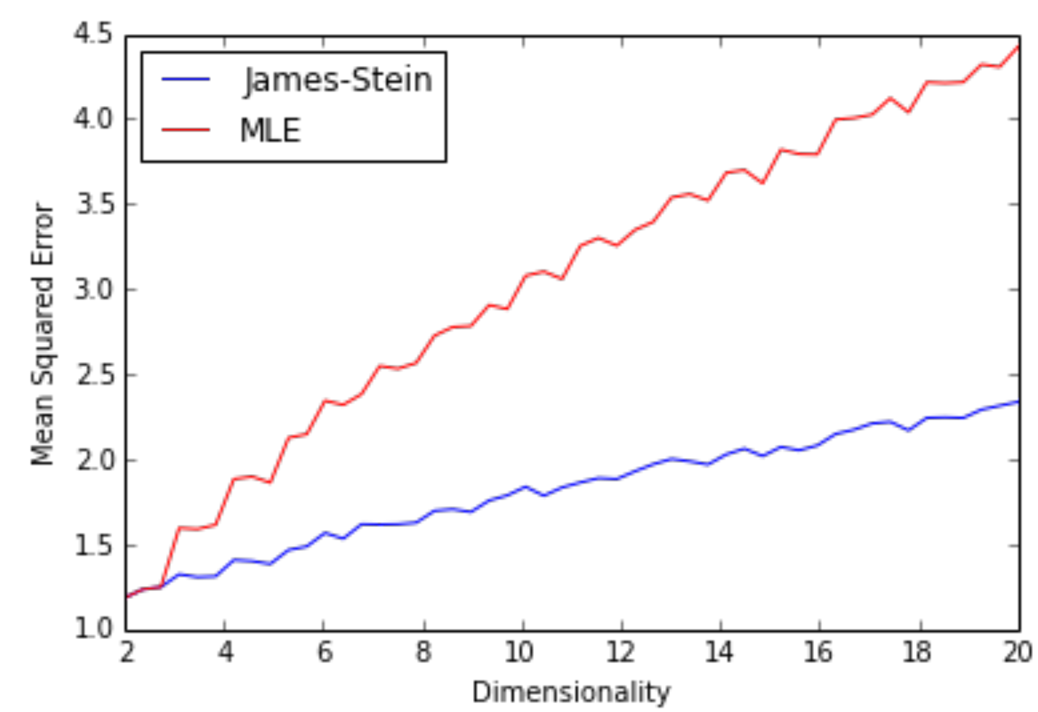
- clearly biased (MLE is not)
- shifts towards origin is not translationally invariant
  x → x′ = x+Δ

The James-Stein estimator seems like a horrible suggestion

$$\hat{\mu}_{JS} = \left(1 - \frac{n-2}{||\bar{x}||^2}\right)\bar{x}$$



- clearly biased (MLE is not)
- shifts towards origin is not translationally invariant
  x ➞ x' = x+Δ



Yet, it has smaller mean squared error than MLE for n>2 !

- it "dominates" the MLE

We introduced Bias and Variance of estimators

$$\text{Var}[\hat{\mu}|\mu] = E[(\hat{\mu} - E[\mu|\mu])^2]\,|\mu]$$

Most physicist are allergic to the idea of a biased estimator

- try to find unbiased estimator with smallest variance
- hence importance of Cramér-Rao bound

But what if we just want to minimize the mean-squared error?

$$MSE[\hat{\mu}|\mu] = E[(\hat{\mu} - \mu)^2]\,|\mu]$$

it decomposes like this

$$MSE[\hat{\mu}|\mu] = \text{Var}[\hat{\mu}|\mu] + (\text{Bias}[\hat{\mu}|\mu])^2$$

So it encodes some relative weight to bias and variance. Think harder!

STATISTICAL DECISION THEORY IN 1 SLIDE

Θ - States of nature; X - possible observations; A - action to be taken

$f(x|\theta)$ - statistical model; $\pi(\theta)$ - prior

$\delta: X \to A$ - **decision rule** (take some action based on observation)

$L: \Theta \times A \to \mathbb{R}$ - **loss function**, real-valued function true parameter and action

$R(\theta,\delta) = E_{f(x|\theta)}[L(\theta, \delta)]$ - **risk**

- A decision δ^* rule **dominates** a decision rule δ if and only if $R(\theta,\delta^*) \leq R(\theta,\delta)$ for all θ, and the inequality is strict for some θ.
- A decision rule is **admissible** if and only if no other rule dominates it; otherwise it is inadmissible

$r(\pi, \delta) = E_{\pi(\theta)}[R(\theta,\delta)]$ - **Bayes risk** (expectation over θ w.r.t. prior and possible observations)

$\rho(\pi, \delta \mid x) = E_{\pi(\theta|x)}[L(\theta,\delta(x))]$ - **expected loss** (expectation over θ w.r.t. posterior $\pi(\theta|x)$)

- δ' is a (generalized) Bayes rule if it minimizes the expected loss
- under mild conditions every admissible rule is a (generalized) Bayes rule (**with respect to some prior** —possibly an improper one—that favors distributions where that rule achieves low risk). Thus, in frequentist decision theory it is sufficient to consider only (generalized) Bayes rules.
- Conversely, while Bayes rules with respect to proper priors are virtually always admissible, generalized Bayes rules corresponding to improper priors need not yield admissible procedures. Stein's example is one such famous situation.

LECTURE 2

Hypothesis Testing ↔ Classification

- Neyman-Pearson, Likelihood Ratio
- "Bayes Optimal" Machine Learning Classifiers & Loss

Extending to include systematics:

- statistical modeling with nuisance parameters
  - RooFit ↔ TensorFlow, automatic differentiation
- Profile Likelihood Ratio & concept of a "pivot"

Parametrized learning

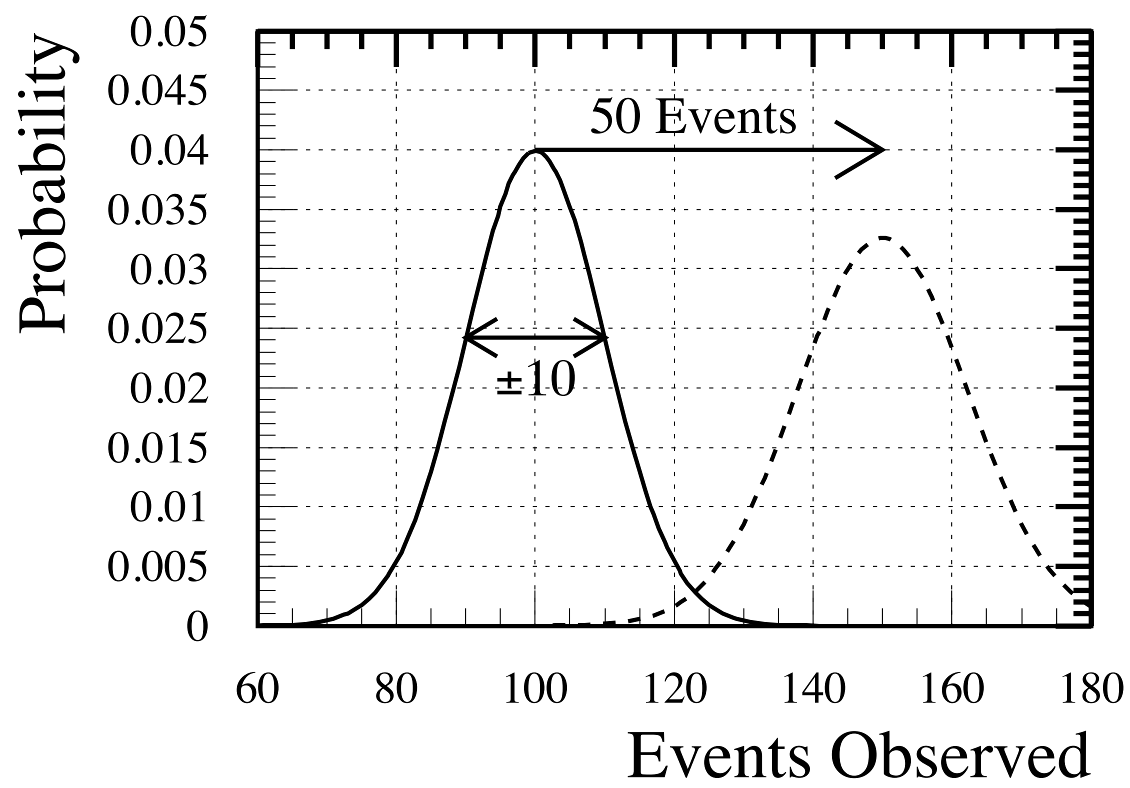
- for classification
- high dimensional reweighting

HYPOTHESIS TESTING

HYPOTHESIS TESTING

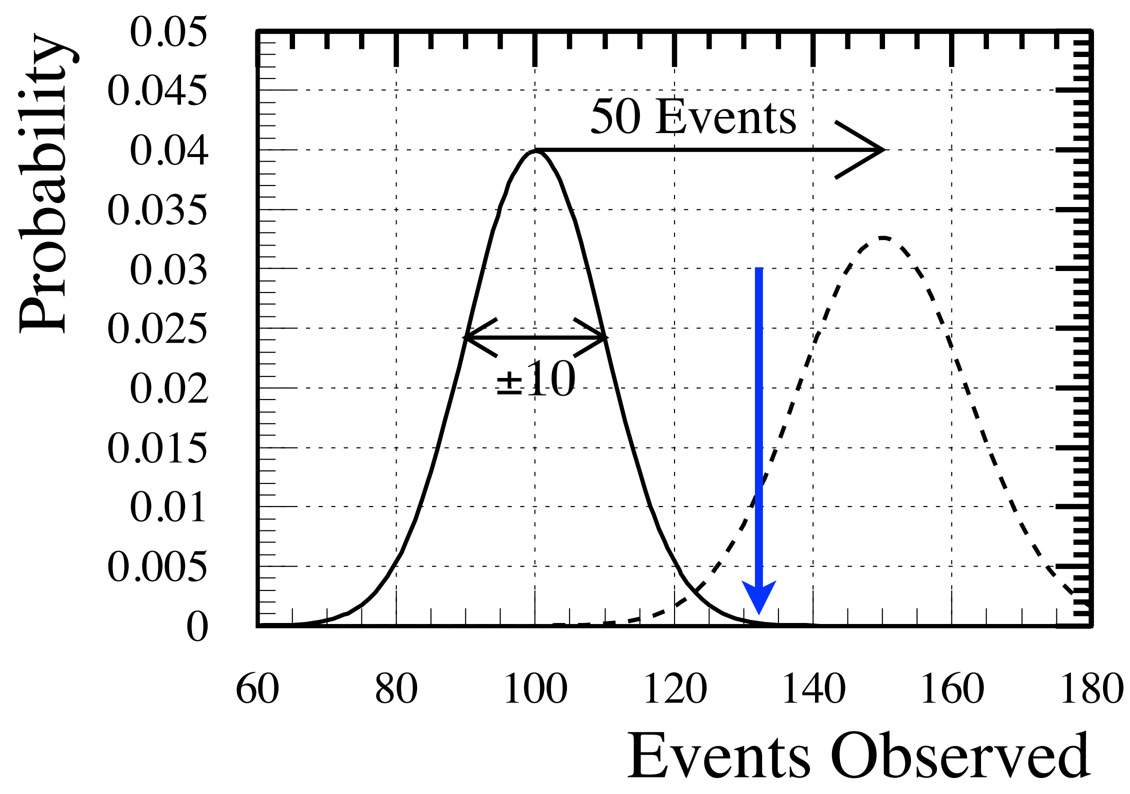
One of the most common uses of statistics in particle physics is Hypothesis Testing (e.g. for discovery of a new particle)

- assume one has pdf for data under two hypotheses:
  - Null-Hypothesis, H_0: eg. background-only
  - Alternate-Hypothesis H_1: eg. signal-plus-background
- one makes a measurement and then needs to decide whether to **reject** or **accept** H_0



One of the most common uses of statistics in particle physics is Hypothesis Testing (e.g. for discovery of a new particle)

- assume one has pdf for data under two hypotheses:
  - Null-Hypothesis, H_0: eg. background-only
  - Alternate-Hypothesis H_1: eg. signal-plus-background
- one makes a measurement and then needs to decide whether to **reject** or **accept** H_0



Before we can make much progress with statistics, we need to decide what it is that we want to do.

- first let us define a few terms:
  - Rate of Type I error α
  - Rate of Type II β
  - Power = $1-\beta$

		Actual condition	
		Guilty	Not guilty
Decision	Verdict of 'guilty'	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

Before we can make much progress with statistics, we need to decide what it is that we want to do.

- first let us define a few terms:
  - Rate of Type I error α
  - Rate of Type II β
  - Power = $1 - \beta$

		Actual condition	
		Guilty	Not guilty
Decision	Verdict of 'guilty'	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

Treat the two hypotheses asymmetrically

- the Null is special.
  - Fix rate of Type I error, call it “the size of the test”

Before we can make much progress with statistics, we need to decide what it is that we want to do.

- first let us define a few terms:
  - Rate of Type I error α
  - Rate of Type II β
  - Power = $1 - \beta$

		Actual condition	
		Guilty	Not guilty
Decision	Verdict of 'guilty'	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

Treat the two hypotheses asymmetrically

- the Null is special.
  - Fix rate of Type I error, call it "the size of the test"

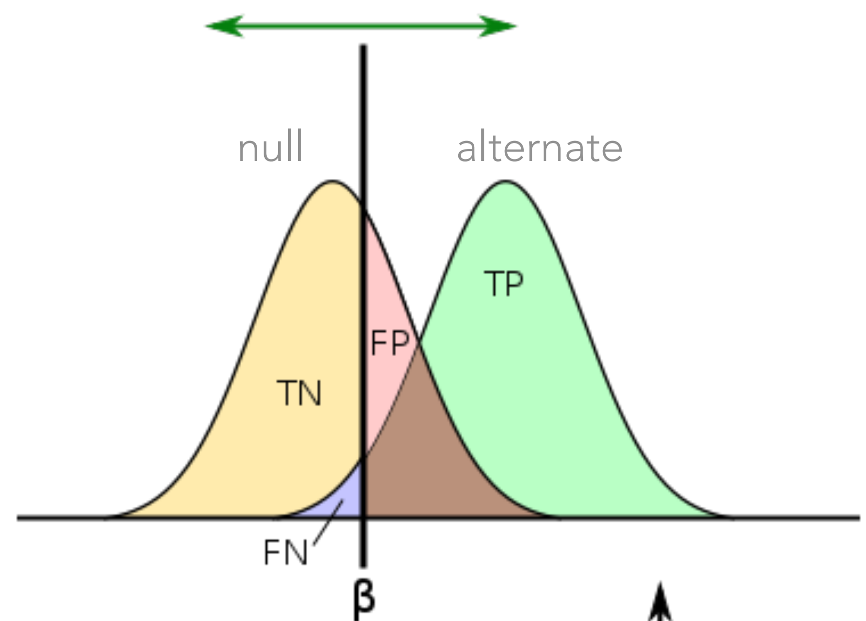
Now one can state "a well-defined goal"

- Maximize power for a fixed rate of Type I error

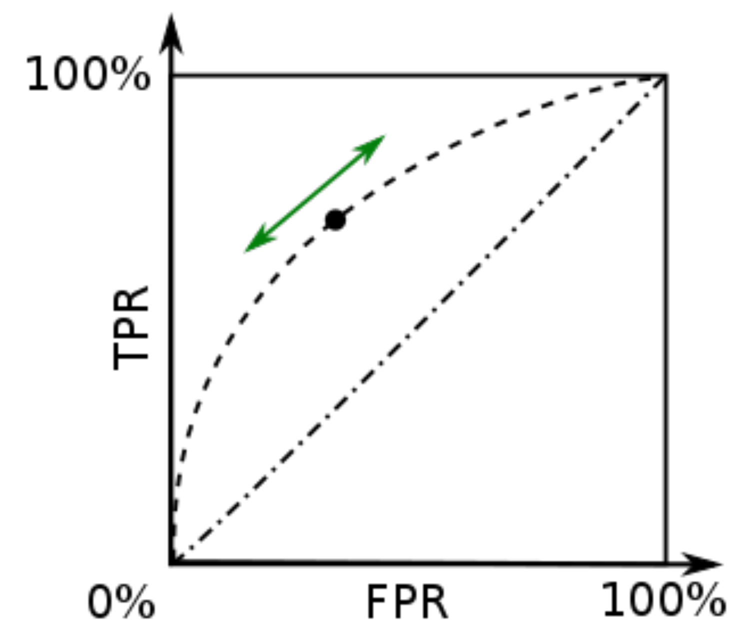
HYPOTHESIS TESTING

Classical hypothesis testing typically framed in terms of true/false : positive/negative

		Actual condition	
		Guilty	**Not guilty**
Decision	**Verdict of 'guilty'**	True Positive **power**	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative



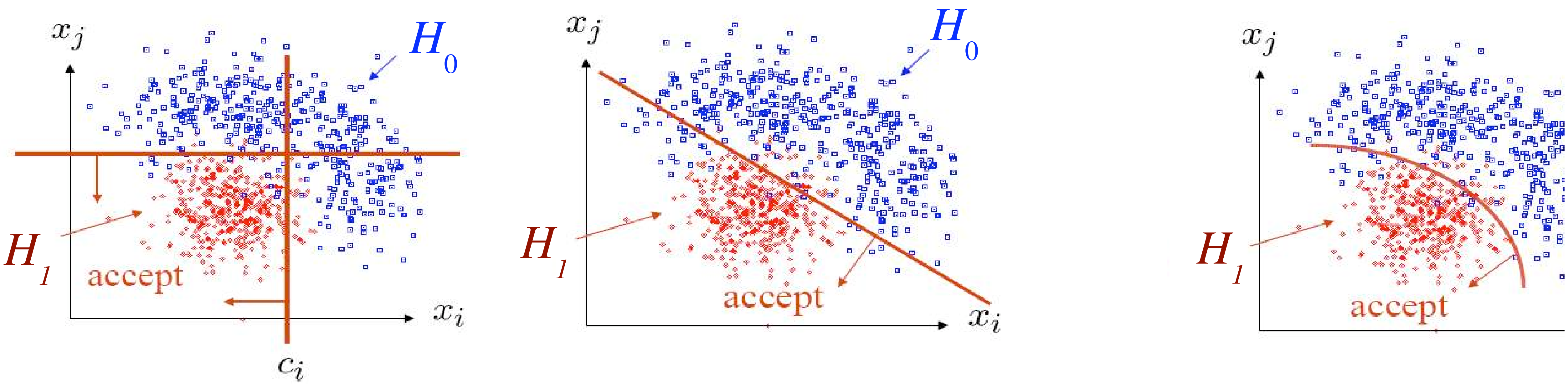
TP	FP
FN	TN



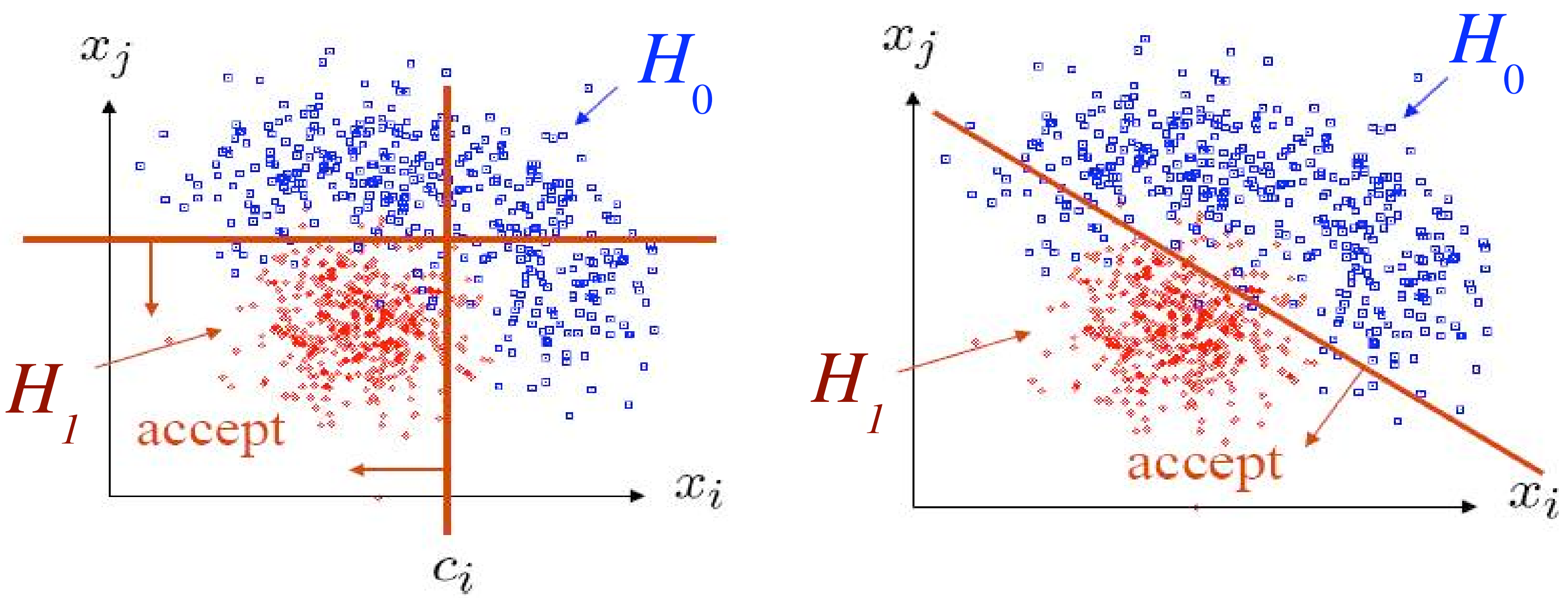
actually guilty ↔ new physics

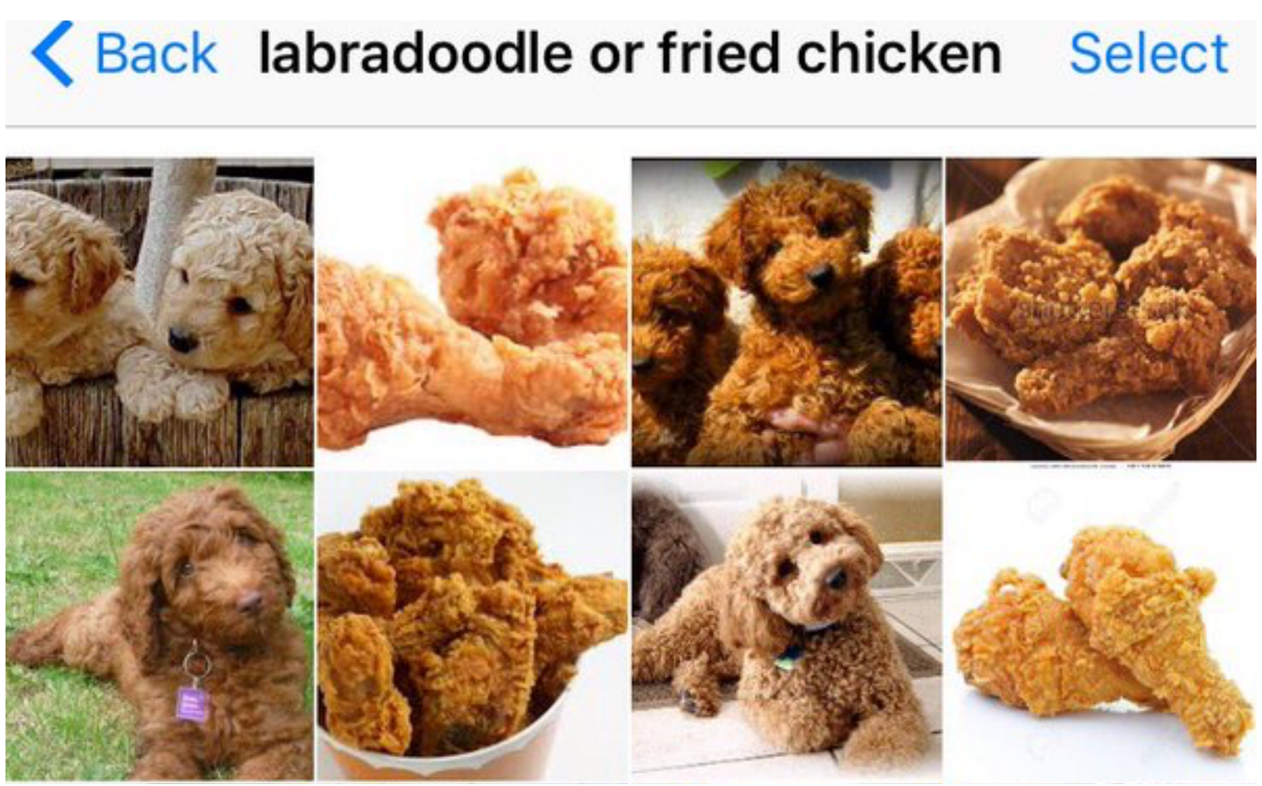
verdict guilty ↔ claim discovery

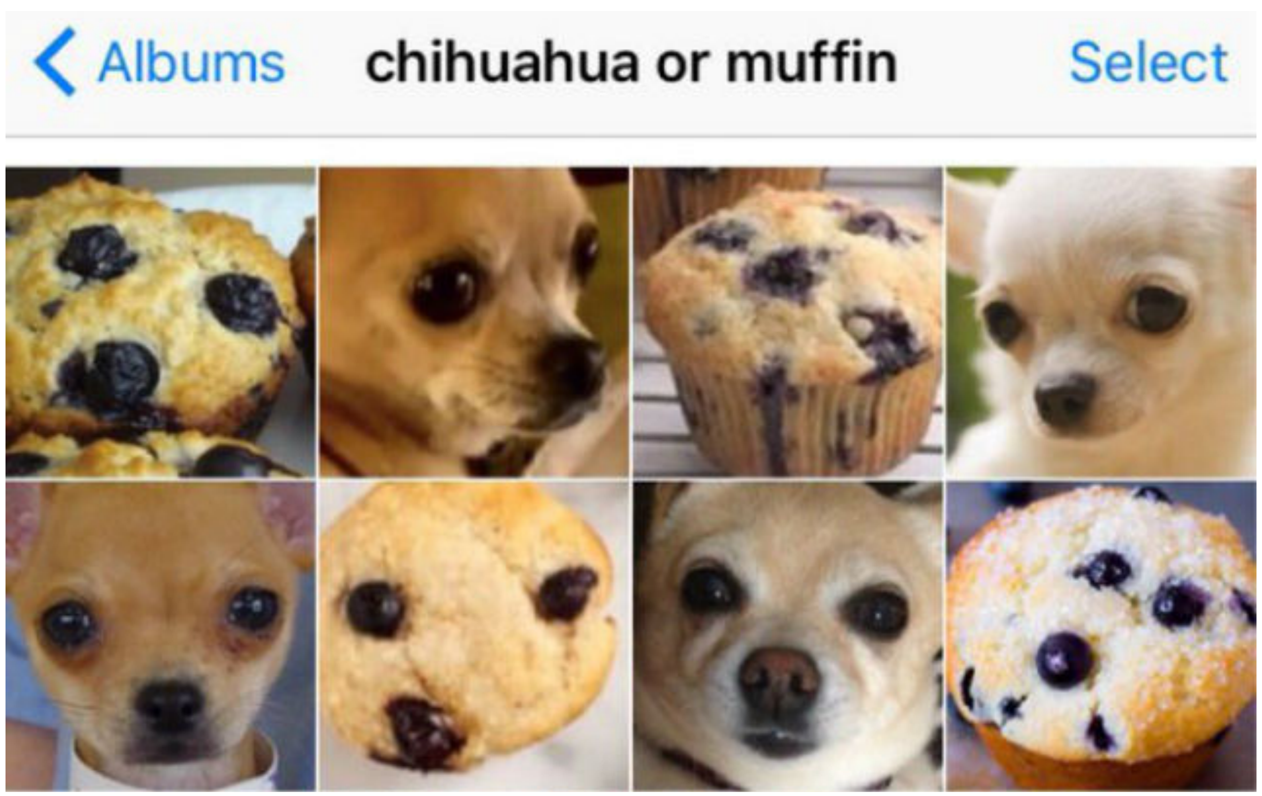
If the data are high-dimensional, it's not obvious how to draw the boundary between accept/reject the null hypothesis



If the data are high-dimensional, it's not obvious how to draw the boundary between accept/reject the null hypothesis







THE NEYMAN-PEARSON LEMMA

In 1928-1938 Neyman & Pearson developed a theory in which one must consider competing Hypotheses:

- the Null Hypothesis H_0 (background only)
- the Alternate Hypothesis H_1 (signal-plus-background)

Given some probability that we wrongly reject the Null Hypothesis

$$\alpha = P(x \notin W | H_0)$$

(Convention: if data falls in W then we accept H_0)

Find the region W such that we minimize the probability of wrongly accepting the H_0 (when H_1 is true)

$$\beta = P(x \in W | H_1)$$

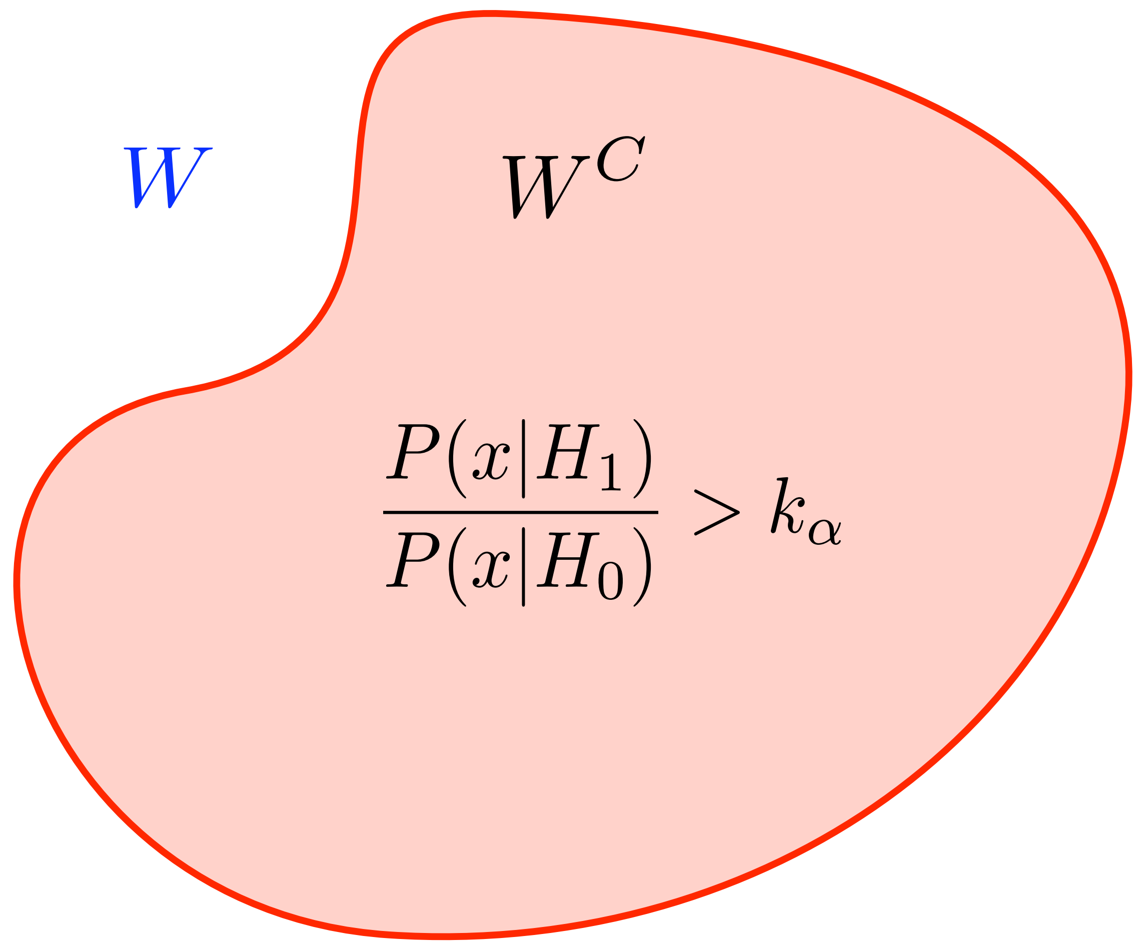
The region W that minimizes the probability of wrongly accepting H_0 is just a contour of the Likelihood Ratio

$$\frac{P(x|H_1)}{P(x|H_0)} > k_\alpha$$

Any other region of the same size will have less power

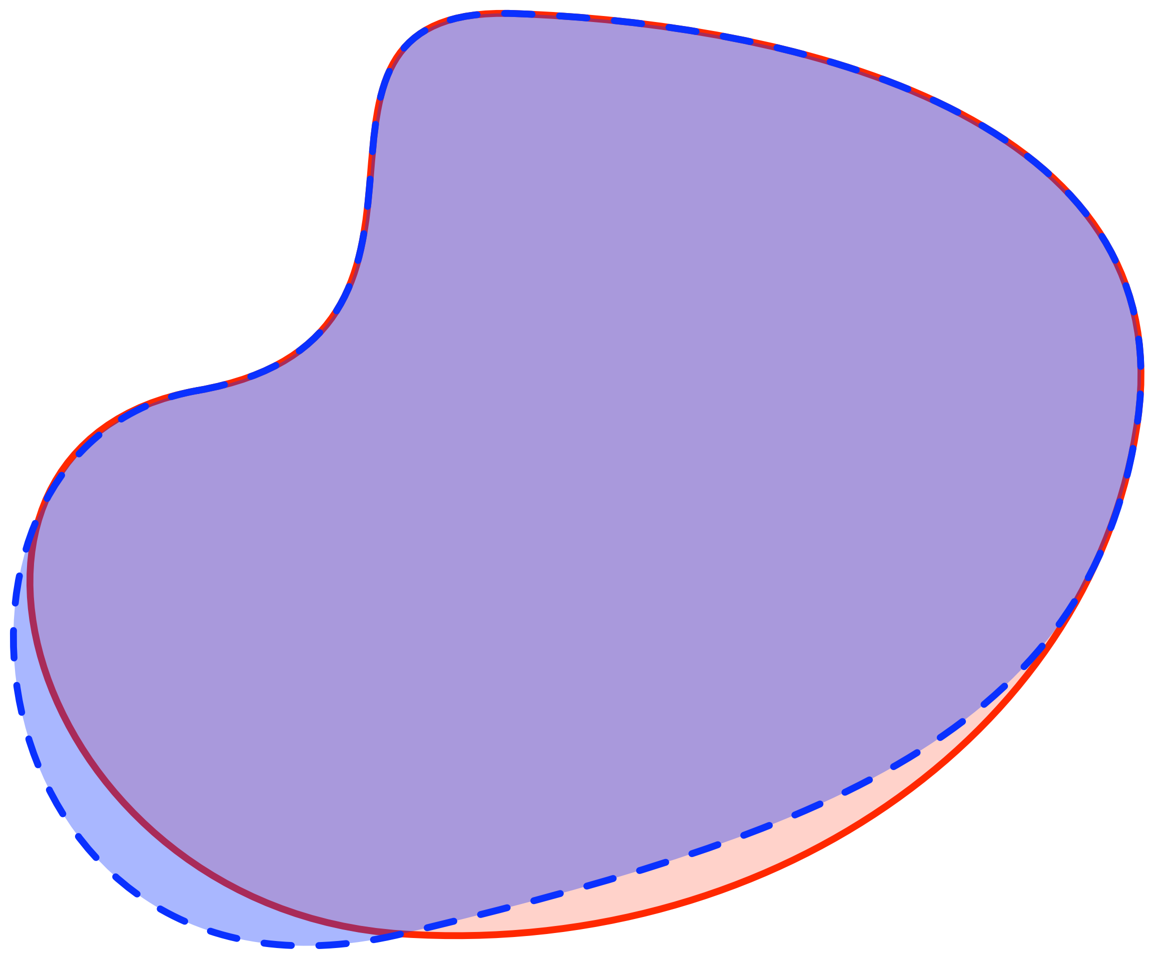
The likelihood ratio is an example of a **Test Statistic**, eg. a real-valued function that summarizes the data in a way relevant to the hypotheses that are being tested

A SHORT PROOF OF NEYMAN-PEARSON



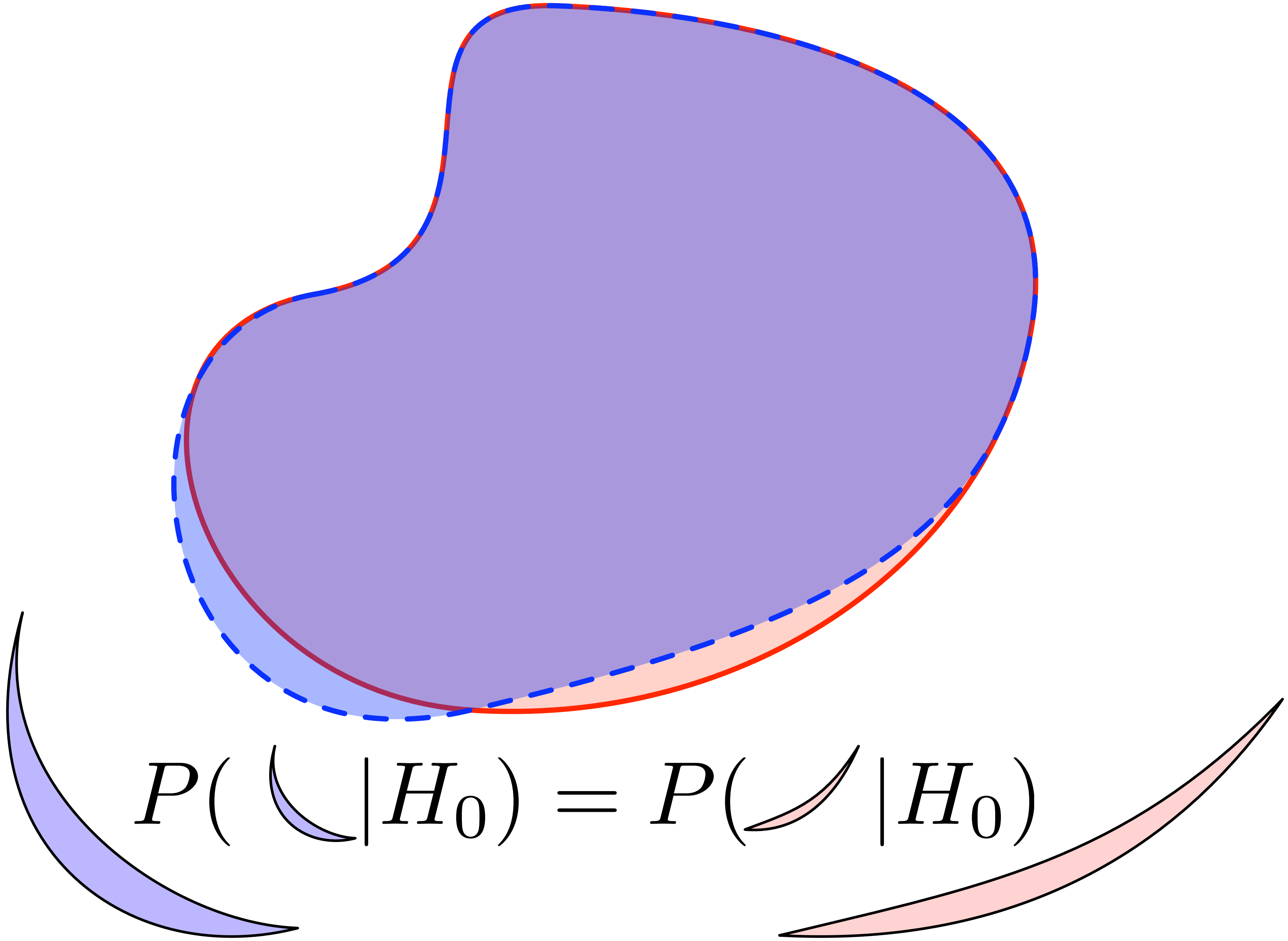
Consider the contour of the likelihood ratio that has size a given size (eg. probability under H_0 is 1-α)

A SHORT PROOF OF NEYMAN-PEARSON



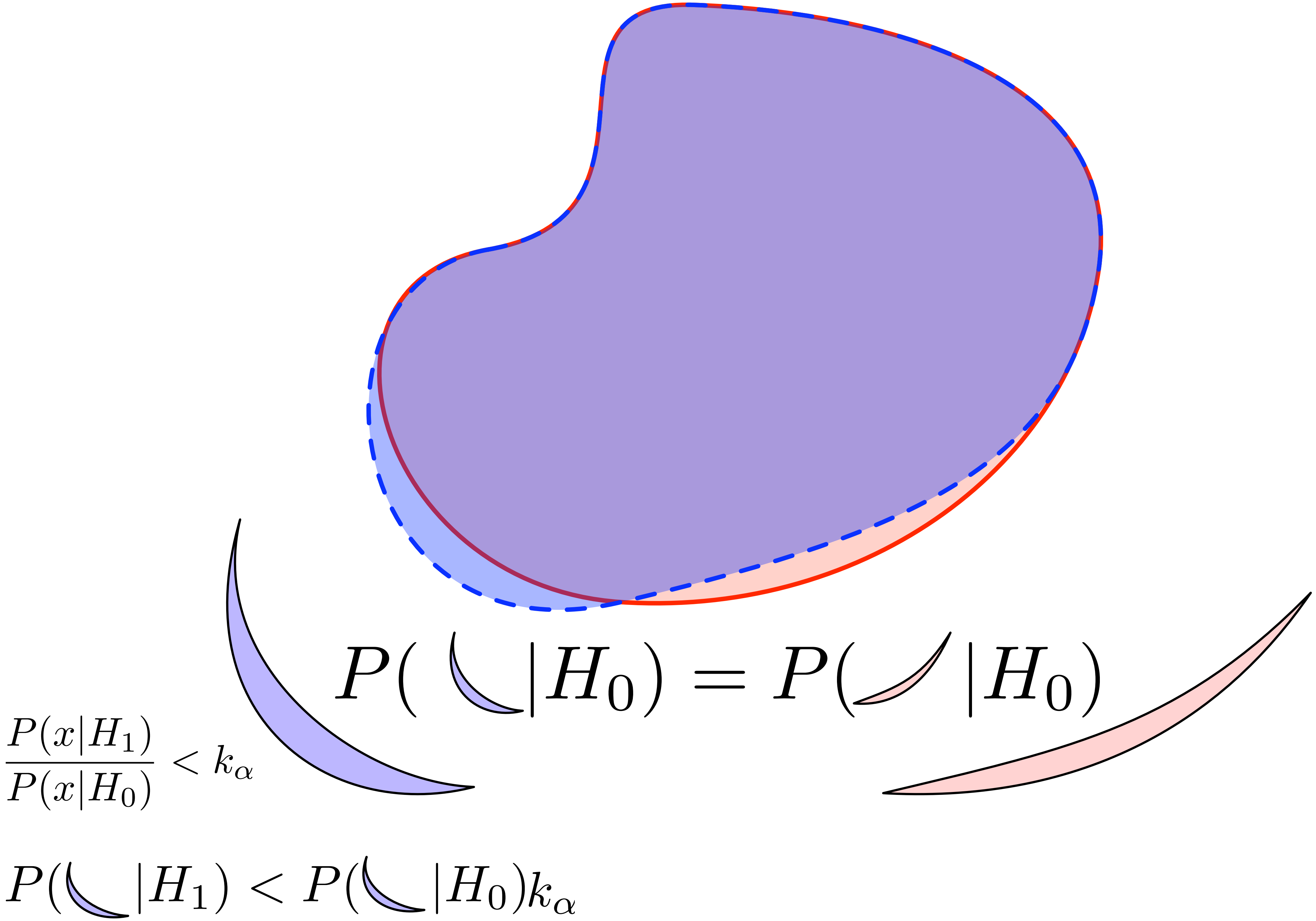
Now consider a variation on the contour that has the same size

A short proof of Neyman-Pearson



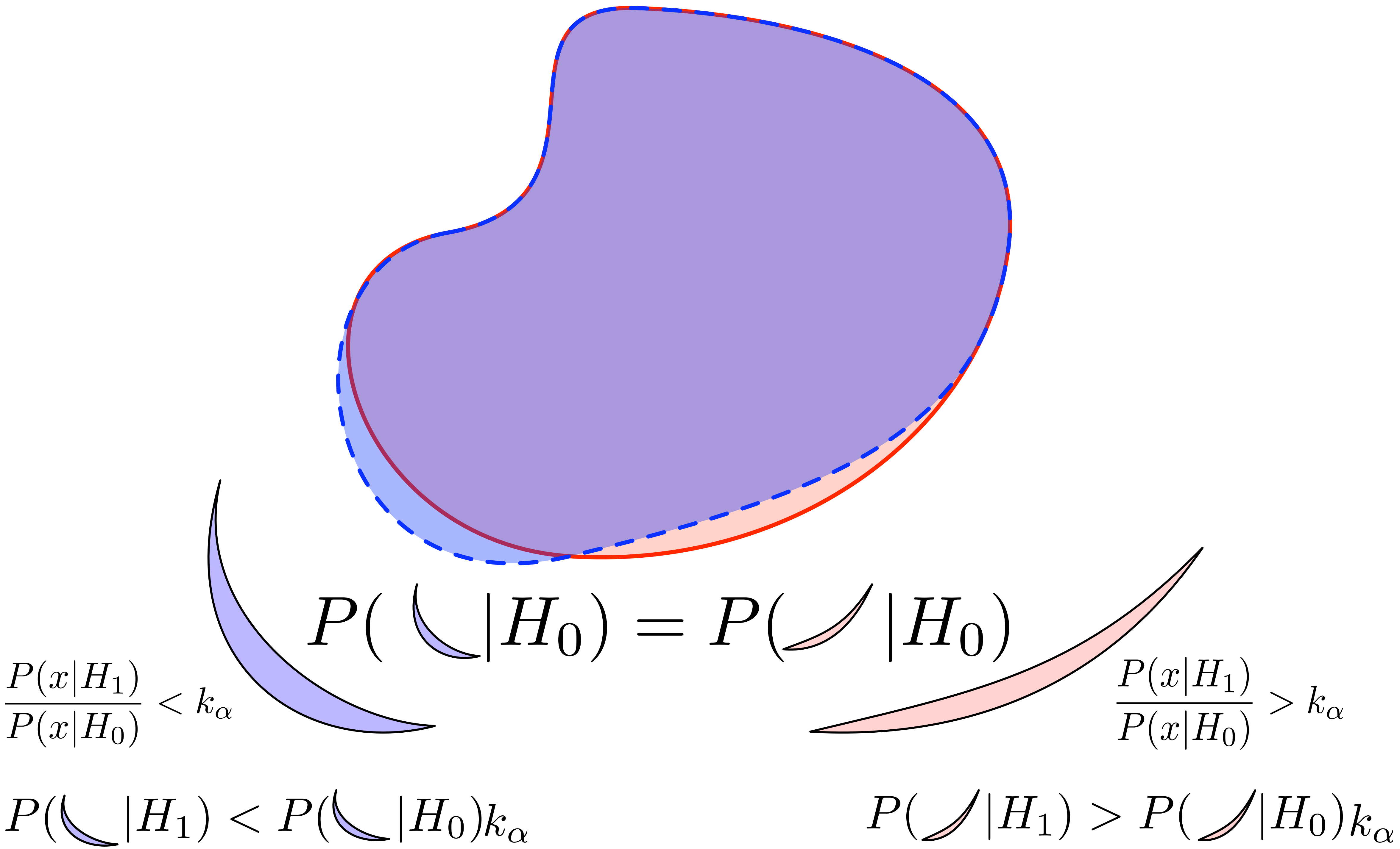
Now consider a variation on the contour that has the same size (eg. same probability under H_0)

A SHORT PROOF OF NEYMAN-PEARSON



Because the new area is outside the contour of the likelihood ratio, we have an inequality

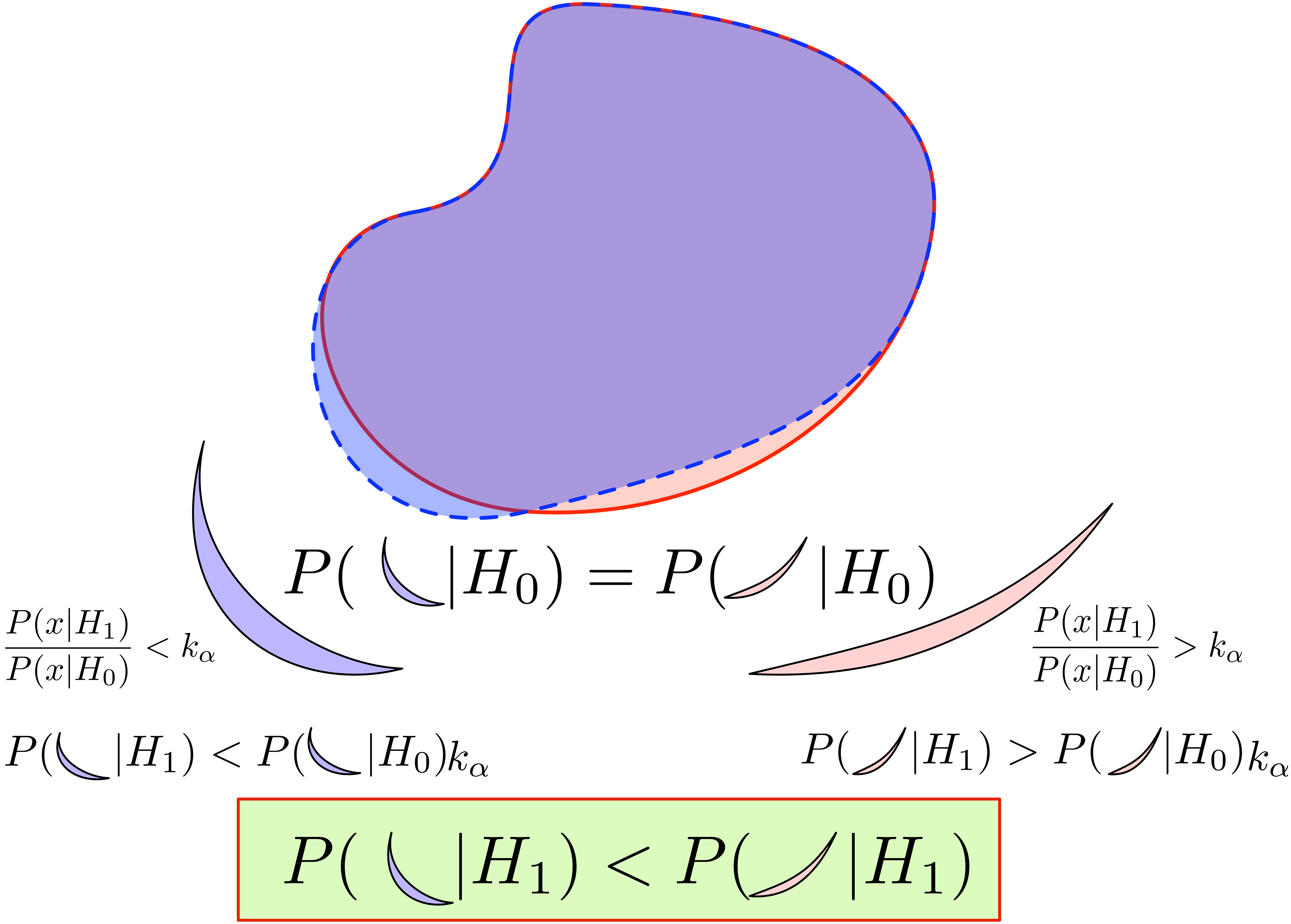
A SHORT PROOF OF NEYMAN-PEARSON



And for the region we lost, we also have an inequality

Together they give...

A short proof of Neyman-Pearson



The new region region has less power.

STATISTICAL DECISION THEORY IN 1 SLIDE

Θ - States of nature; X - possible observations; A - action to be taken

$f(x|\theta)$ - statistical model; $\pi(\theta)$ - prior

$\delta: X \to A$ - **decision rule** (take some action based on observation)

$L: \Theta \times A \to \mathbb{R}$ - **loss function**, real-valued function true parameter and action

$R(\theta,\delta) = E_{f(x|\theta)}[L(\theta, \delta)]$ - **risk**

- A decision δ^* rule **dominates** a decision rule δ if and only if $R(\theta,\delta^*) \leq R(\theta,\delta)$ for all θ, and the inequality is strict for some θ.
- A decision rule is **admissible** if and only if no other rule dominates it; otherwise it is inadmissible

$r(\pi, \delta) = E_{\pi(\theta)}[R(\theta,\delta)]$ - **Bayes risk** (expectation over θ w.r.t. prior and possible observations)

$\rho(\pi, \delta \mid x) = E_{\pi(\theta|x)}[L(\theta,\delta(x))]$ - **expected loss** (expectation over θ w.r.t. posterior $\pi(\theta|x)$)

- δ' is a (generalized) Bayes rule if it minimizes the expected loss
- under mild conditions every admissible rule is a (generalized) Bayes rule (**with respect to some prior** —possibly an improper one—that favors distributions where that rule achieves low risk). Thus, in frequentist decision theory it is sufficient to consider only (generalized) Bayes rules.
- Conversely, while Bayes rules with respect to proper priors are virtually always admissible, generalized Bayes rules corresponding to improper priors need not yield admissible procedures. Stein's example is one such famous situation.

The longest highest bridge in the world

- Optimality theory: Data X. Model $f(x|\theta), \theta \in \Theta$.
- Decision problem: observe X, make decision $d(X)$.
- Lose $L(d(X), \theta)$ – real valued.
- Judge quality of $d(X)$ by long run average risk:

$$R(d, \theta) = \langle L(d(X), \theta \rangle_\theta = \mathrm{E}\left[L(d(X), \theta | \theta\right].$$

- Key idea: admissibility.
- Procedure d_1 is better than d_2 if, for *all* θ,

$$R(d_1, \theta) < R(d_2, \theta).$$

- We call d_2 *inadmissible*.

Admissibility and Bayes

Theorem

Every admissible procedure is Bayes.

Theorem

Every Bayes procedure is admissible

Written separately because neither is quite right.
But meaning is – sensible procedures need to be Bayes.
Not always an easy restriction to impose – but wise, in my view, to remember.

Famous example – Neyman-Pearson lemma

- Data X with density f_0 or f_1.
- Decision: observe X guess which density. Hypothesis testing.
- Loss: 1 if wrong, 0 if right.
- Risk is

$$(P_0(\text{Reject}), P_1(\text{Accept}))$$

- Neyman Pearson say minimize second component subject to constraint on first.

Implied Priors

- Langrange multipliers. Minimize

$$P_1(\text{Accept}) + \lambda P_0(\text{Reject}) = \beta + \lambda\alpha.$$

- Same as Bayes for prior $P(f_1 \text{ true}) = 1/(1+\lambda)$.
- Then adjust prior (λ) to find Bayes procedure which satisfies constraint.
- Notice that $\lambda/(1+\lambda) = P(H_o)$.
- Procedure implies (at least one) prior.

Motivation for likelihood-free inference
& machine learning

Overview of Predictions

$$
\begin{aligned}
\mathcal{L}_{SM} = \; & \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}} \\
& + \underbrace{\bar{L}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})L \; + \; \bar{R}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu})R}_{\text{kinetic energies and electroweak interactions of fermions}} \\
& + \underbrace{\frac{1}{2}\left|(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})\,\phi\right|^{2} - V(\phi)}_{W^{\pm},Z,\gamma,\text{and Higgs masses and couplings}} \\
& + \underbrace{g''(\bar{q}\gamma^{\mu}T_{a}q)\,G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} \; + \; \underbrace{(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.)}_{\text{fermion masses and couplings to Higgs}}
\end{aligned}
$$

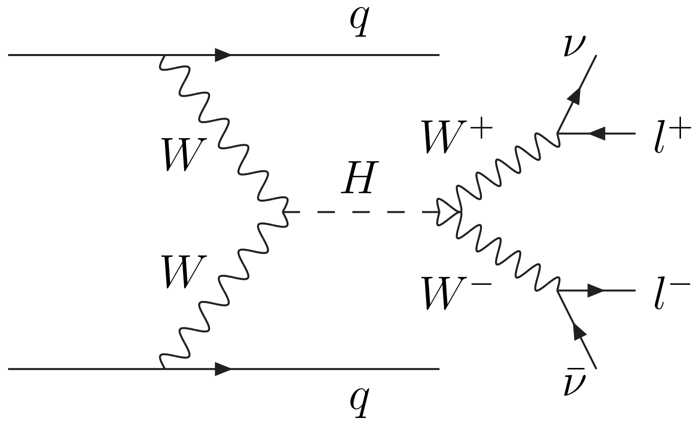
1) The language is Quantum Field Theory

OVERVIEW OF PREDICTIONS

$$
\begin{aligned}
\mathcal{L}_{SM} = \; & \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a}_{\text{kinetic energies and self-interactions of the gauge bosons}} \\
& + \underbrace{\bar{L}\gamma^\mu(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)L + \bar{R}\gamma^\mu(i\partial_\mu - \frac{1}{2}g'YB_\mu)R}_{\text{kinetic energies and electroweak interactions of fermions}} \\
& + \underbrace{\frac{1}{2}\left|(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)\,\phi\right|^2 - V(\phi)}_{W^\pm, Z, \gamma, \text{and Higgs masses and couplings}} \\
& + \underbrace{g''(\bar{q}\gamma^\mu T_a q)\,G^a_\mu}_{\text{interactions between quarks and gluons}} + \underbrace{(G_1\bar{L}\phi R + G_2\bar{L}\phi_c R + h.c.)}_{\text{fermion masses and couplings to Higgs}}
\end{aligned}
$$

1) The language is Quantum Field Theory

2) Feynman Diagrams are used to predict high-energy interaction among fundamental particles

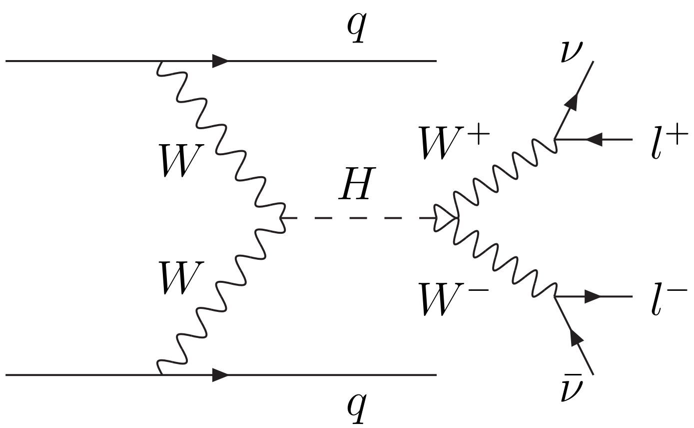


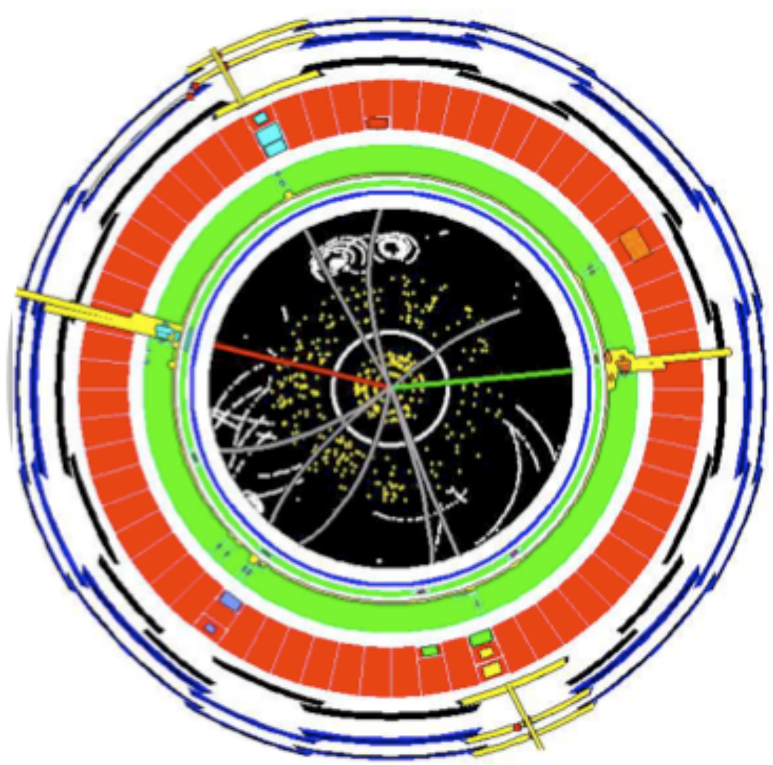
Overview of Predictions

$$
\begin{aligned}
\mathcal{L}_{SM} = \; & \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}} \\
& + \underbrace{\bar{L}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})L + \bar{R}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu})R}_{\text{kinetic energies and electroweak interactions of fermions}} \\
& + \underbrace{\frac{1}{2}\left|(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})\,\phi\right|^{2} - V(\phi)}_{W^{\pm}, Z, \gamma, \text{and Higgs masses and couplings}} \\
& + \underbrace{g''(\bar{q}\gamma^{\mu}T_{a}q)\,G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} + \underbrace{(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.)}_{\text{fermion masses and couplings to Higgs}}
\end{aligned}
$$

1) The language is Quantum Field Theory

2) Feynman Diagrams are used to predict high-energy interaction among fundamental particles





3) The interaction of outgoing particles with the detector is simulated.

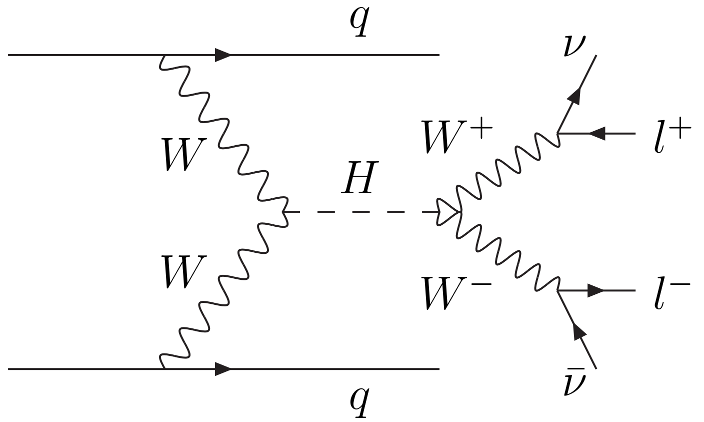
>100 million sensors

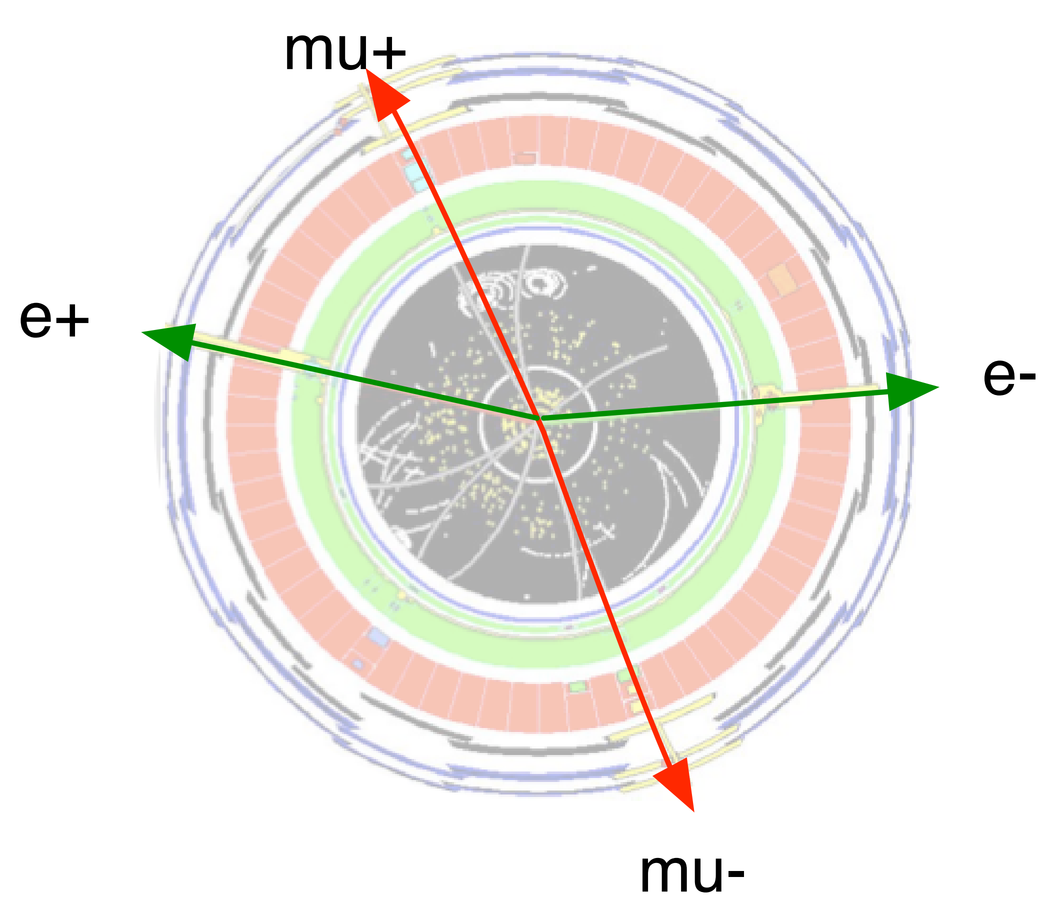
Overview of Predictions

$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$
$$+ \underbrace{\bar{L}\gamma^\mu(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)L + \bar{R}\gamma^\mu(i\partial_\mu - \frac{1}{2}g'YB_\mu)R}_{\text{kinetic energies and electroweak interactions of fermions}}$$
$$+ \underbrace{\frac{1}{2}\left|(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)\phi\right|^2 - V(\phi)}_{W^\pm, Z, \gamma, \text{and Higgs masses and couplings}}$$
$$+ \underbrace{g''(\bar{q}\gamma^\mu T_a q)G^a_\mu}_{\text{interactions between quarks and gluons}} + \underbrace{(G_1\bar{L}\phi R + G_2\bar{L}\phi_c R + h.c.)}_{\text{fermion masses and couplings to Higgs}}$$

1) The language is Quantum Field Theory

2) Feynman Diagrams are used to predict high-energy interaction among fundamental particles





3) The interaction of outgoing particles with the detector is simulated.

>100 million sensors

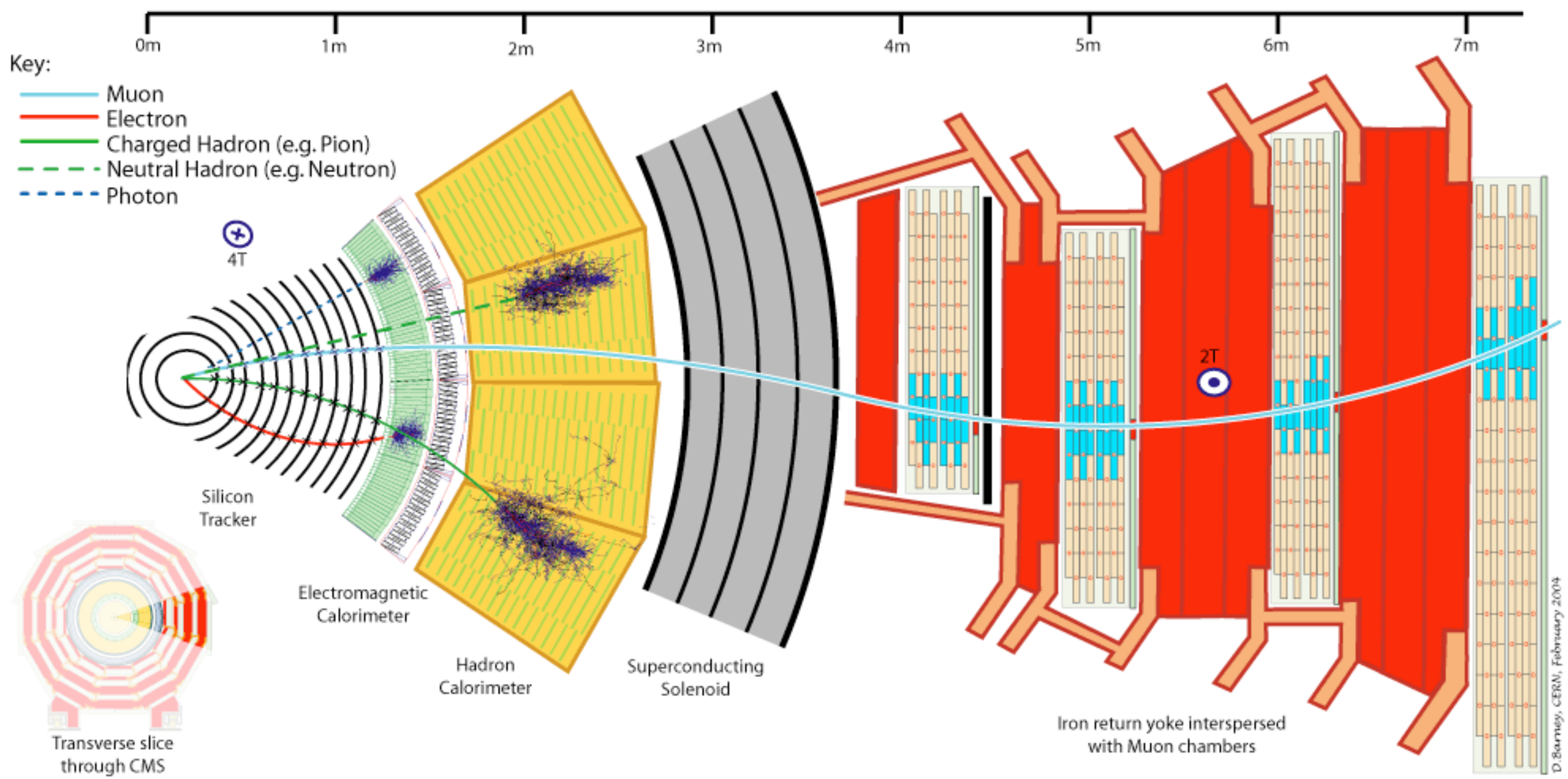
4) Finally, we run particle identification algorithms on the simulated data as if they were from real collisions.

~10-30 features describe interesting part

Conceptually: Prob(detector response | particles)

Implementation: Monte Carlo integration over micro-physics

Consequence: cannot evaluate likelihood for a given event



Conceptually: Prob(detector response | particles)

Implementation: Monte Carlo integration over micro-physics

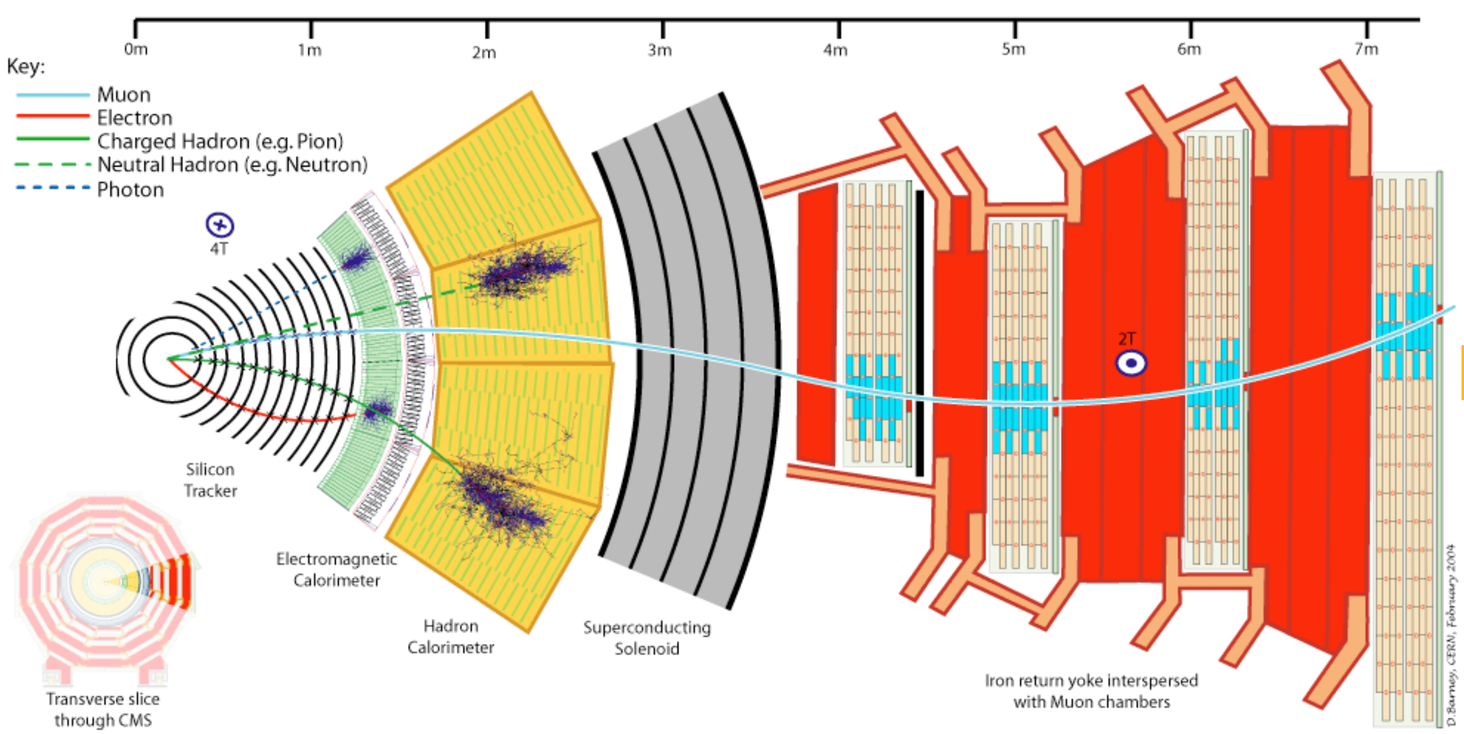
Consequence: cannot evaluate likelihood for a given event

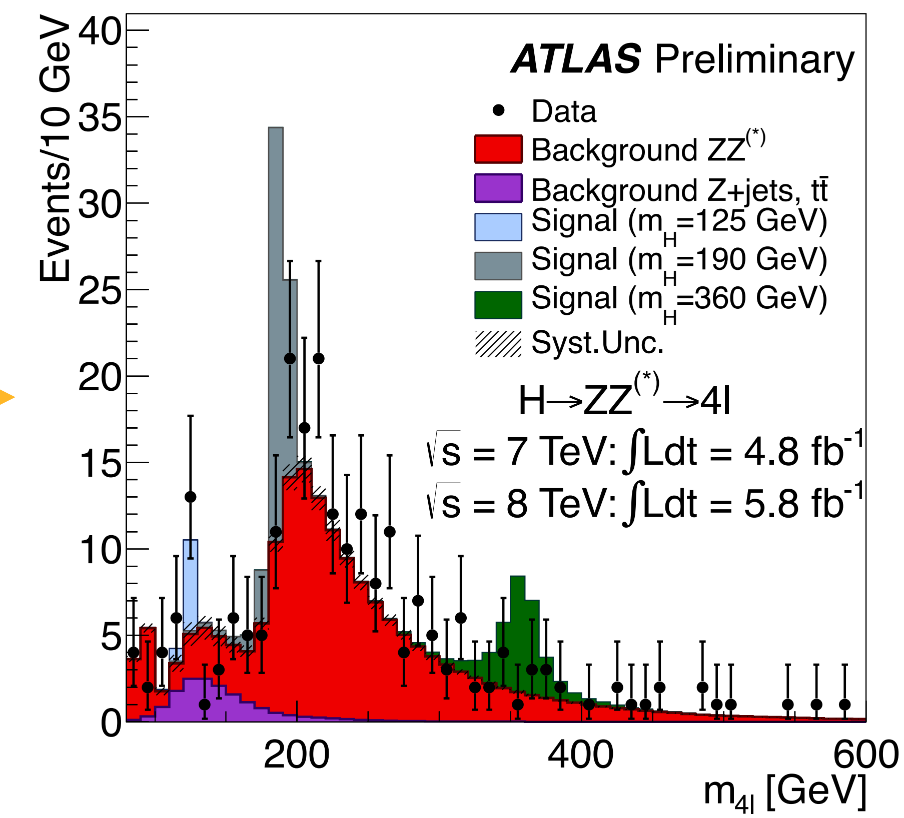
This motivates a new class of algorithms for what is called **likelihood-free inference**, which only require ability to generate samples from the simulation in the "forward mode"

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 variable or feature

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

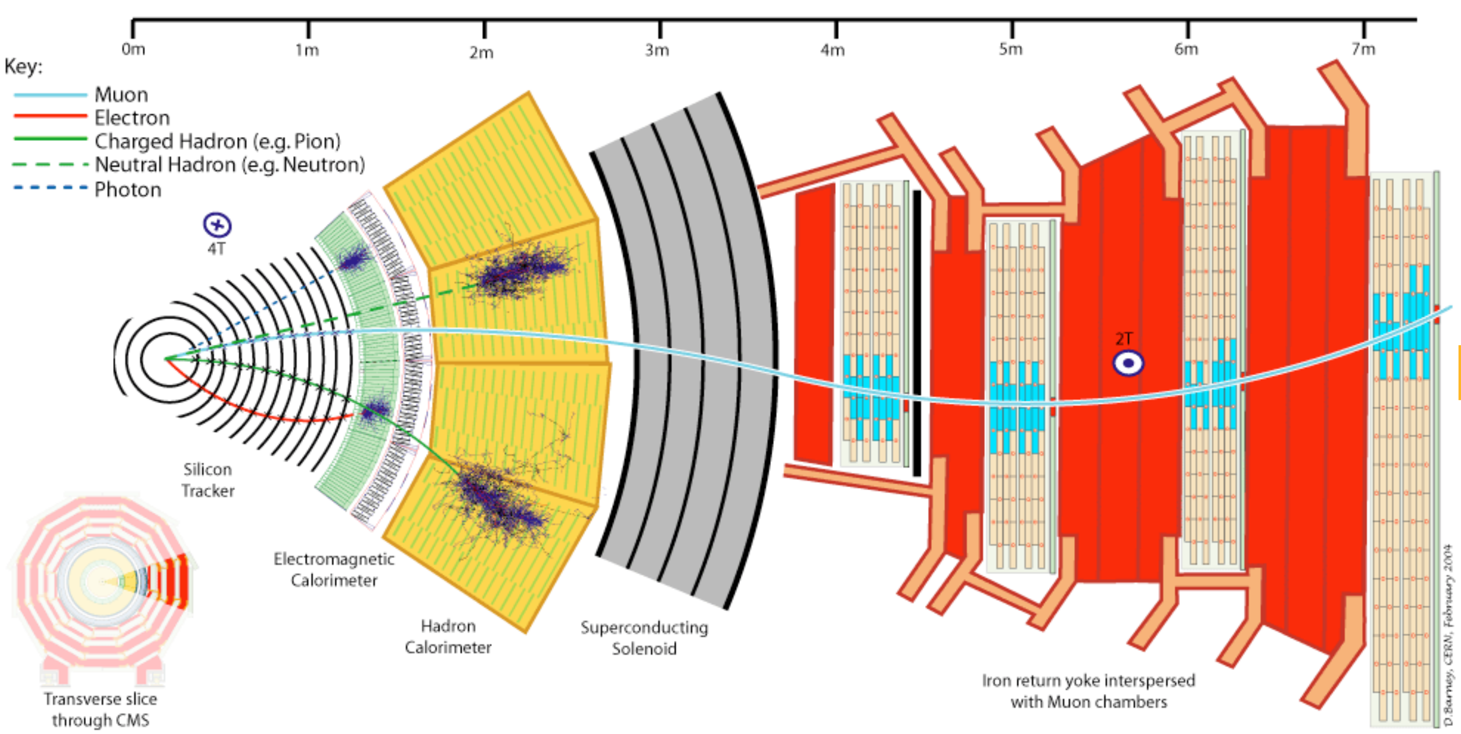


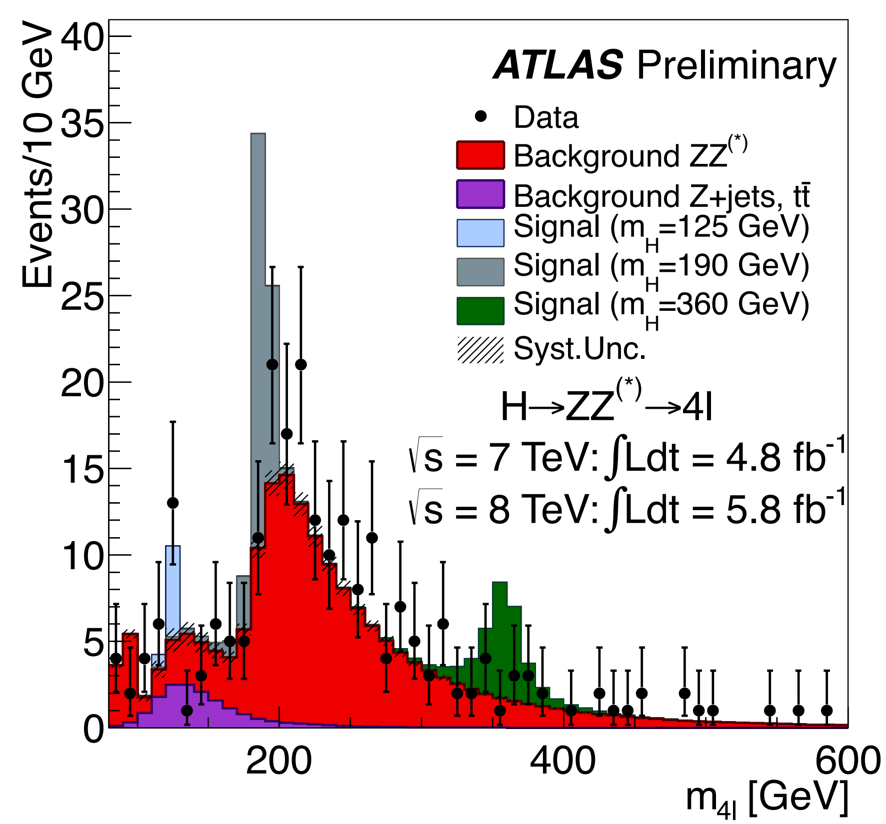


10^8 SENSORS → 1 REAL-VALUED QUANTITY

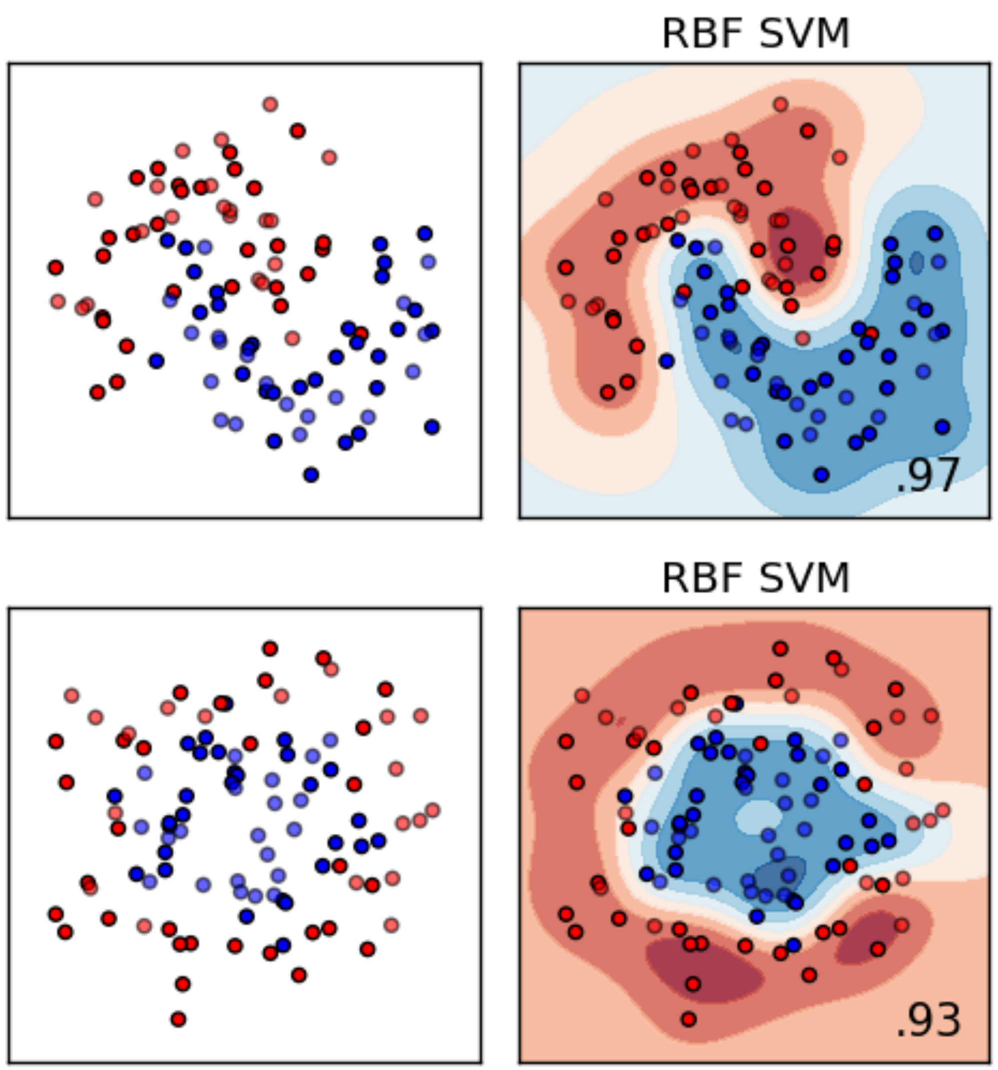
Most measurements and searches for new particles at the LHC are based on the distribution of a single variable or feature

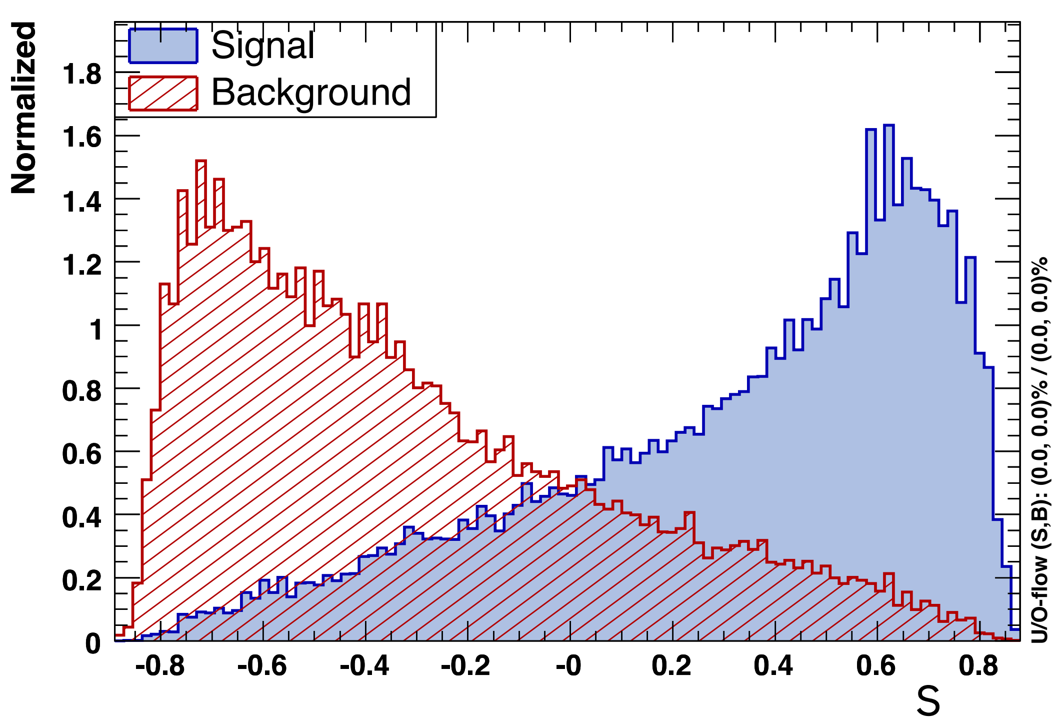
- choosing a good variable (feature engineering) is a task for a skilled physicist and tailored to the goal of measurement or new particle search
- likelihood p(x|θ) **approximated** using histograms (univariate density estimation)





This doesn't scale if x is high dimensional!

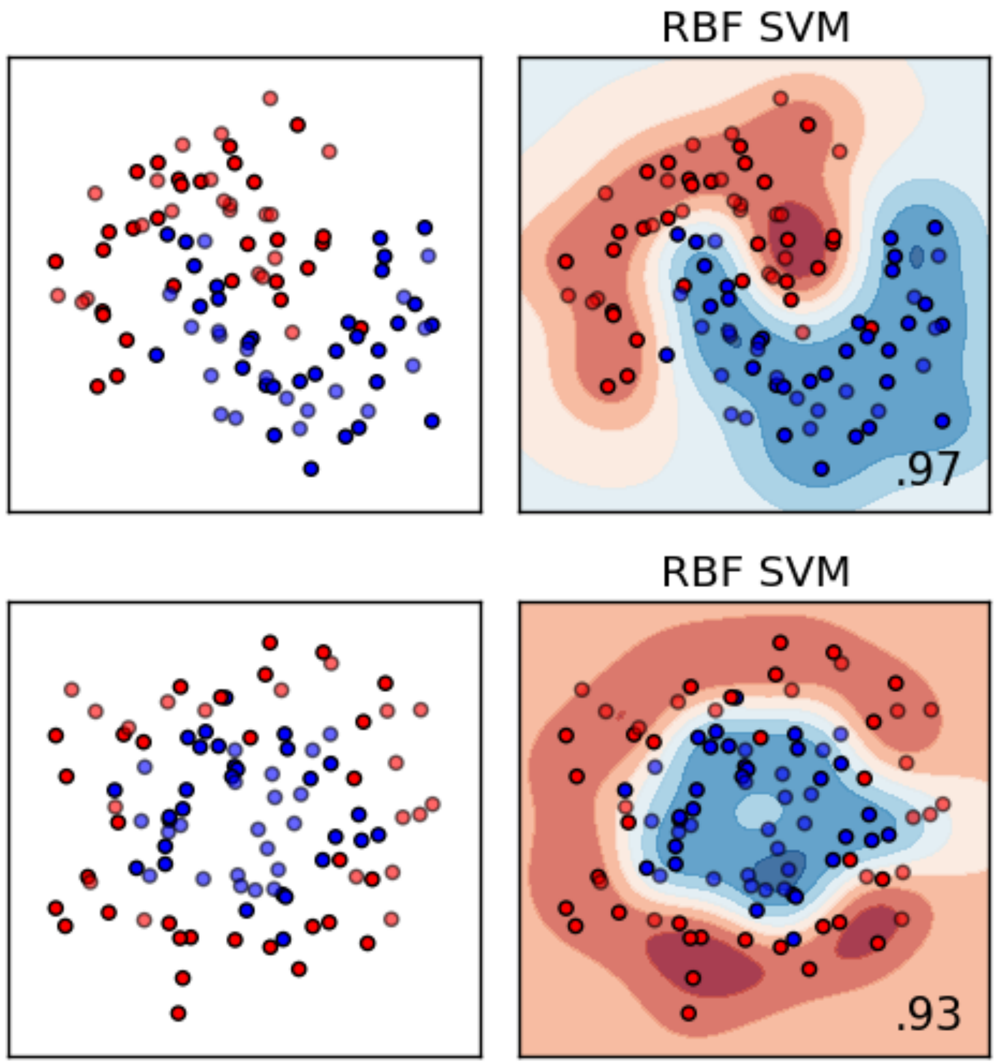


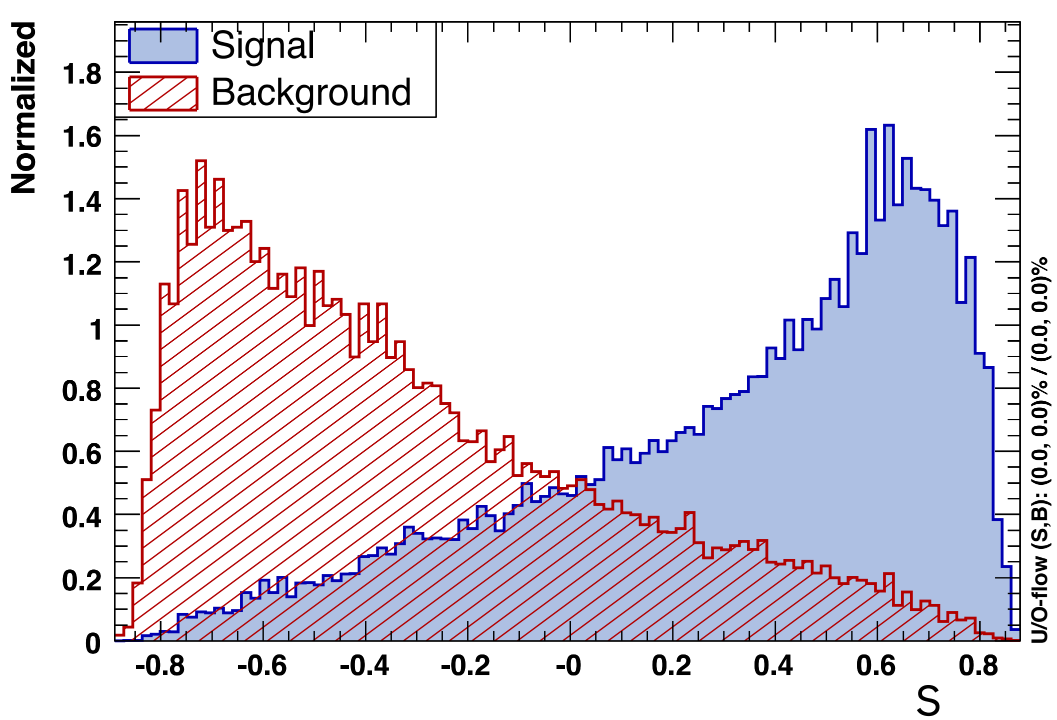


Common to use machine learning classifiers to separate signal (H_1) vs. background (H_0)

- want a function s: X→ Y that maps signal to y=1 and background to y=0
- **calculus of variations**: find function s(x) that minimizes ***loss***:

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





- **applied calculus of variations**: find function s(x) that minimizes ***loss***:

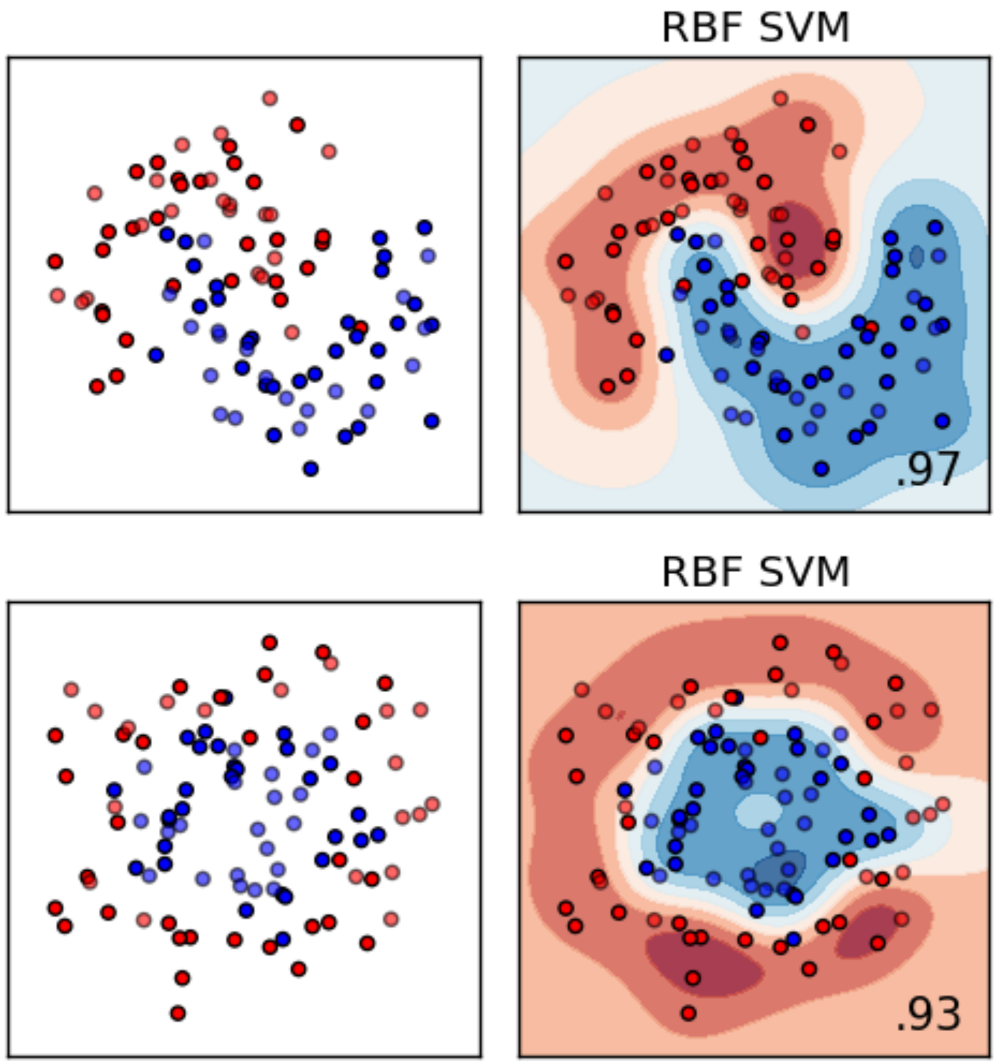
$$L[s] = \int p(x|H_0)\,(0 - s(x))^2\,dx + \int p(x|H_1)\,(1 - s(x))^2 dx$$

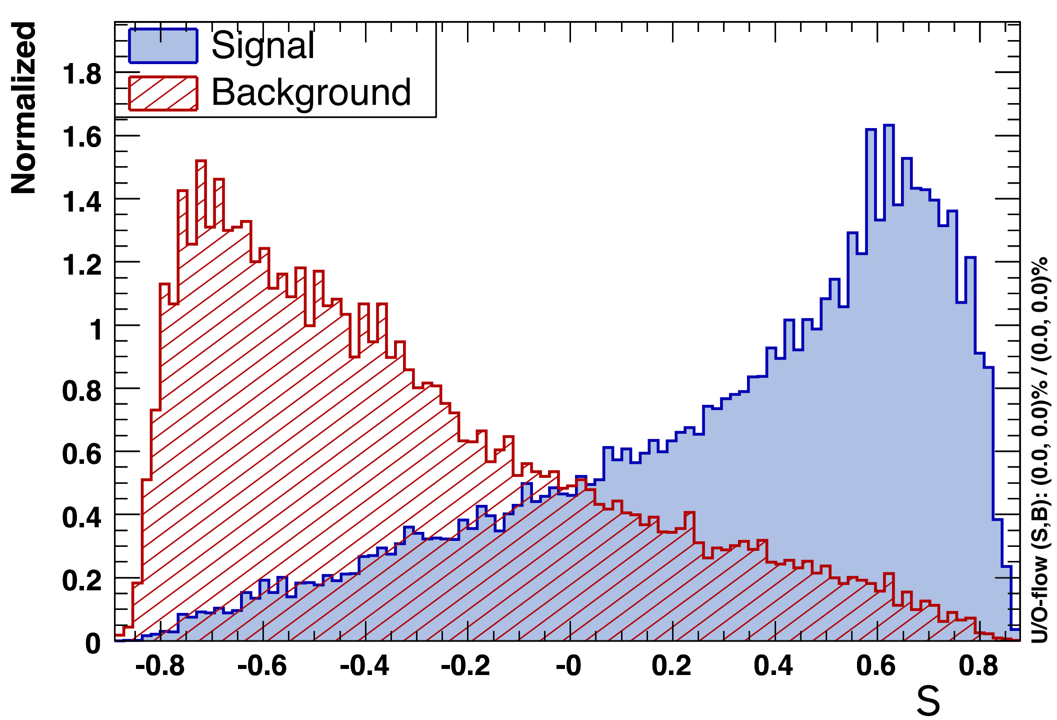
- i.e. approximate the optimal classifier

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

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

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





- **applied calculus of variations**: 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

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

Loss Functions

What function r(x) minimizes the "cross-entropy" loss?

$$L[r] = -\int \underbrace{p(x)\log r(x)}_{F(x,r)}\, dx$$

- Subject to $\int r(x)dx = 1$

What function r(x) minimizes the "cross-entropy" loss?

$$L[r] = -\int \underbrace{p(x)\log r(x)}_{F(x,r)}\,dx \approx \frac{1}{N}\sum_{i=1}^{N}\log r(x_i)$$

- Subject to $\int r(x)dx = 1$

What function r(x) minimizes the "cross-entropy" loss?

$$L[r] = -\int \underbrace{p(x) \log r(x)}_{F(x,r)} dx \approx \frac{1}{N} \sum_{i=1}^{N} \log r(x_i)$$

- Subject to $\int r(x) dx = 1$

Euler-Lagrange Equation w/ Lagrange-multiplier

$$L[r, \lambda] = F(x, r) + \lambda r(x)$$

$$\underbrace{\frac{d}{dx}\left(\frac{\delta L}{\delta r'}\right)}_{=0} - \frac{\delta L}{\delta r} = 0 \qquad \frac{\delta L}{\delta r} = 0 = \frac{-p(x)}{r(x)} + \lambda$$

$$r(x) = p(x)/\lambda$$

imposing the constraint gives $\lambda = 1$ thus $r(x) = p(x)$

What function r(x) minimizes the squared loss?

$$L[r] = -\int \underbrace{p(x)(p(x) - r(x))^2}_{F(x,r)} dx$$

- Subject to $\int r(x) dx = 1$

What function r(x) minimizes the squared loss?

$$L[r] = -\int \underbrace{p(x)(p(x)-r(x))^2}_{F(x,r)}\,dx$$

- Subject to $\int r(x)dx = 1$

Euler-Lagrange Equation w/ Lagrange-multiplier

$$L[r,\lambda] = F(x,r) + \lambda r(x)$$

$$\underbrace{\frac{d}{dx}\left(\frac{\delta L}{\delta r'}\right)}_{=0} - \frac{\delta L}{\delta r} = 0 \qquad \frac{\delta L}{\delta r} = 0 = \lambda - 2p(x)(p(x)-r(x))$$

$$r(x) = p - \frac{\lambda}{2p}$$

imposing the constraint gives $\lambda = 0$ thus $r(x) = p(x)$

If we have samples from an unknown p(x): $\{x_i\}_{i=1}^N \sim p(x)$

We can effectively approximate the true cross-entropy loss:

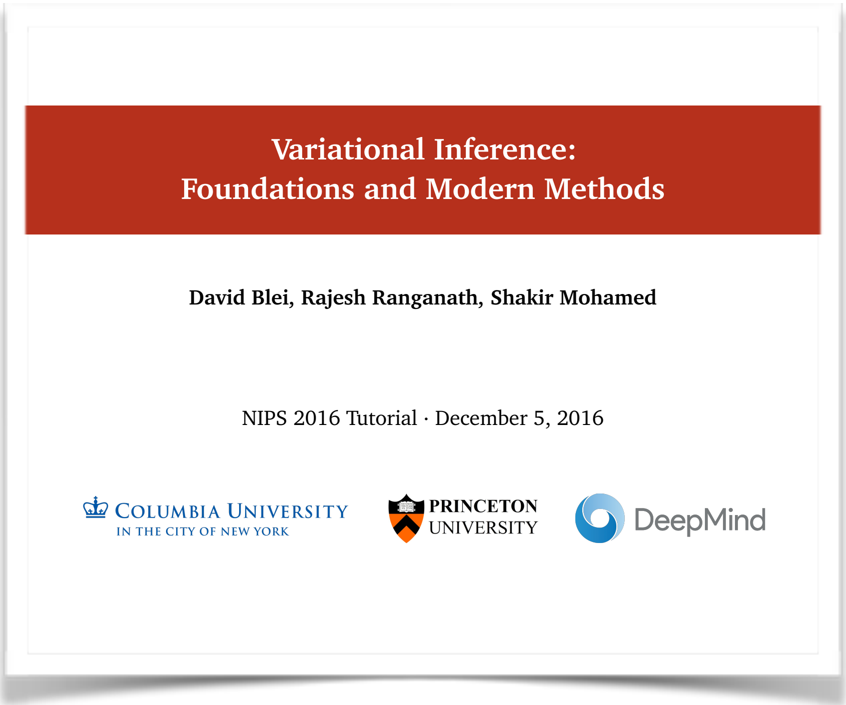
$$L[r] = -\int \underbrace{p(x)\log r(x)}_{F(x,r)}\,dx \approx \frac{1}{N}\sum_{i=1}^{N}\log r(x_i)$$

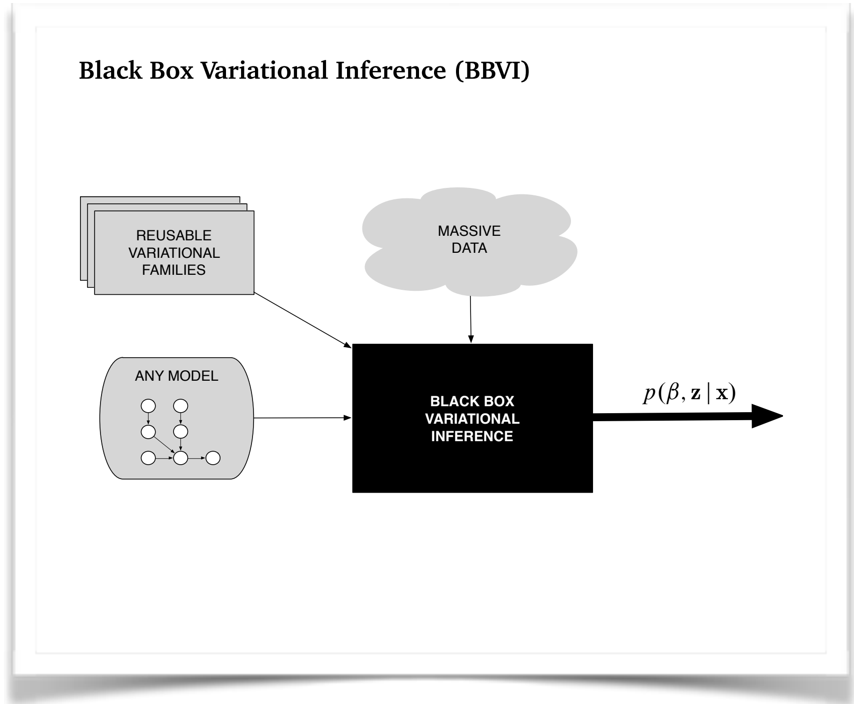
and approximate p(x) even though we can't evaluate it.

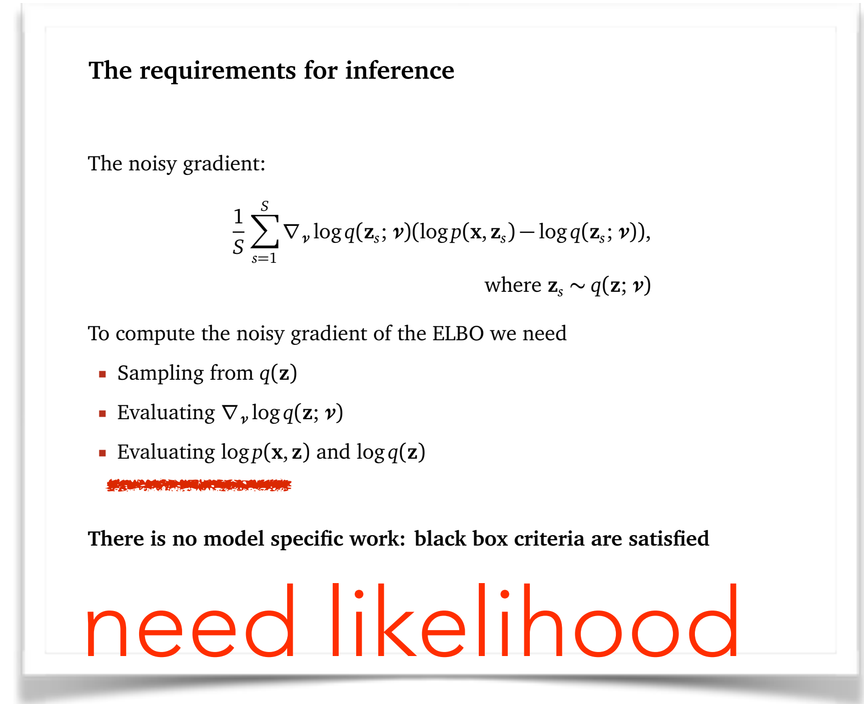
In contrast, we can't use the squared loss if since can't evaluate p(x):

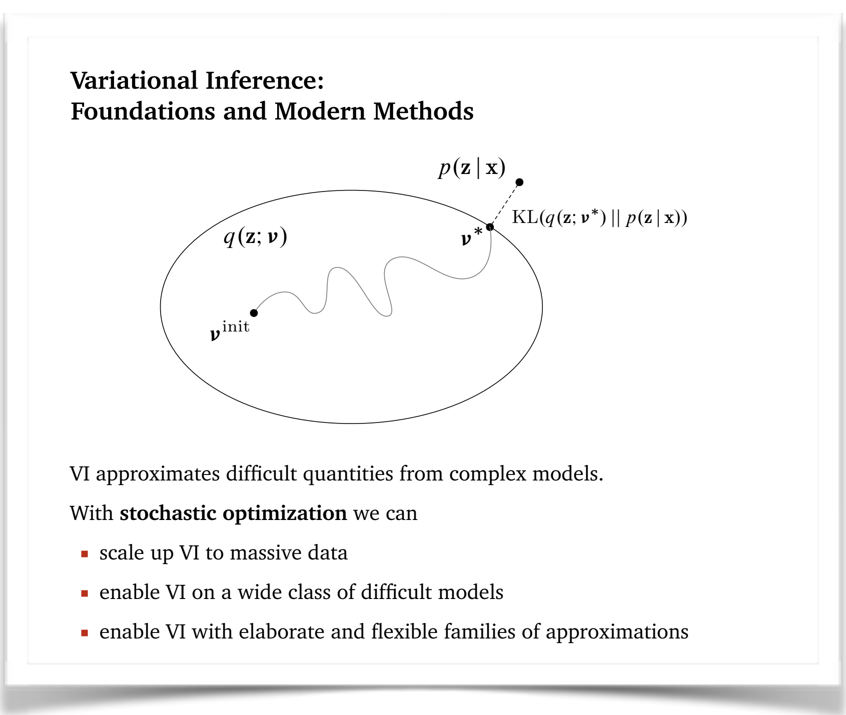
$$L[r] = -\int \underbrace{p(x)(p(x)-r(x))^2}_{F(x,r)}\,dx \approx \frac{1}{N}\sum_{i=1}^{N}\log(p(x_i)-r(x_i))^2$$

VARIATIONAL INFERENCE









How do we create complicated probability densities $p(x)$ that are tractable

and

are normalized such that $\int p(x)\, dx = 1$?

If I have a bijection: $f : X \to Z$

and an arbitrary tractable density on Z: $p(z)$

Then density on X follows from a simple change of variables

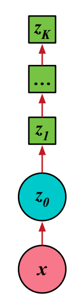
$$p(x) = p(f_\phi(x)) \left| \det \left(\frac{\partial f_\phi(x)}{\partial x_T} \right) \right|$$

Now construct neural networks f_ϕ that are bijections & optimize "cross entropy" loss

If it is a bijection, I can generate samples of x from inverse transformation $f^{-1}(z)$

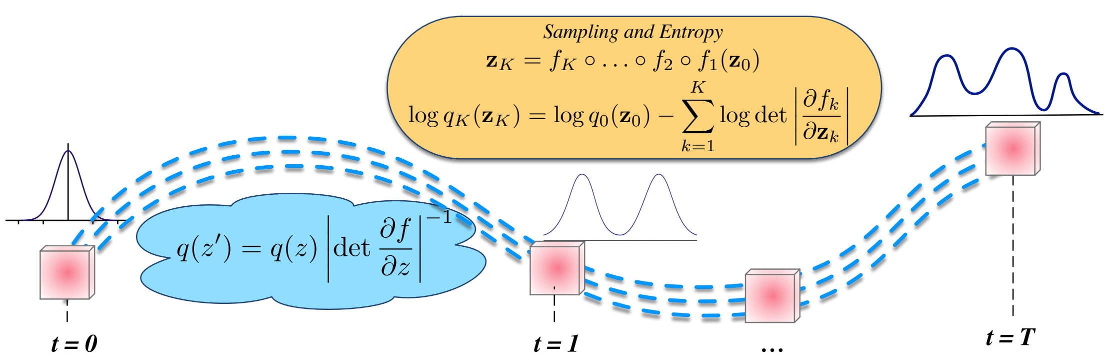
ENGINEERING BIJECTIONS

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.



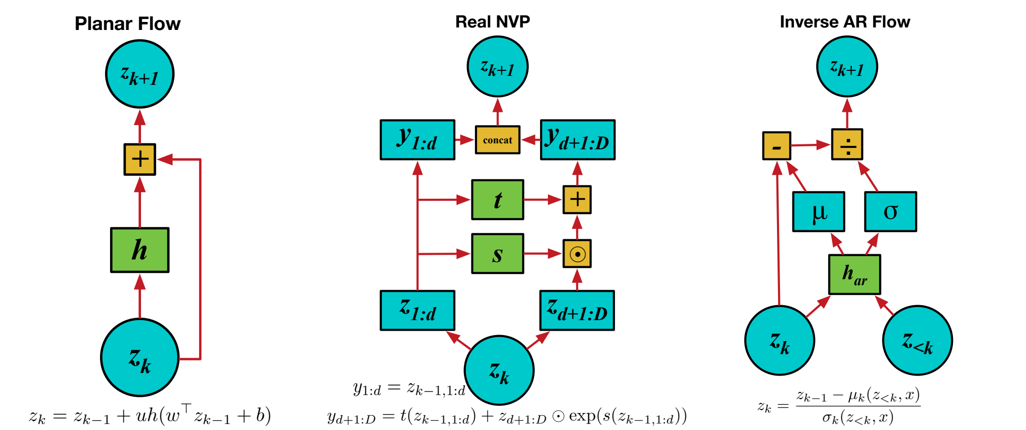
Distribution flows through a sequence of invertible transforms

[Rezende and Mohamed, 2015]

Choice of Transformation Function

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

- 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.

BIJECTIONS: FLOWS & AUTOREGRESSIVE MODELS

K.C. & G. Louppe: http://beta.briefideas.org/ideas/5c2f74aedbf3618ca180382e393c7617

Recent work in density estimation uses a bijection $f : X \rightarrow Z$ (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 $f_\theta : X \rightarrow Z$ parametrized by θ (e.g. [5] [6]).

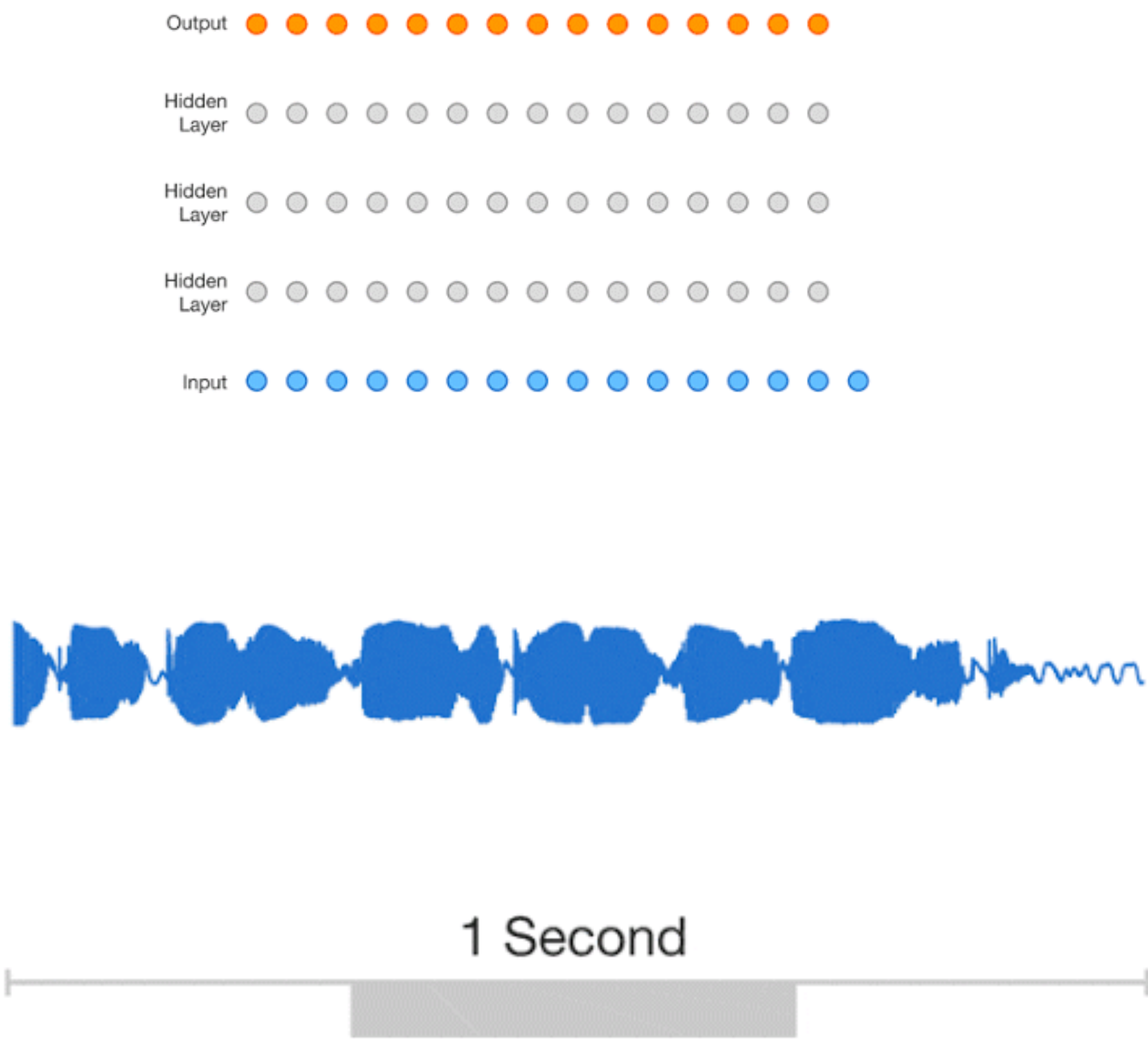
$$p(x|\theta) = p(f_{\phi;\theta}(x)) \left|\det\left(\frac{\partial f_{\phi;\theta}(x)}{\partial x_T}\right)\right|$$

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.

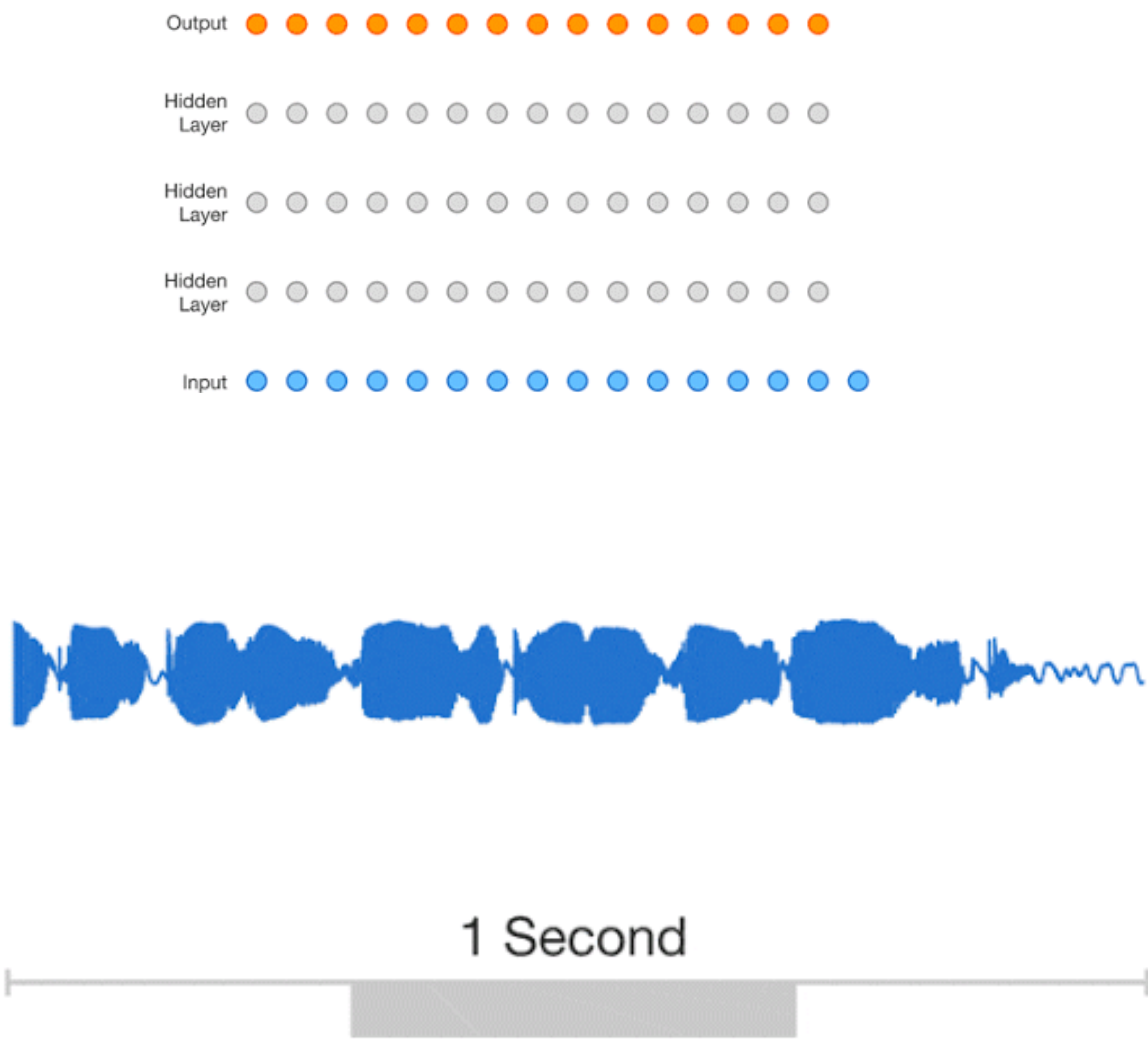
We observe that not only can this model be used as a conditional generative model $p(x|\theta)$, but it can also be used to perform asymptotically exact, amortized likelihood-free inference on θ.

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.

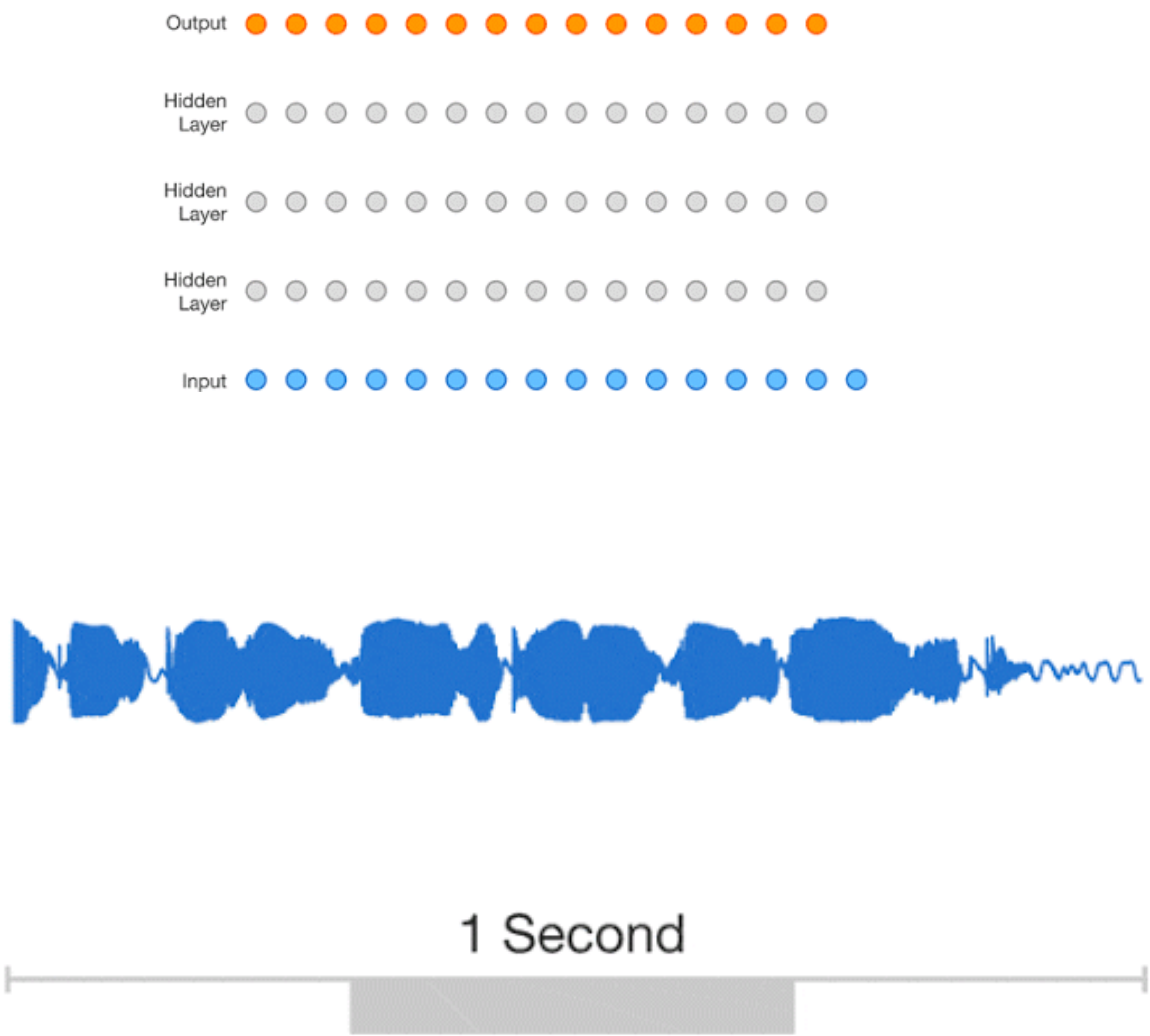
WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



WAVENET: A GENERATIVE MODEL FOR RAW AUDIO

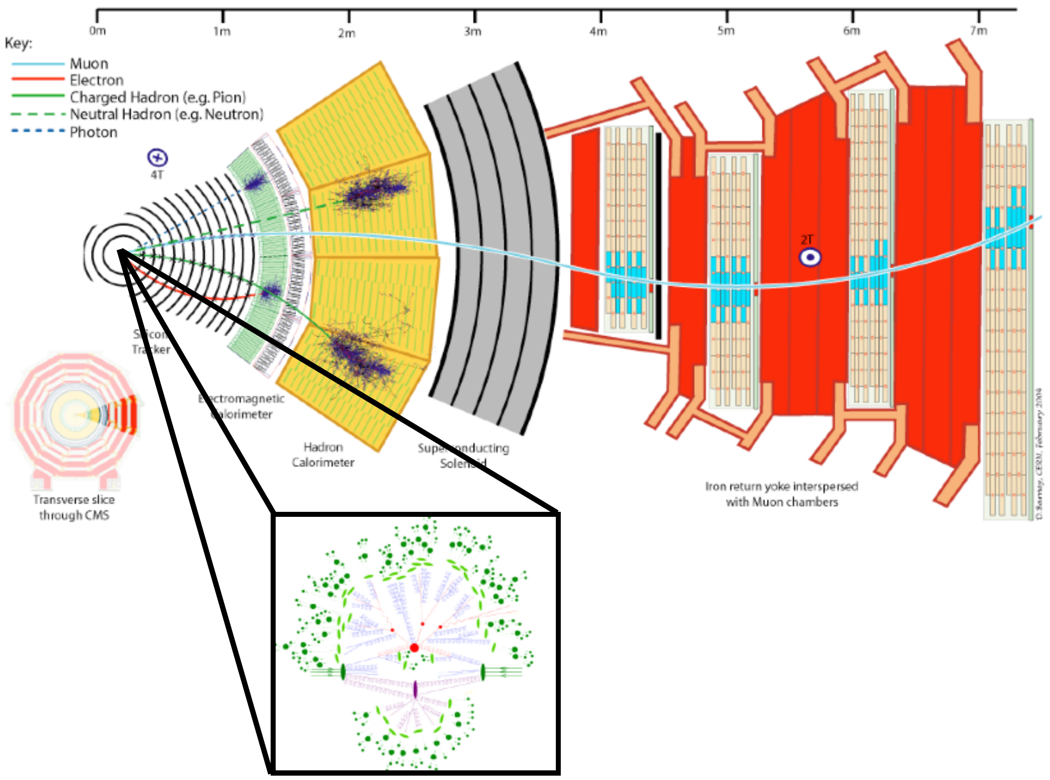


WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



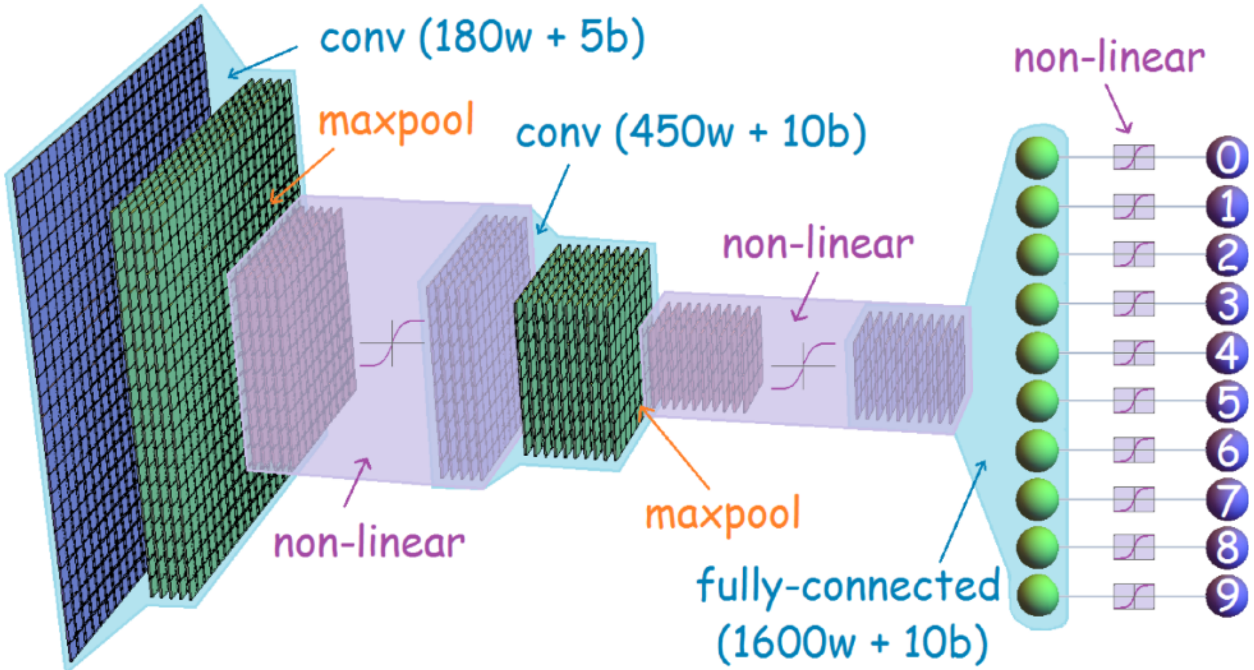
TWO APPROACHES

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]

LECTURE 3

Note: This lecture was largely on the board

Generative Adversarial Networks

- Loss functions → Adversarial minimax games
- comparison to bijective approaches
  - eg. can't use for inference

The "Data Manifold" (on the board)

- auto-encoders (on the board)

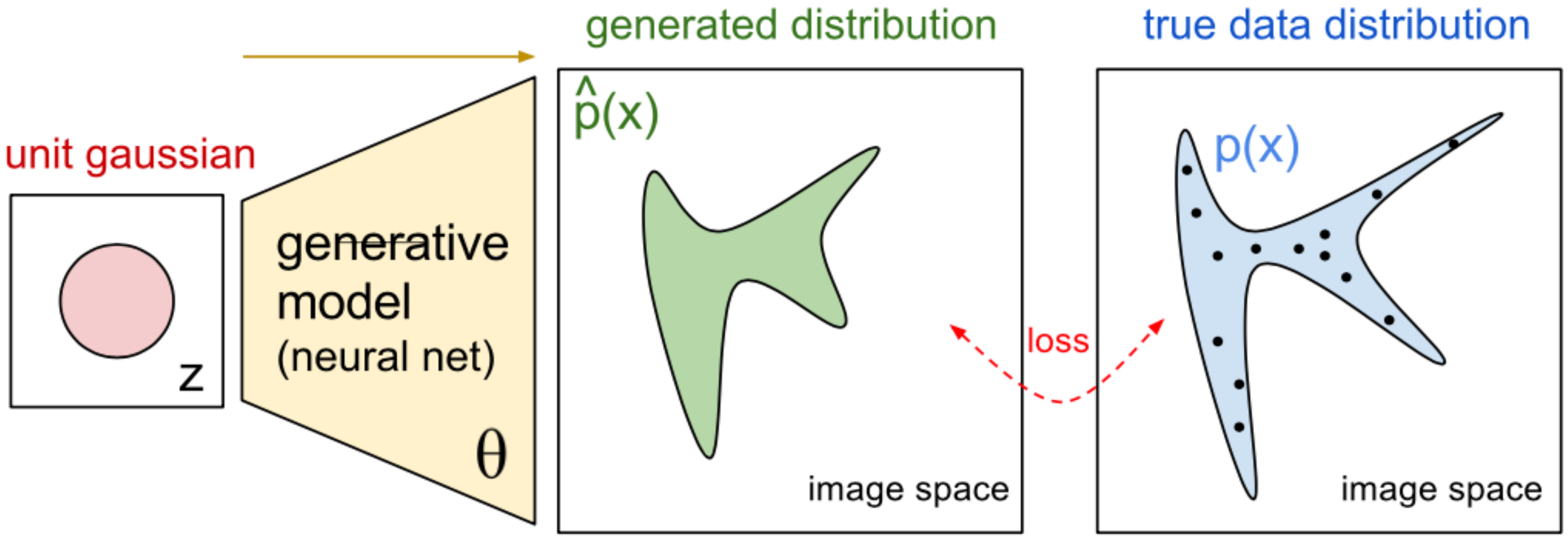
Adversarial Variational Optimization

"Learning to pivot with Adversarial Neural Networks:

Adversarial Training
(not just for GANs)

GENERATIVE ADVERSARIAL NETWORKS

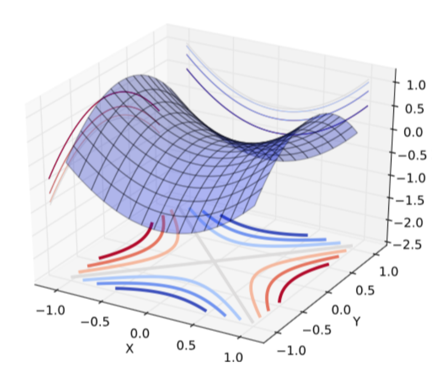
Goodfellow, et al arXiv:1406.2661
slide Gilles Louppe



- Two-player game:
  - a discriminator D,
  - a generator G;
- D is a classifier $\mathcal{X} \mapsto \{0, 1\}$ that tries to distinguish between
  - a sample from the data distribution ($D(\mathbf{x}) = 1$, for $\mathbf{x} \sim p_{\text{data}}$),
  - and a sample from the model distribution ($D(G(\mathbf{z})) = 0$, for $\mathbf{z} \sim p_{\text{noise}}$);
- G is a generator $\mathcal{Z} \mapsto \mathcal{X}$ trained to produce samples $G(\mathbf{z})$ (for $\mathbf{z} \sim p_{\text{noise}}$) that are difficult for D to distinguish from data.

$$(D^*, G^*) = \max_D \min_G V(D, G).$$

Leo is G Tom is D



G. Louppe & K.C. arXiv:1707.07113

NEW! AVO

Adversarial Variational Optimization of Non-Differentiable Simulators

Gilles Louppe[1] and Kyle Cranmer[1]
[1]*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.

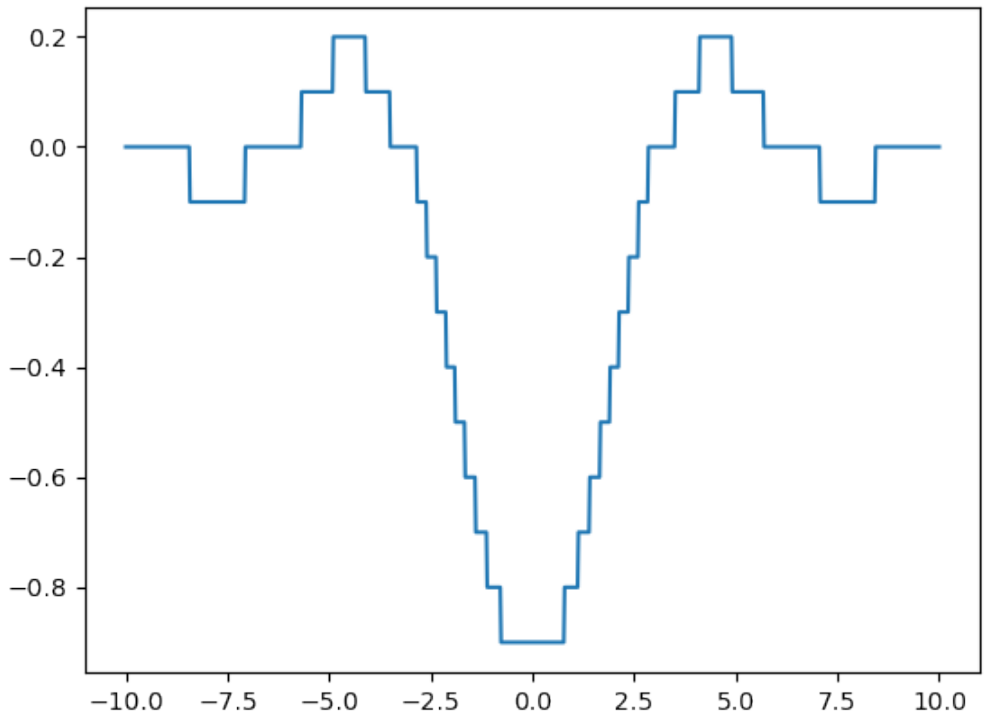
Difficulty: the simulator isn't differentiable, but there's a **trick**!

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

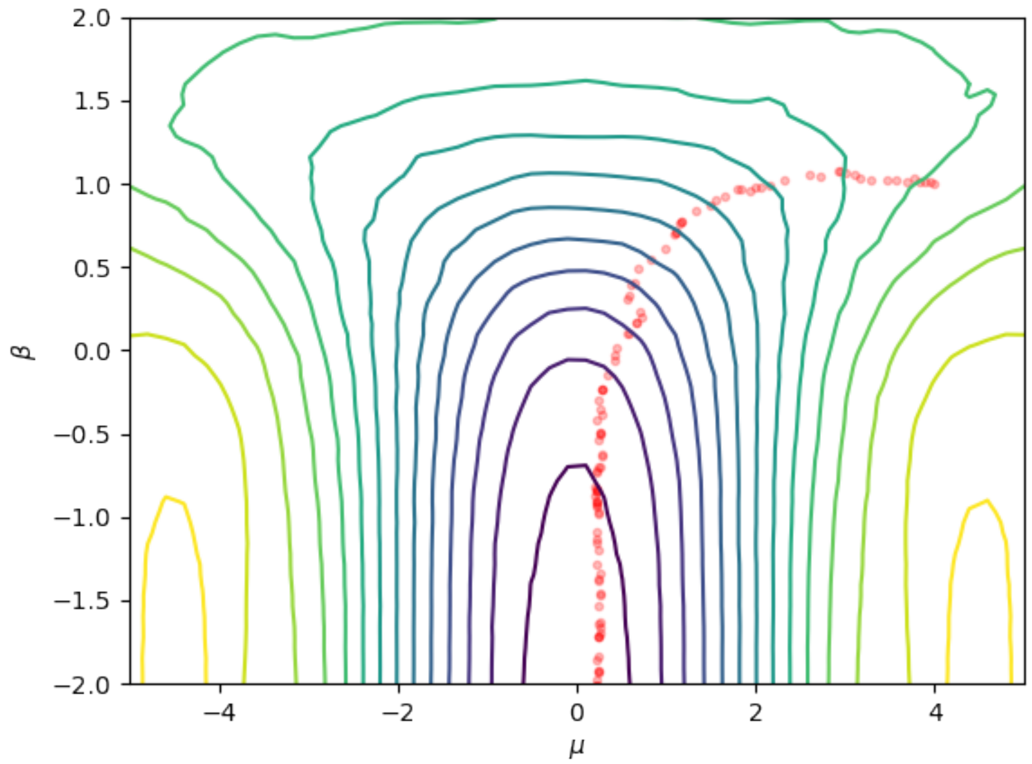
VARIATIONAL OPTIMIZATION

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

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



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



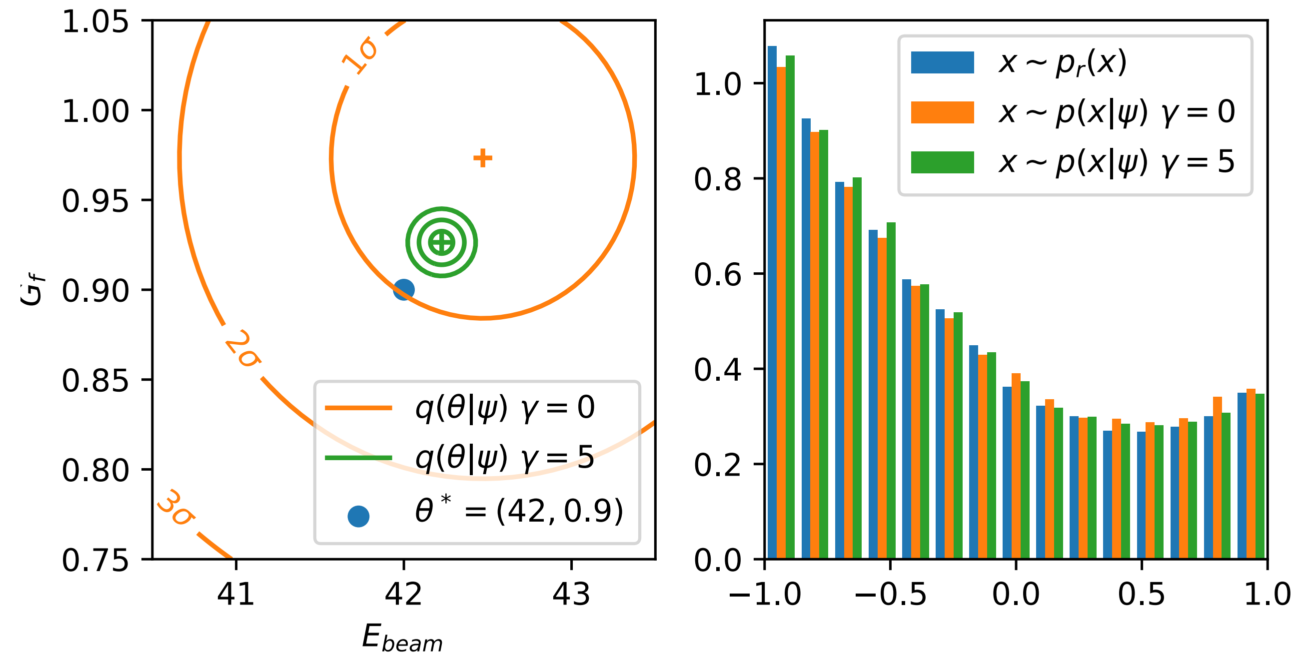
$q(\boldsymbol{\theta}|\boldsymbol{\psi} = (\mu, \beta)) = \mathcal{N}(\mu, e^{\beta})$

Like a GAN, but generative model is non-differentiable and the parameters of simulator have meaning

- Replace the generative network with a non-differentiable forward simulator $g(\mathbf{z};\boldsymbol{\theta})$.
- With VO, optimize upper bounds of the adversarial objectives:

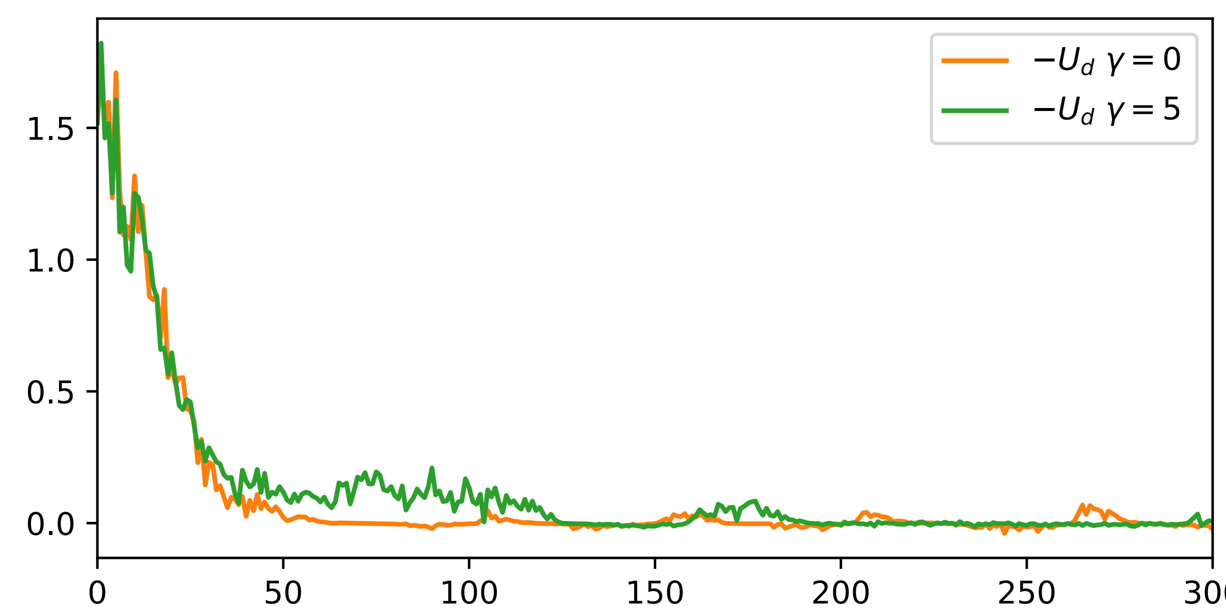
$$U_d = \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\boldsymbol{\psi})}[\mathcal{L}_d] \quad (1)$$
$$U_g = \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\boldsymbol{\psi})}[\mathcal{L}_g] \quad (2)$$

respectively over $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$.



Effectively sampling from marginal model

$$\mathbf{x} \sim q(\mathbf{x}|\boldsymbol{\psi}) \equiv \boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\boldsymbol{\psi}), \mathbf{z} \sim p(\mathbf{z}|\boldsymbol{\theta}), \mathbf{x} = g(\mathbf{z};\boldsymbol{\theta})$$



We use Wasserstein distance, as in WGAN

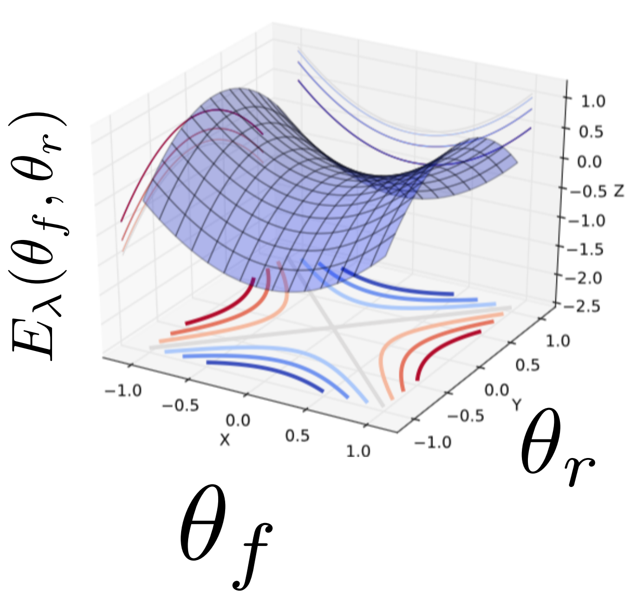
G. Louppe & K.C. arXiv:1707.07113

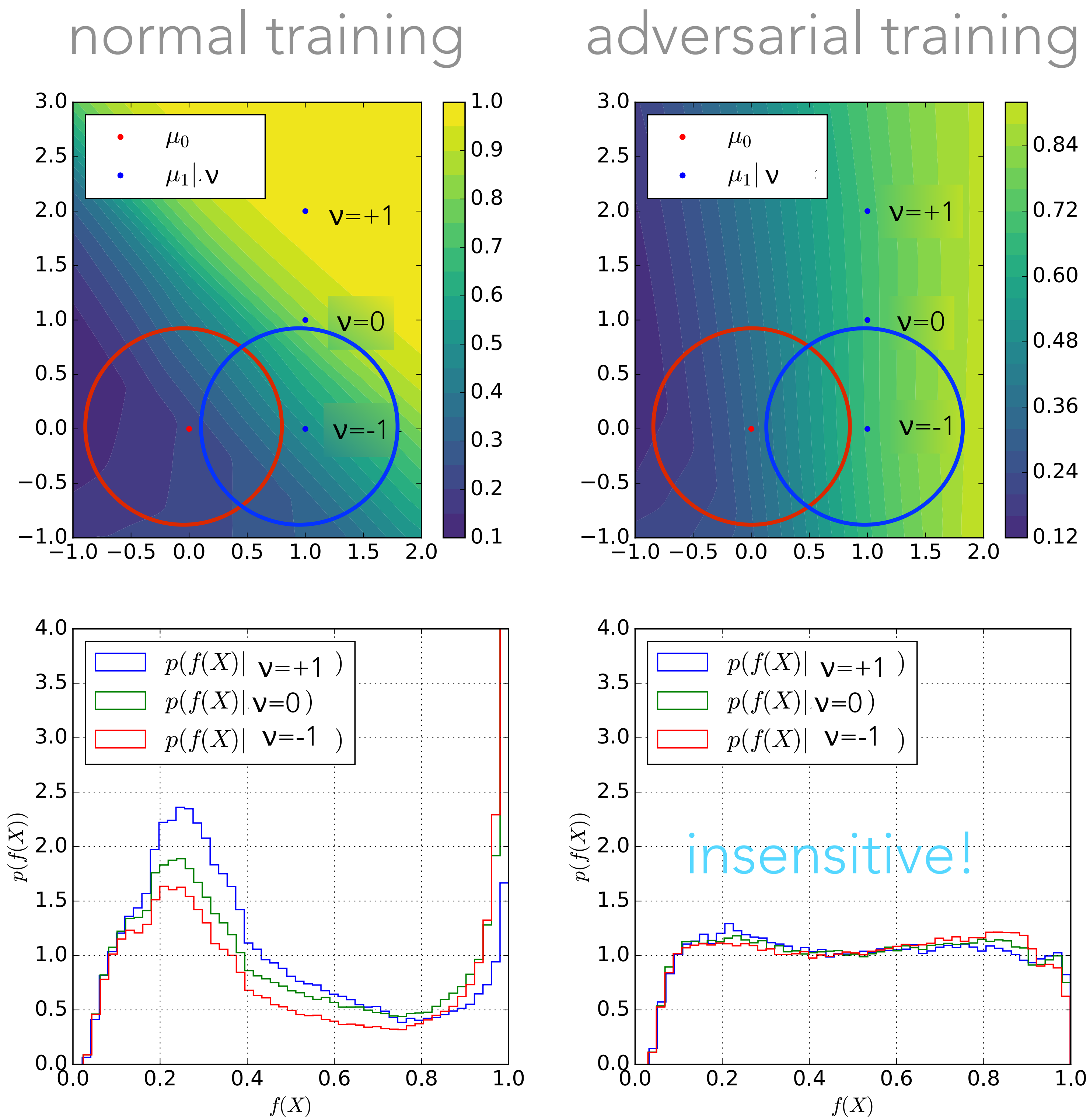
Typically classifier **f(x)** trained to minimize loss $\mathbf{L_f}$.

- 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:

$$\hat{\theta}_f, \hat{\theta}_r = \arg \min_{\theta_f} \max_{\theta_r} E(\theta_f, \theta_r).$$

$$E_\lambda(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r)$$





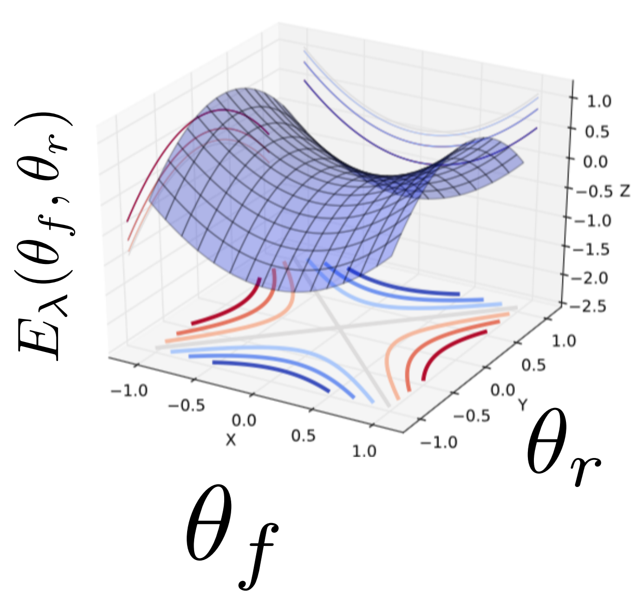
G. Louppe, M. Kagan, K. Cranmer, arXiv:1611.01046

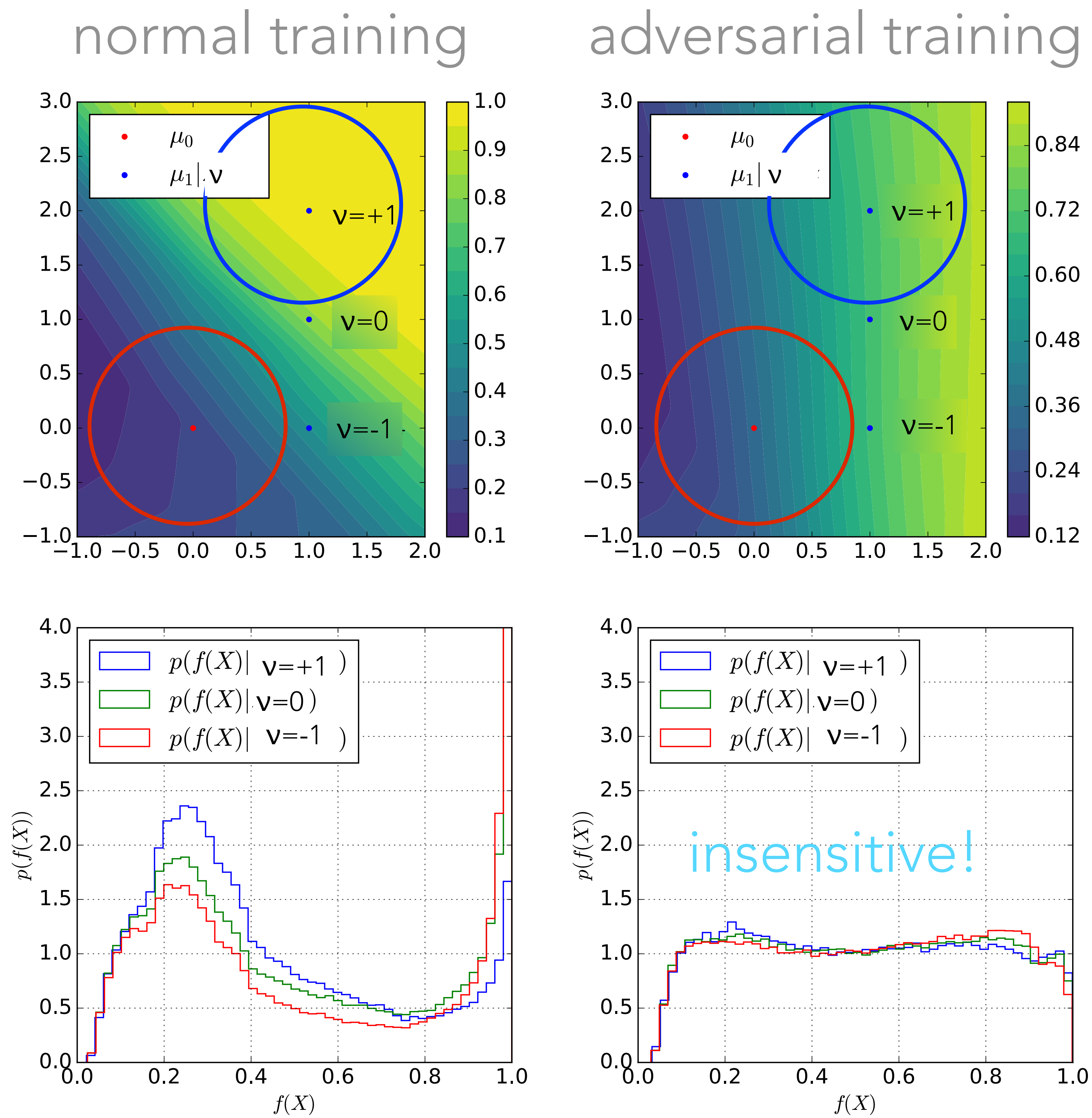
Typically classifier **f(x)** trained to minimize loss **L_f**.

- 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:

$$\hat{\theta}_f, \hat{\theta}_r = \arg \min_{\theta_f} \max_{\theta_r} E(\theta_f, \theta_r).$$

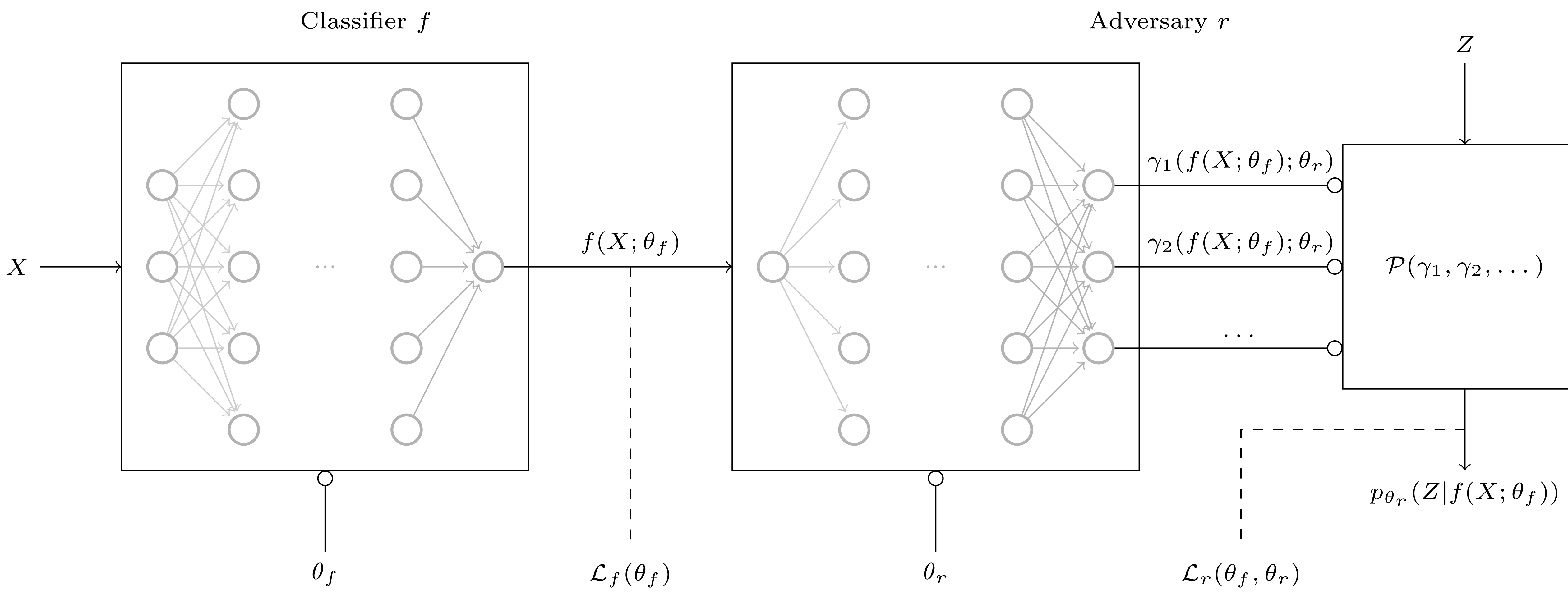
$$E_\lambda(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r)$$





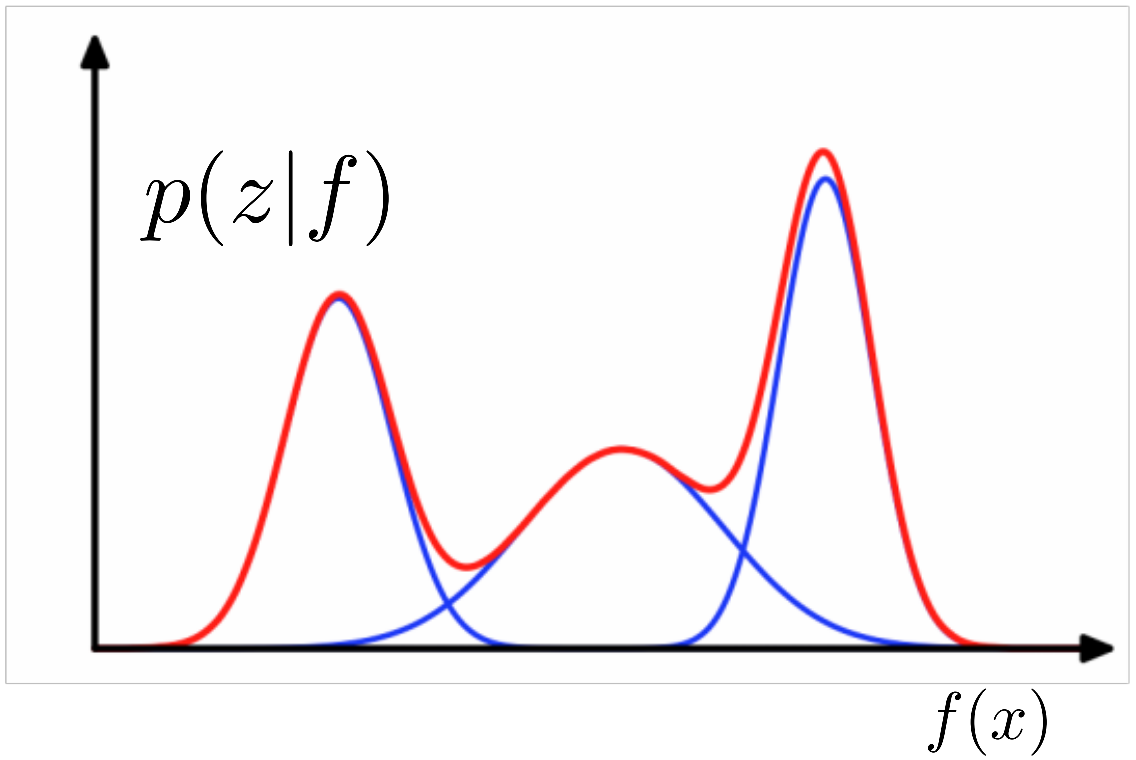
G. Louppe, M. Kagan, K. Cranmer, arXiv:1611.01046

THE ADVERSARIAL MODEL



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, …



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

Technique allows us to tune **λ**, the tradeoff between classification power and robustness to systematic uncertainty

An example:

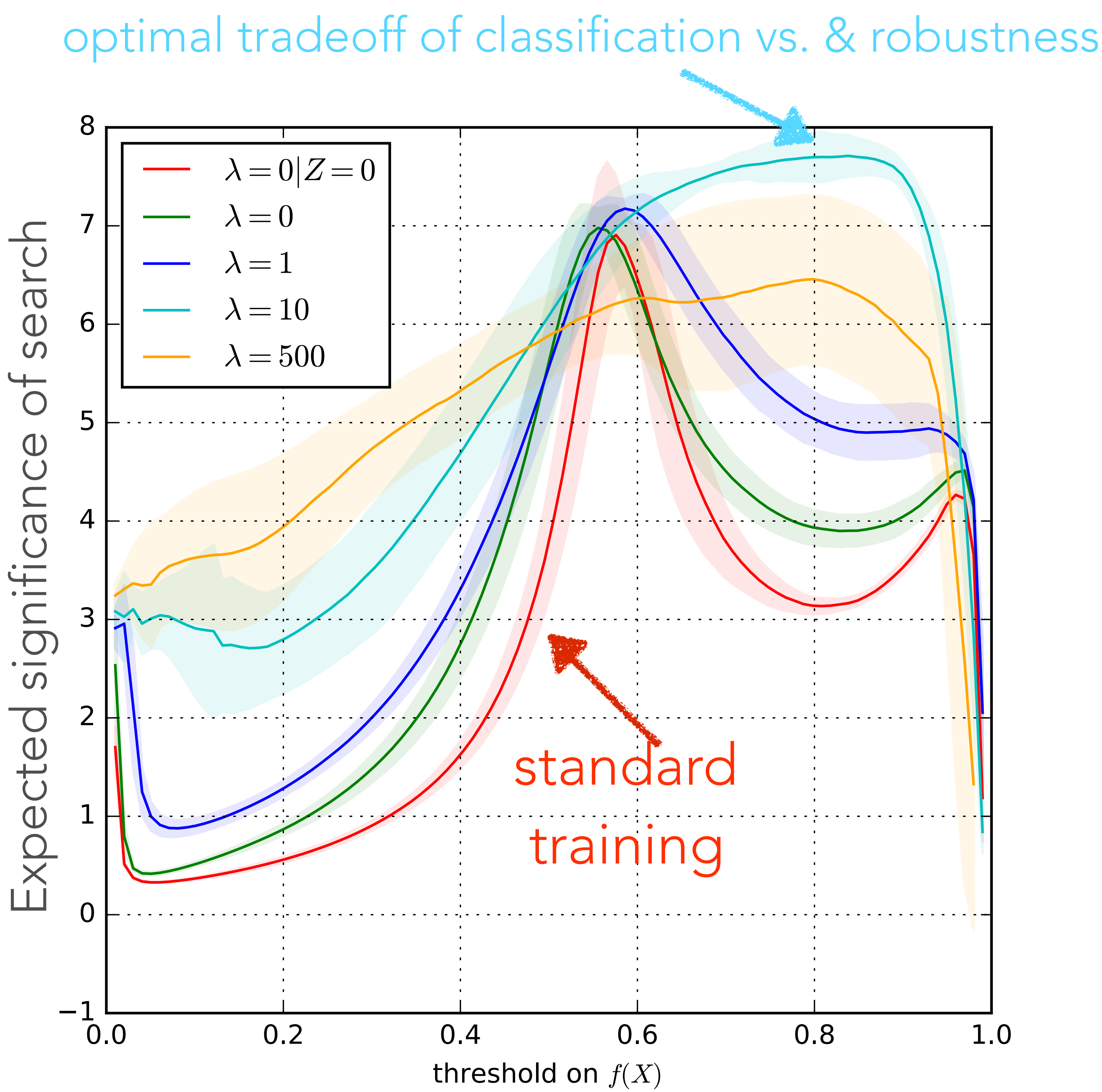
background: 1000 QCD jets

signal: 100 boosted W's

Train W vs. QCD classifier

Pileup as source of uncertainty

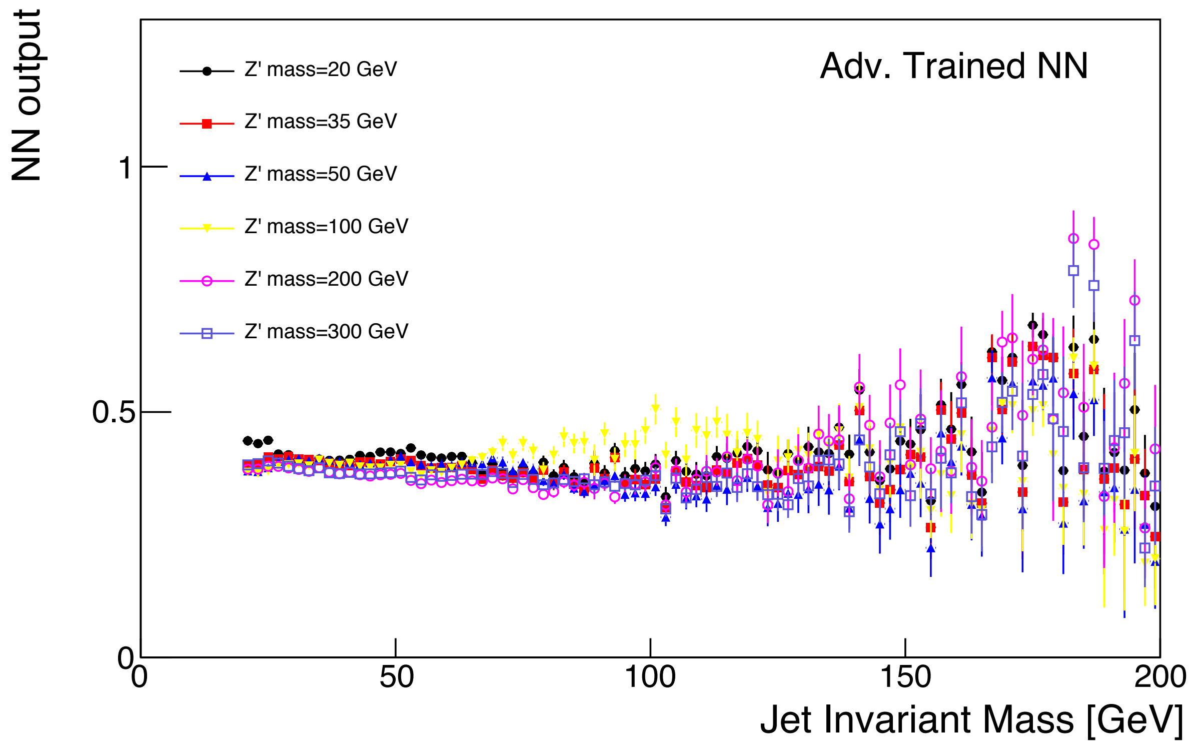
Simple cut-and-count analysis with background uncertainty.

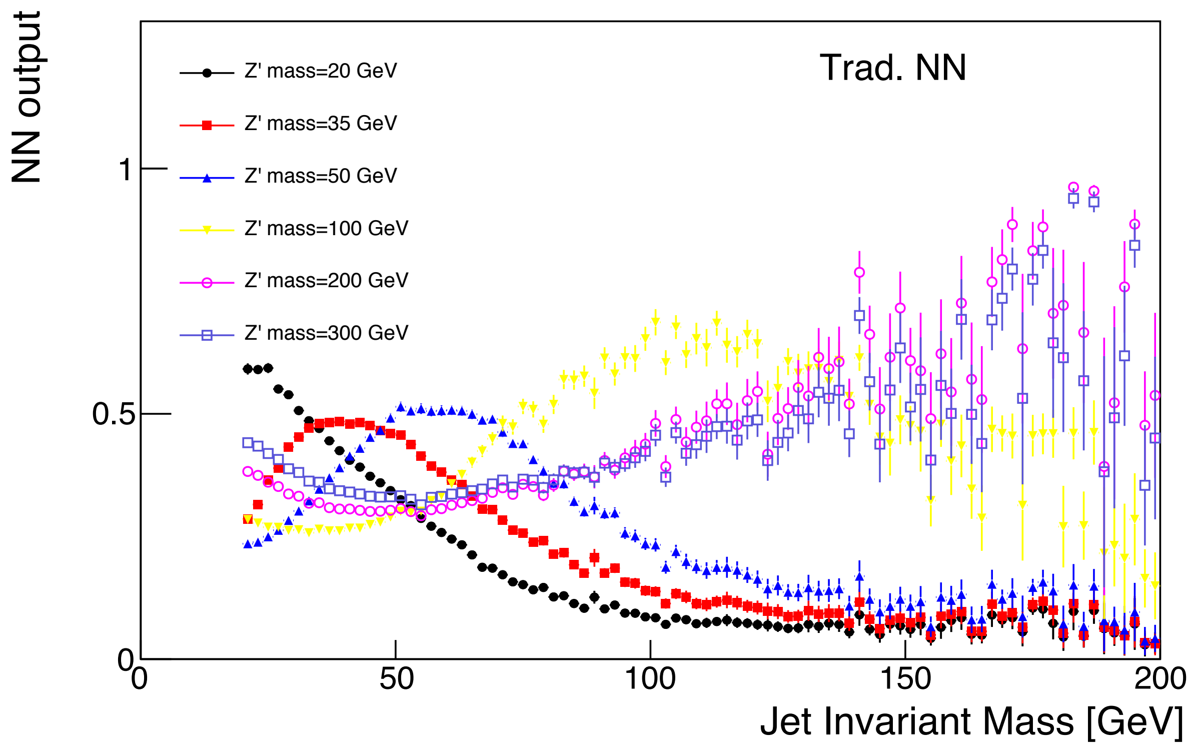


K.C, J. Pavez, and G. Louppe, arXiv:1506.02169
P. Baldi, K.C, T. Faucett, P. Sadowski, D. Whiteson arXiv:1601.07913
G. Louppe, M. Kagan, K.C, arXiv:1611.01046
Shimmin, et. al. arXiv:1703.03507

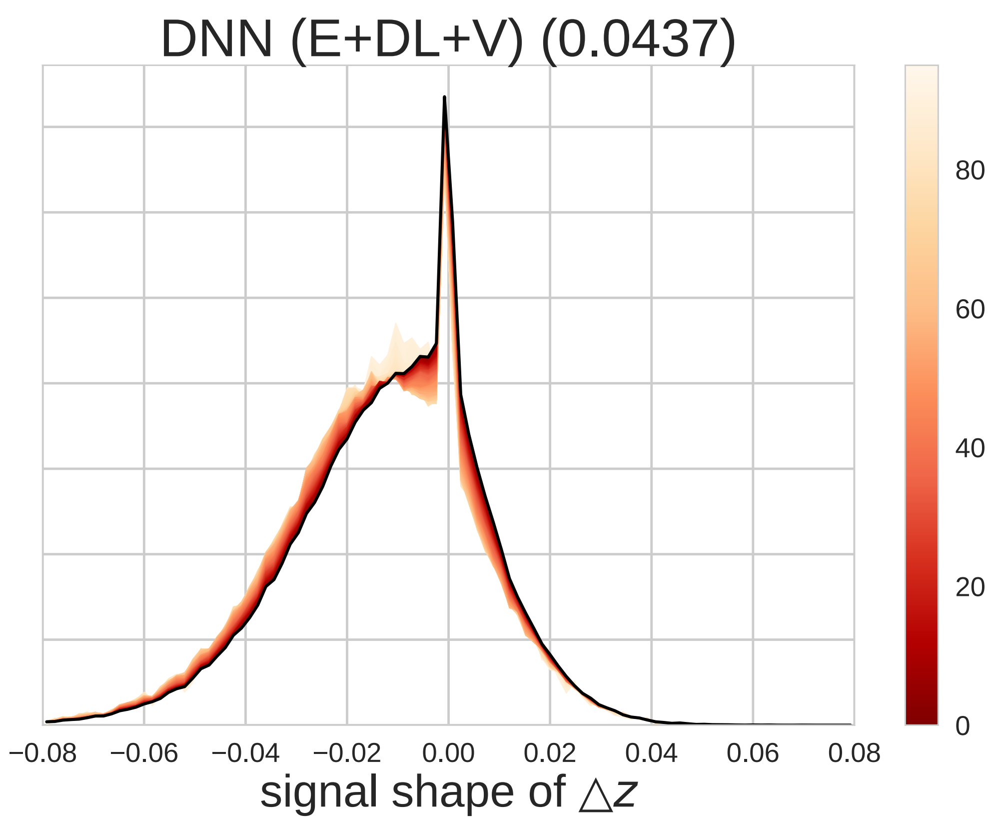
Adversarial approach of "Learning to Pivot" can also be used to train a classifier that is "decorrelated" to some other variable.

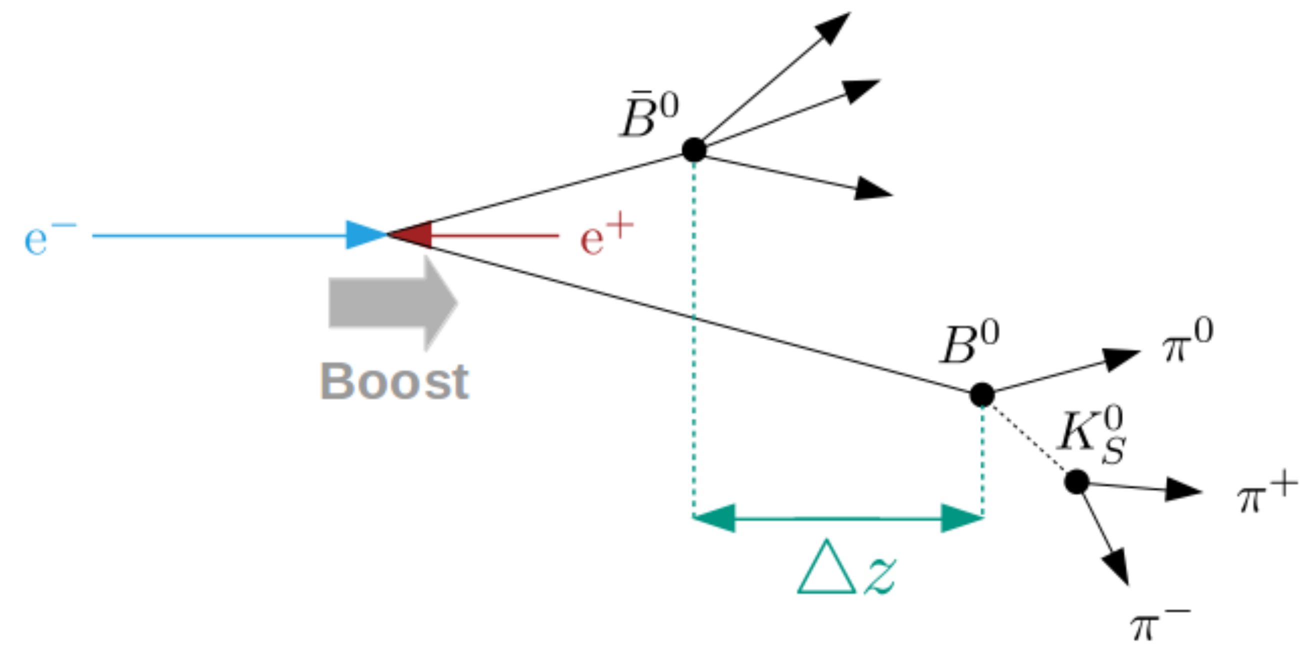
- want jet taggers that are decorrelated with jet invariant mass
- so that analysis can still search for a bump using jet invariant mass
- avoids sculpting background

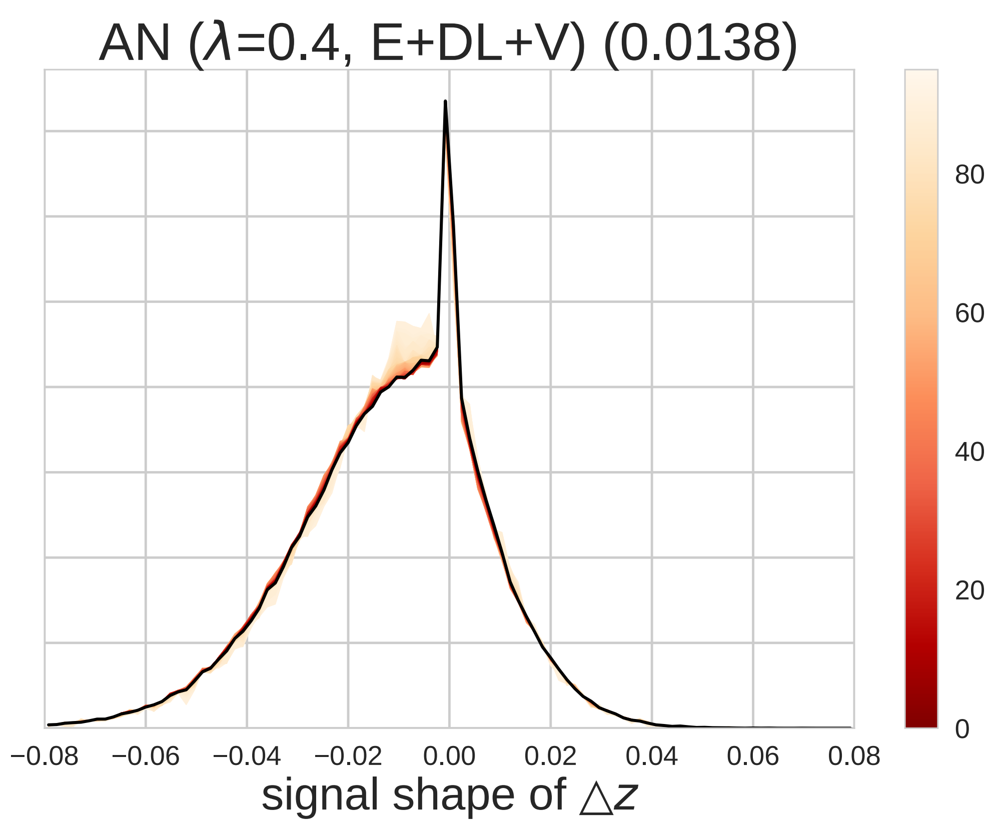


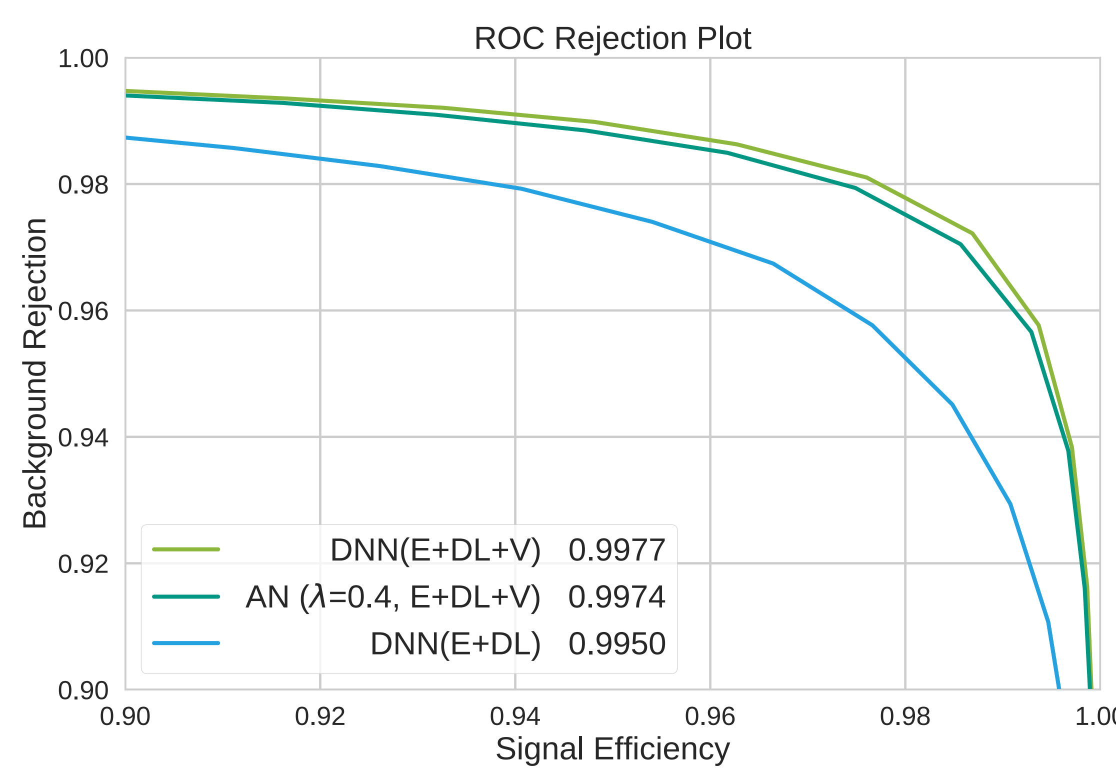


DECORRELATION IN BELLE II

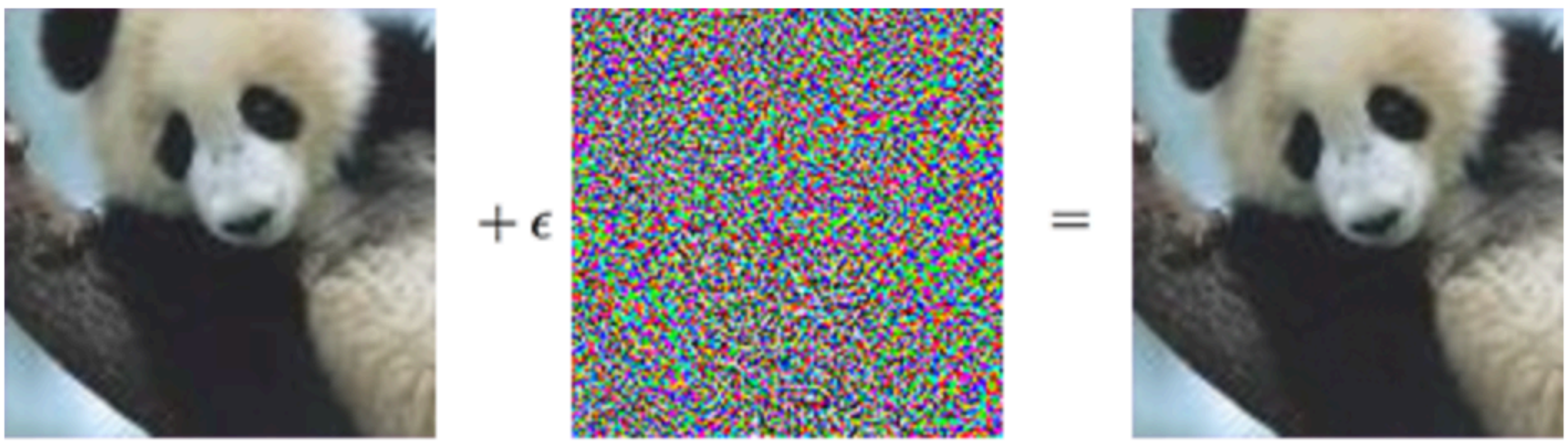








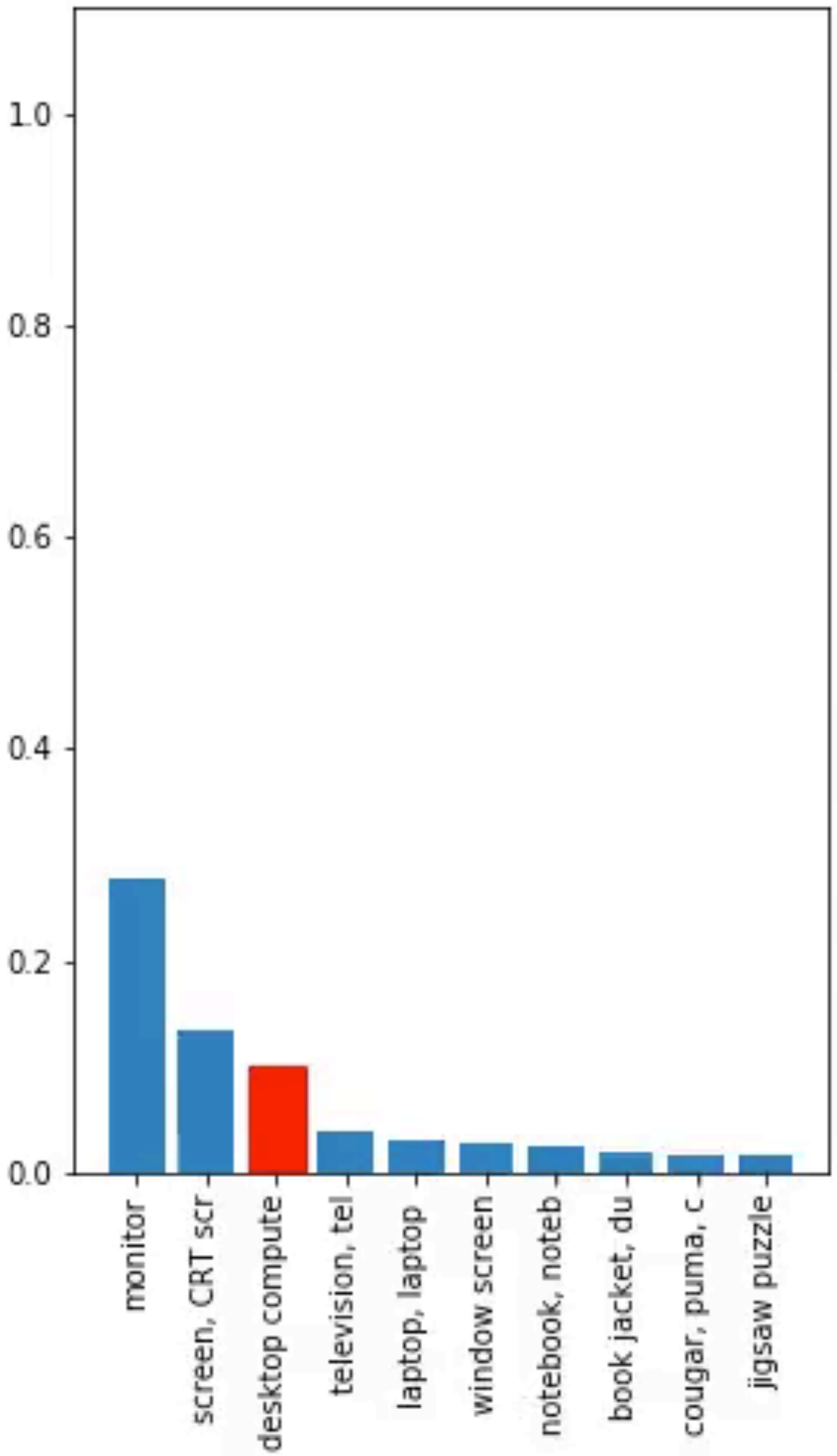
ADVERSARIAL EXAMPLES



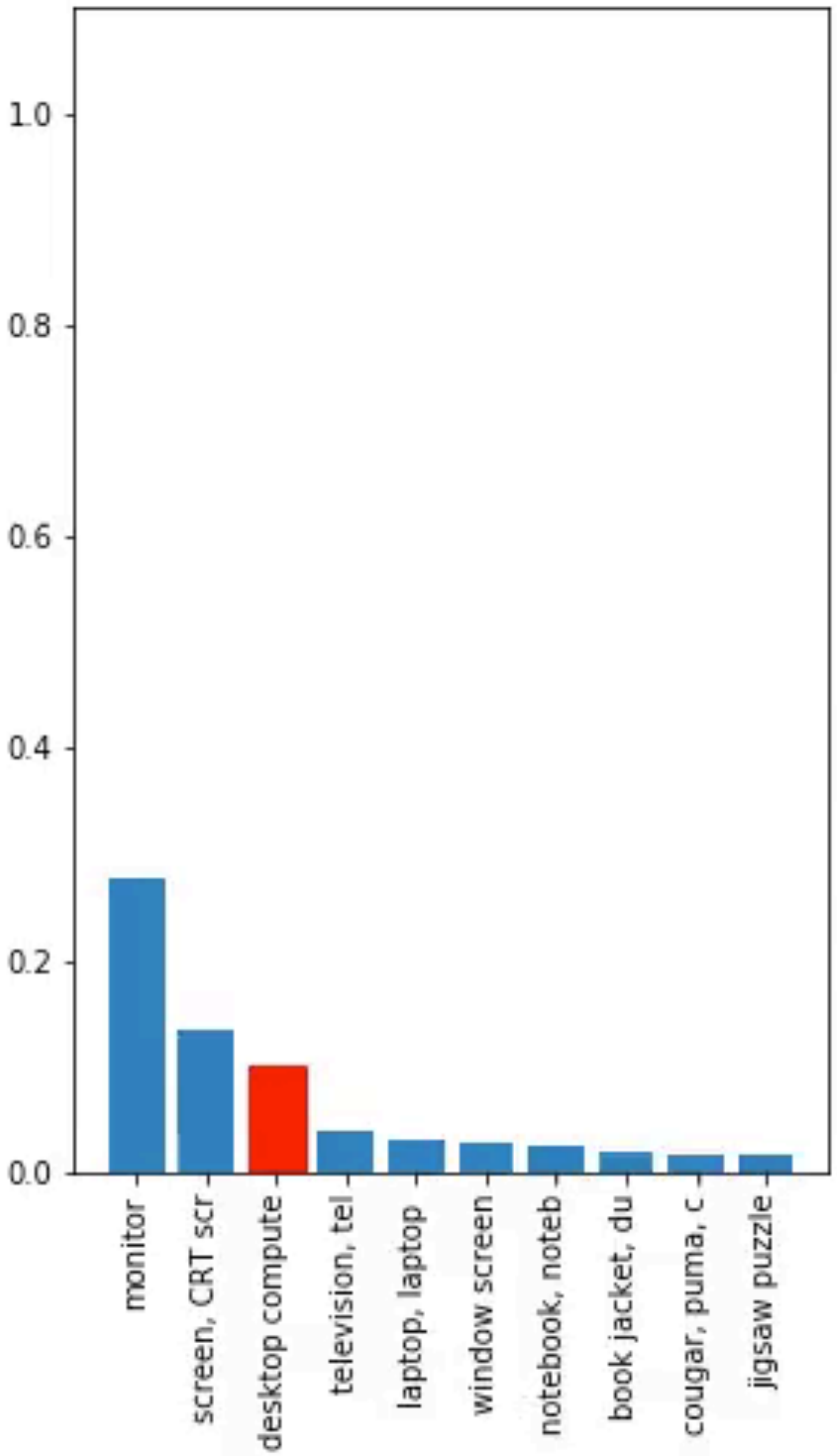
"panda"
57.7% confidence

"gibbon"
99.3% confidence

ADVERSARIAL EXAMPLES



ADVERSARIAL EXAMPLES



LECTURE 4

Extending to include systematics:

- statistical modeling with nuisance parameters
    - RooFit ↔ TensorFlow, automatic differentiation
- Profile Likelihood Ratio & concept of a “pivot”

Parametrized learning

- for classification
- high dimensional reweighting

Other Likelihood Free techniques

- ABC & probabilistic programming

Gaussian Processes

- physics-aware kernels

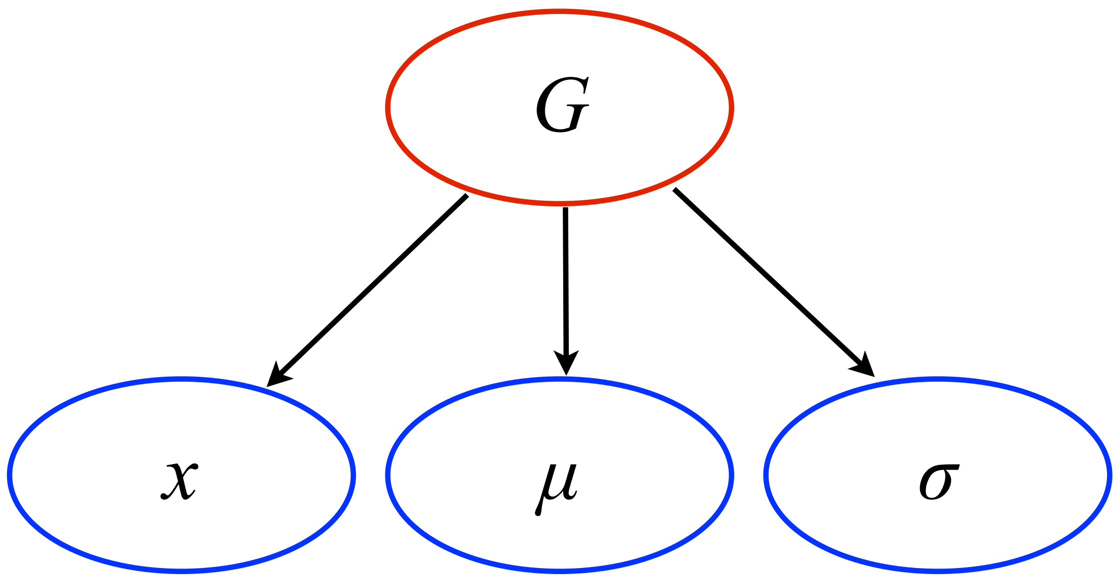
QCD-aware neural networks

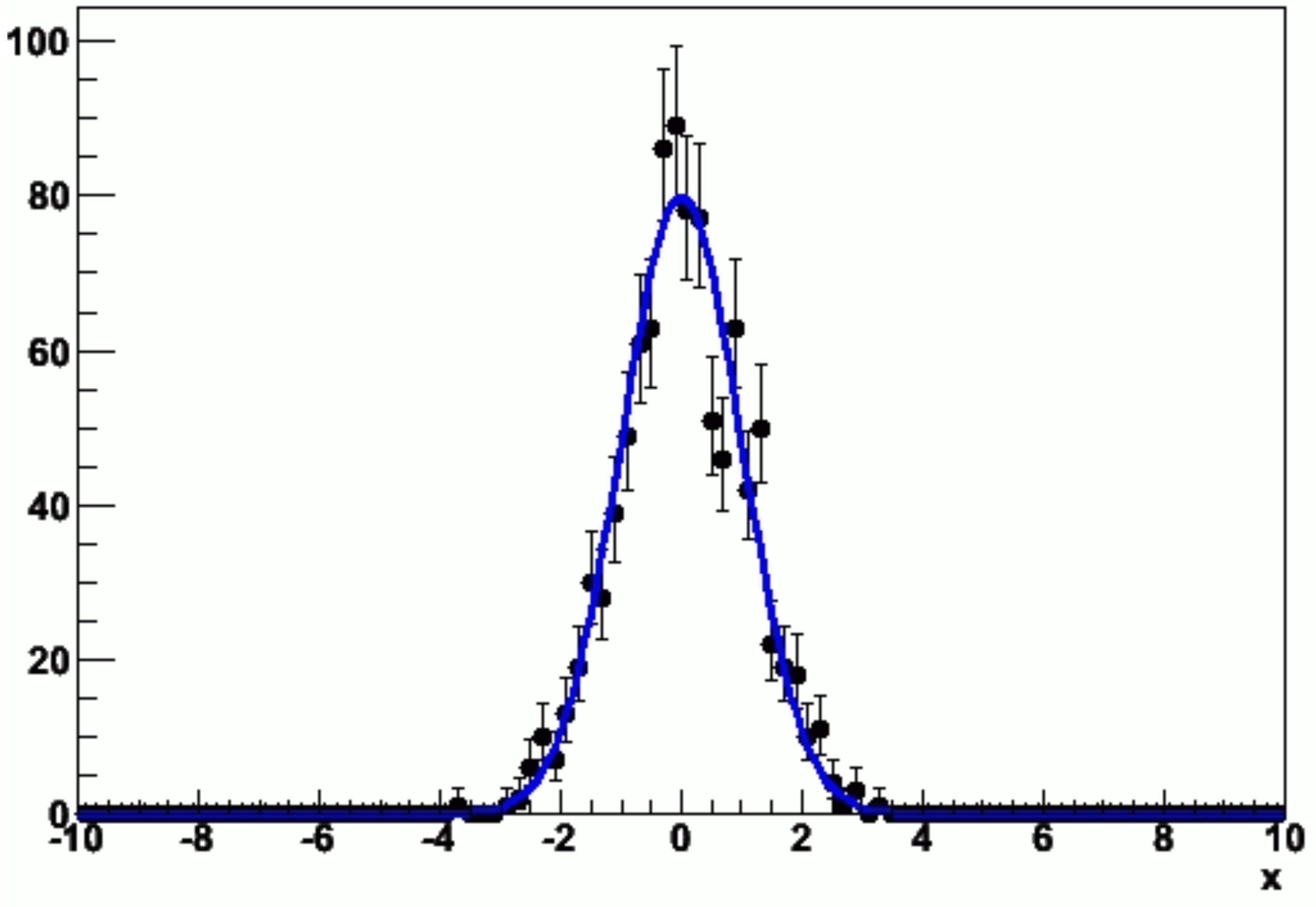
Building a Statistical Model
Systematics & Nuisance Parameters

VISUALIZING PROBABILITY MODELS

I will represent PDFs graphically as below (directed acyclic graph)

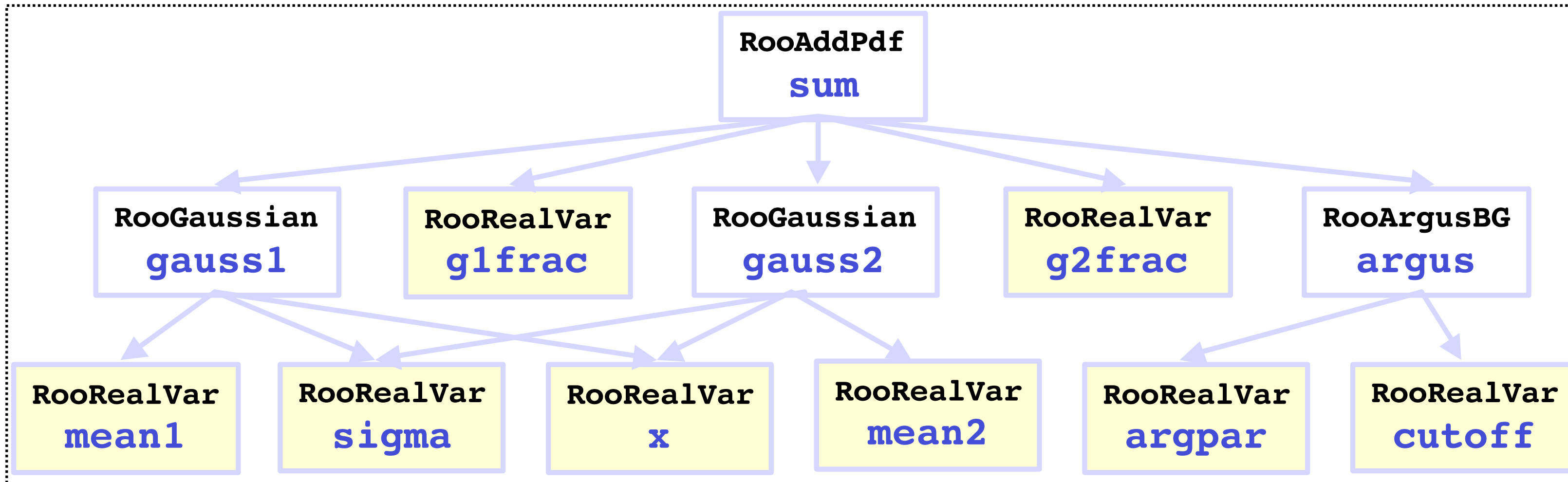
- eg. a Gaussian $G(x|\mu,\sigma)$ is parametrized by (μ,σ)
- every node is a real-valued function of the nodes below



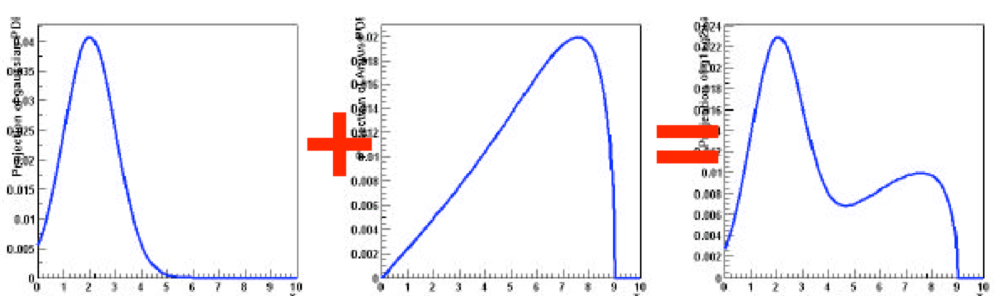


RooFit: A data modeling toolkit

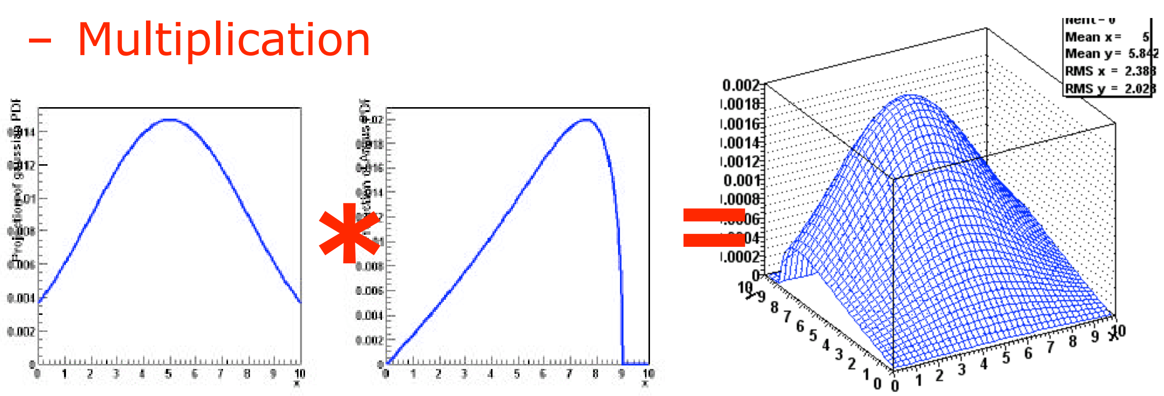
RooFit is a major tool developed at BaBar for data modeling.
RooStats provides higher-level statistical tools based on these PDFs.



– Addition

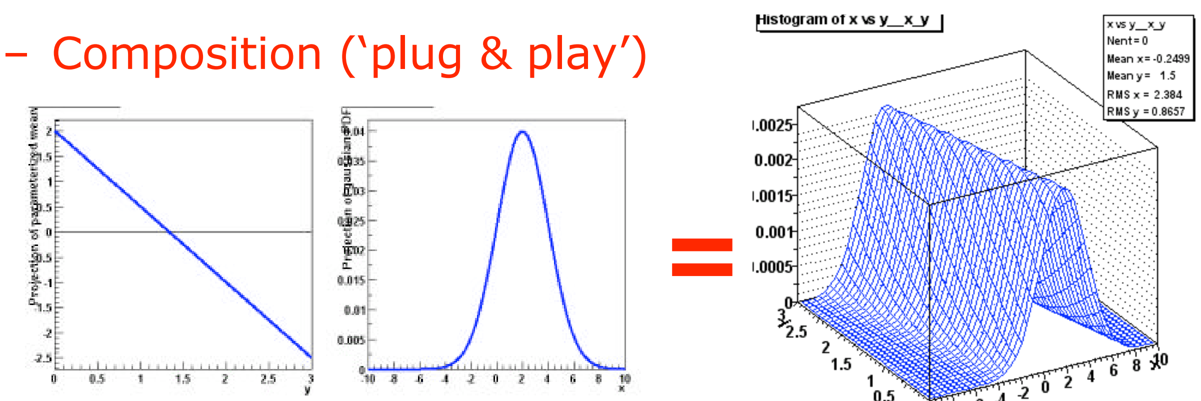


– Multiplication



Wouter Verkerke,

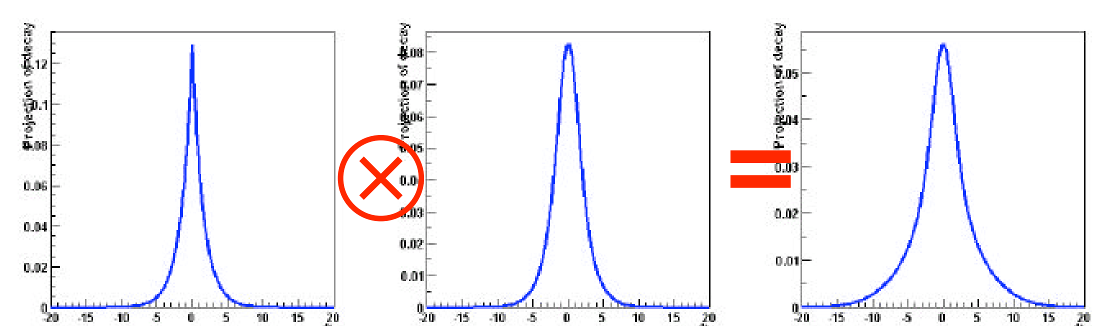
– Composition ('plug & play')



$m(y;a_0,a_1)$ $g(x;m,s)$ $g(x,y;a0,a1,s)$

Possible in *any* PDF
No explicit support in PDF code needed

– Convolution



Wouter Verkerke, UCSB

Marked Poisson Process

Channel: a subset of the data defined by some selection requirements.

- eg. all events with 4 electrons with energy > 10 GeV
- n: number of events observed in the channel
- ν: number of events expected in the channel

Discriminating variable: a property of those events that can be measured and which helps discriminate the signal from background

- eg. the invariant mass of two particles
- $f(x)$: the p.d.f. of the discriminating variable x

$$\mathcal{D} = \{x_1, \ldots, x_n\}$$

Marked Poisson Process / Extended Likelihood:

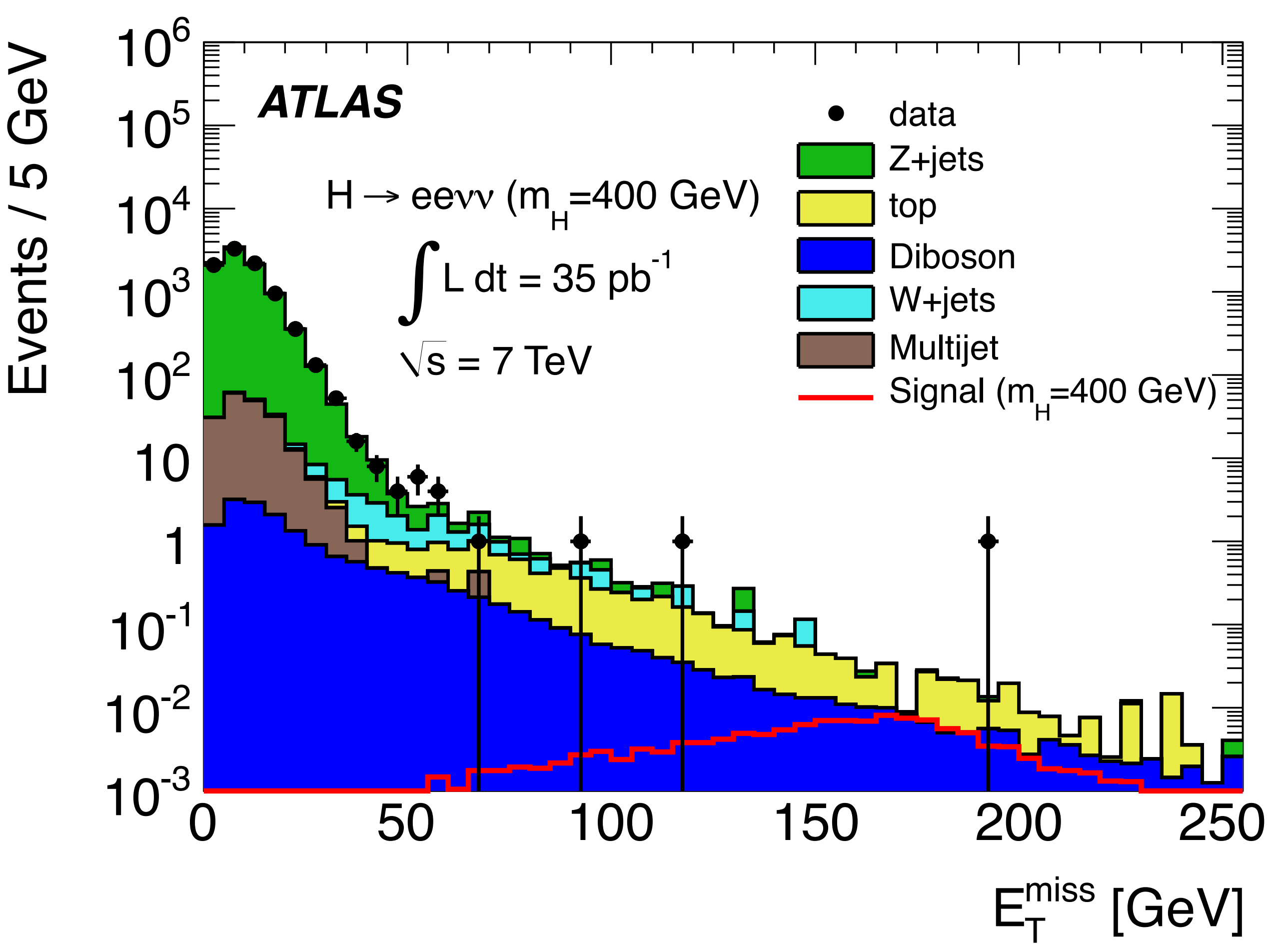
$$\mathbf{f}(\mathcal{D}|\nu) = \text{Pois}(n|\nu) \prod_{e=1}^{n} f(x_e)$$

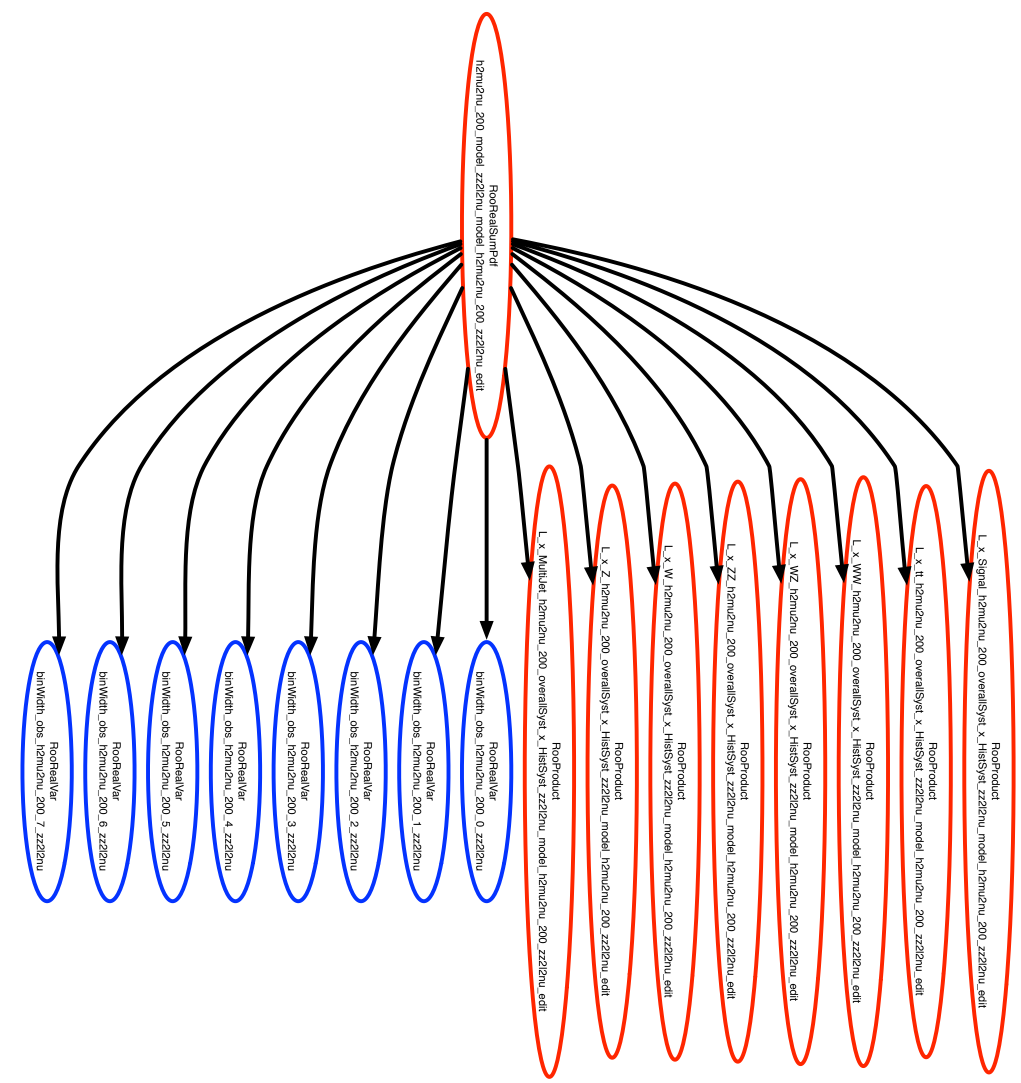
MIXTURE MODEL

Sample: a sample of simulated events corresponding to particular type interaction that populates the channel.

- statisticians call this a mixture model

$$f(x) = \frac{1}{\nu_{\rm tot}} \sum_{s \in \text{samples}} \nu_s f_s(x) \; , \qquad \nu_{\rm tot} = \sum_{s \in \text{samples}} \nu_s$$





PARAMETRIZING THE MODEL $\boldsymbol{\alpha} = (\mu, \boldsymbol{\theta})$

Parameters of interest (*μ*): parameters of the theory that modify the rates and shapes of the distributions, eg.

- the mass of a hypothesized particle
- the "signal strength" μ=0 no signal, μ=1 predicted signal rate

Nuisance parameters (*θ* or α_p): associated to uncertainty in:

- response of the detector (calibration)
- phenomenological model of interaction in non-perturbative regime

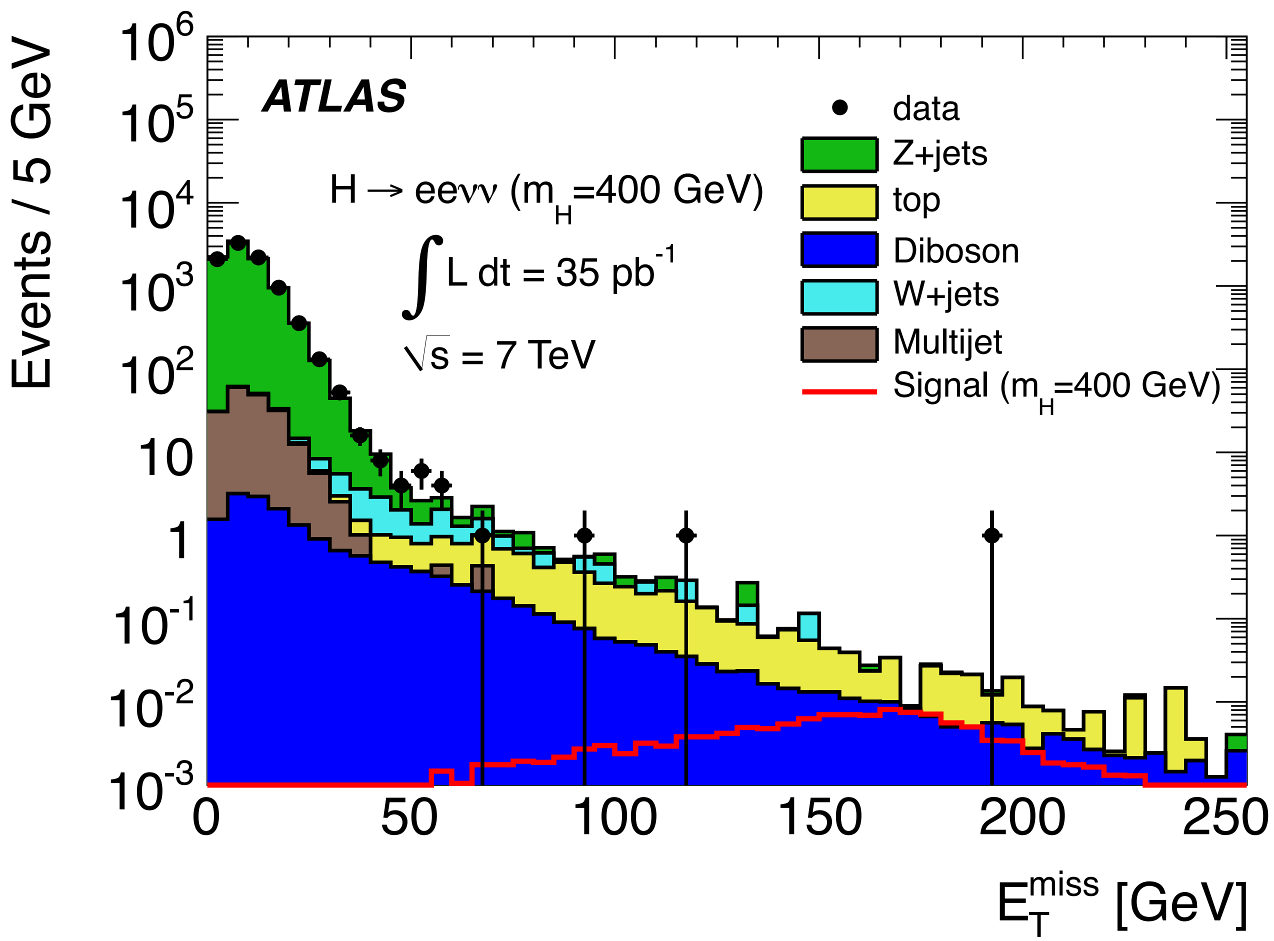
Lead to a parametrized model: $\nu \to \nu(\boldsymbol{\alpha}), f(x) \to f(x|\boldsymbol{\alpha})$

$$\mathbf{f}(\mathcal{D}|\boldsymbol{\alpha}) = \mathrm{Pois}(n|\nu(\boldsymbol{\alpha})) \prod_{e=1}^{n} f(x_e|\boldsymbol{\alpha})$$

INCORPORATING SYSTEMATIC EFFECTS

Tabulate effect of individual variations of sources of systematic uncertainty

- typically one at a time evaluated at nominal and "± 1 σ"
- use some form of interpolation to parametrize p^{th} variation in terms of **nuisance parameter** α_p



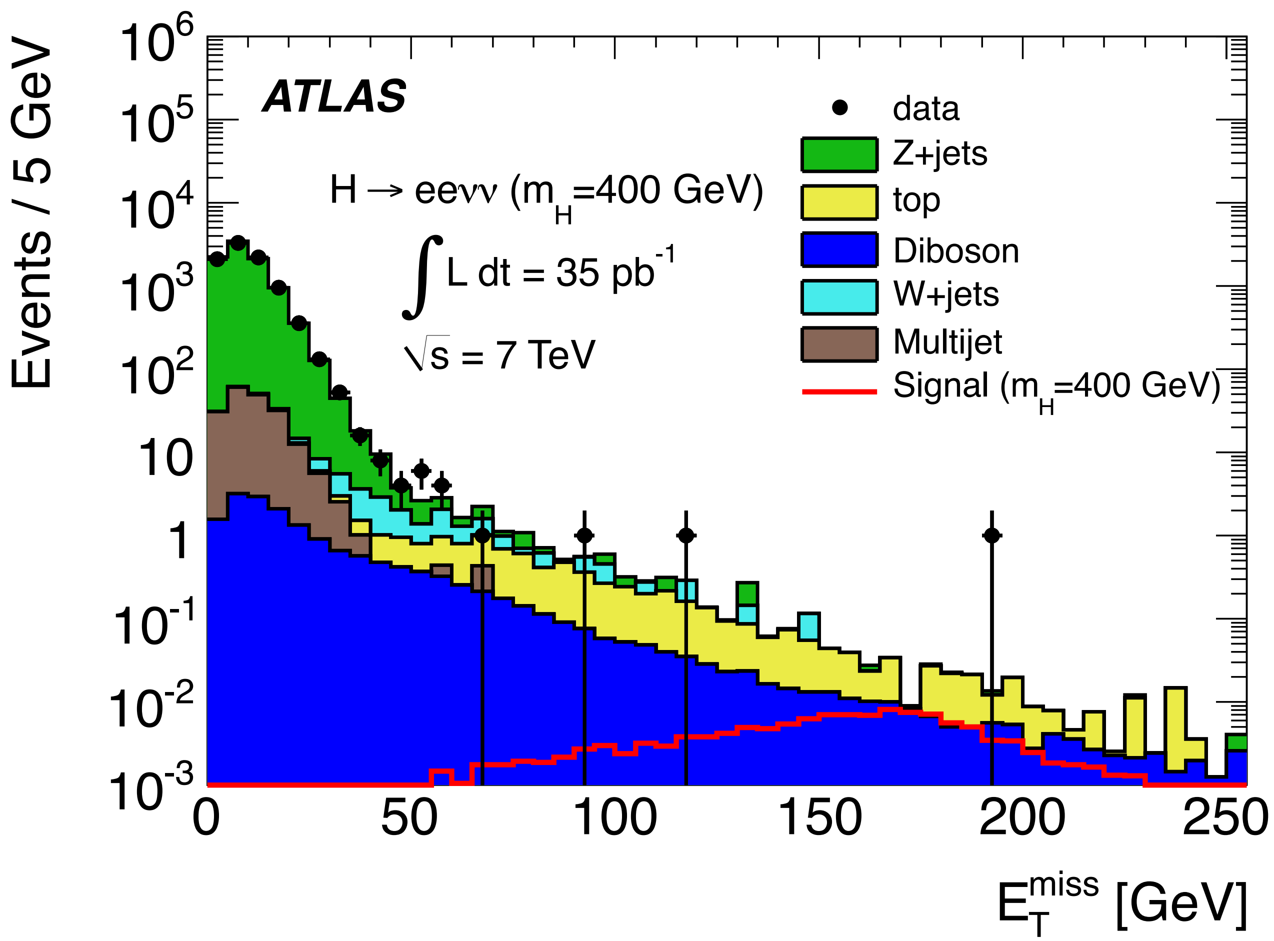
	Z+jets	top	Diboson	...
syst 1				
syst 2				
...				

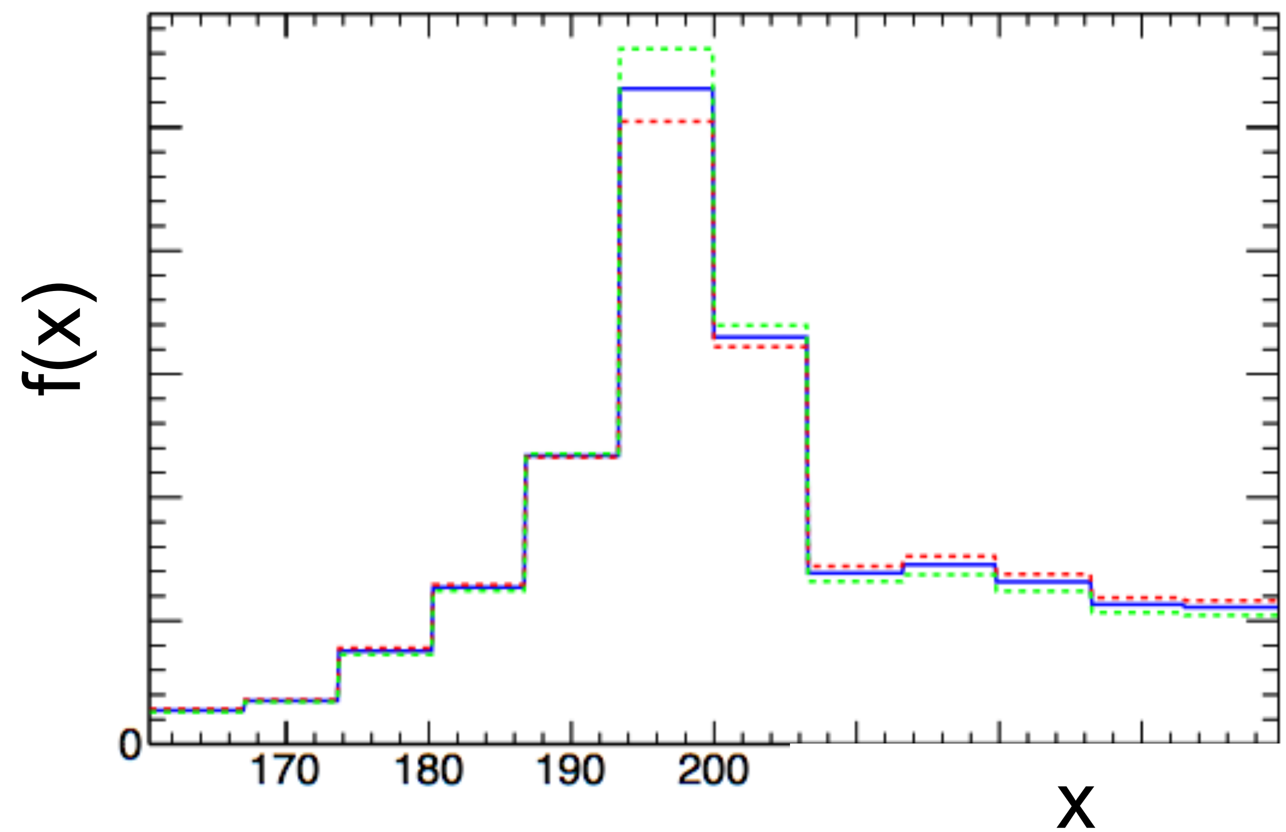
$$\mathbf{f}(\mathcal{D}|\boldsymbol{\alpha}) = \mathrm{Pois}(n|\nu(\boldsymbol{\alpha})) \prod_{e=1}^{n} f(x_e|\boldsymbol{\alpha})$$

INCORPORATING SYSTEMATIC EFFECTS

Tabulate effect of individual variations of sources of systematic uncertainty

- typically one at a time evaluated at nominal and "± 1 σ"
- use some form of interpolation to parametrize p^{th} variation in terms of **nuisance parameter** α_p



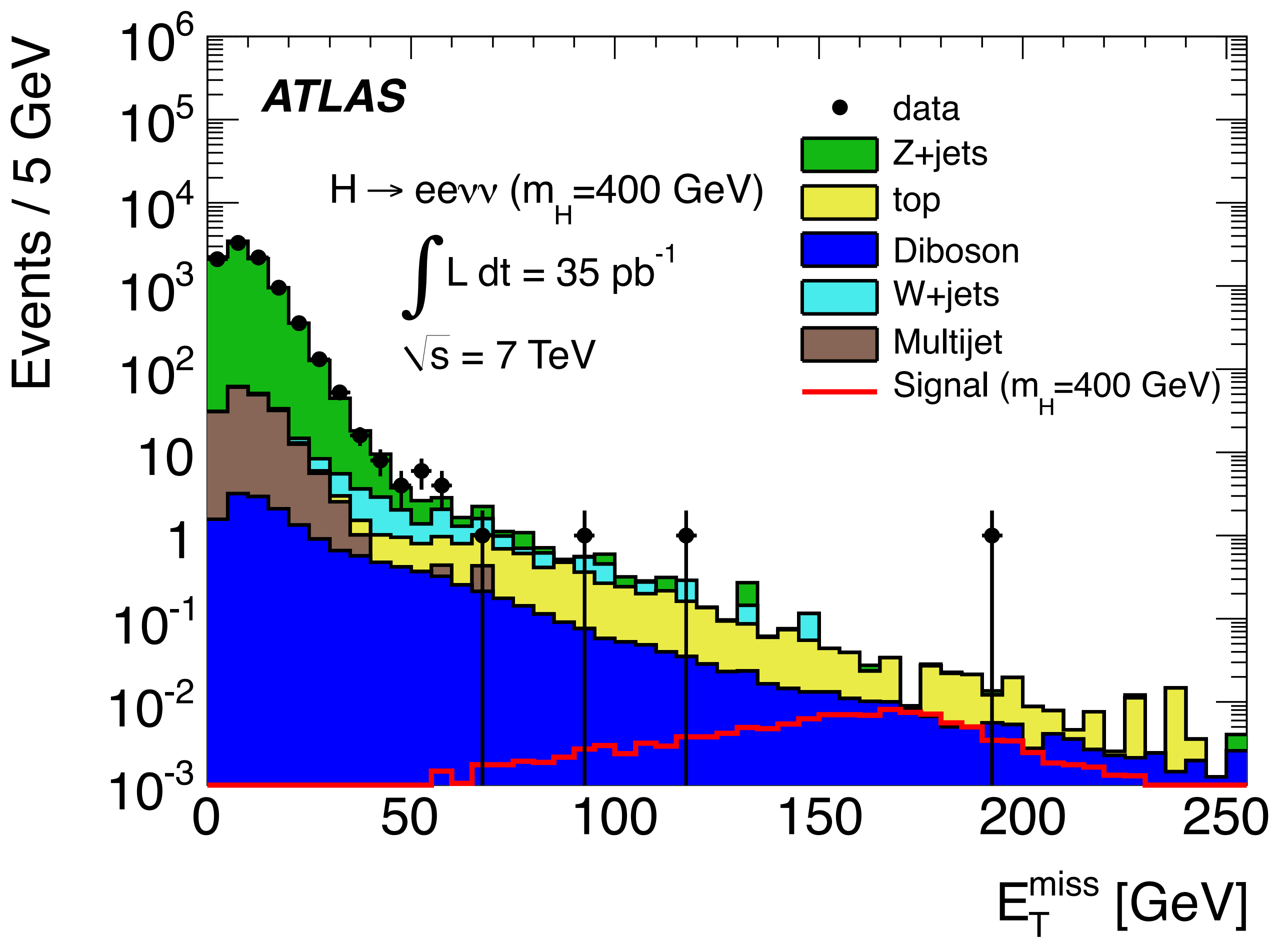


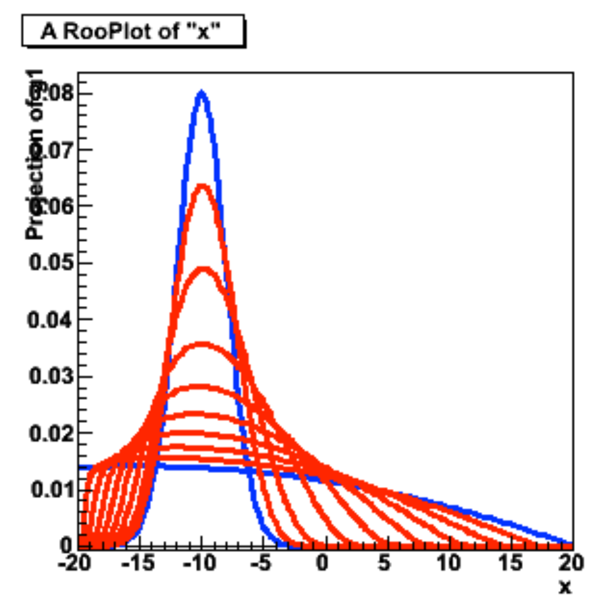
$$\mathbf{f}(\mathcal{D}|\boldsymbol{\alpha}) = \text{Pois}(n|\nu(\boldsymbol{\alpha})) \prod_{e=1}^{n} f(x_e|\boldsymbol{\alpha})$$

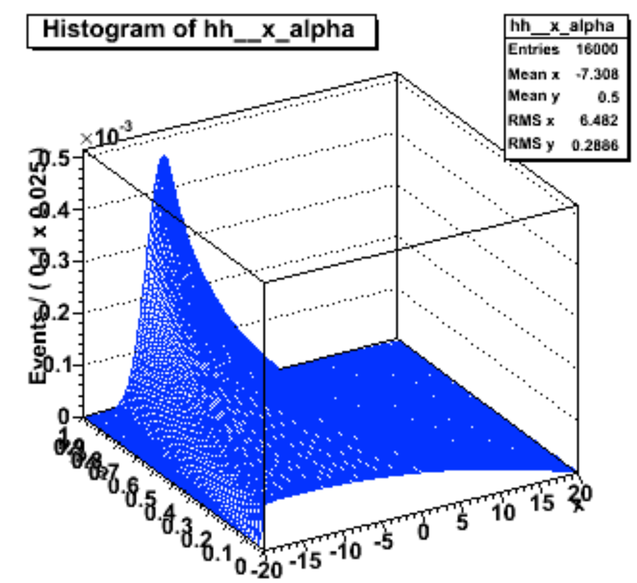
Incorporating Systematic Effects

Tabulate effect of individual variations of sources of systematic uncertainty

- typically one at a time evaluated at nominal and "± 1 σ"
- use some form of interpolation to parametrize p^{th} variation in terms of **nuisance parameter** α_p

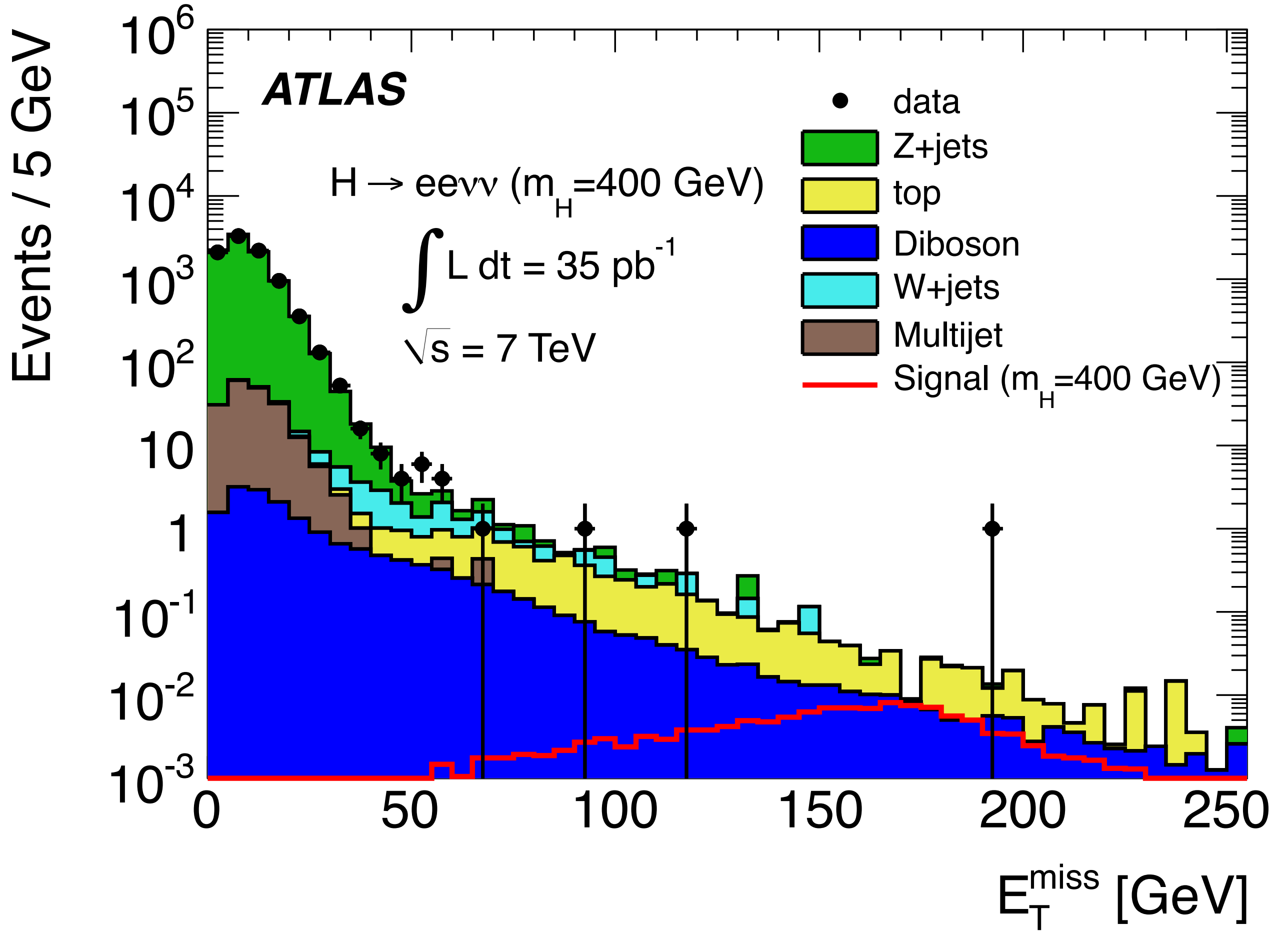


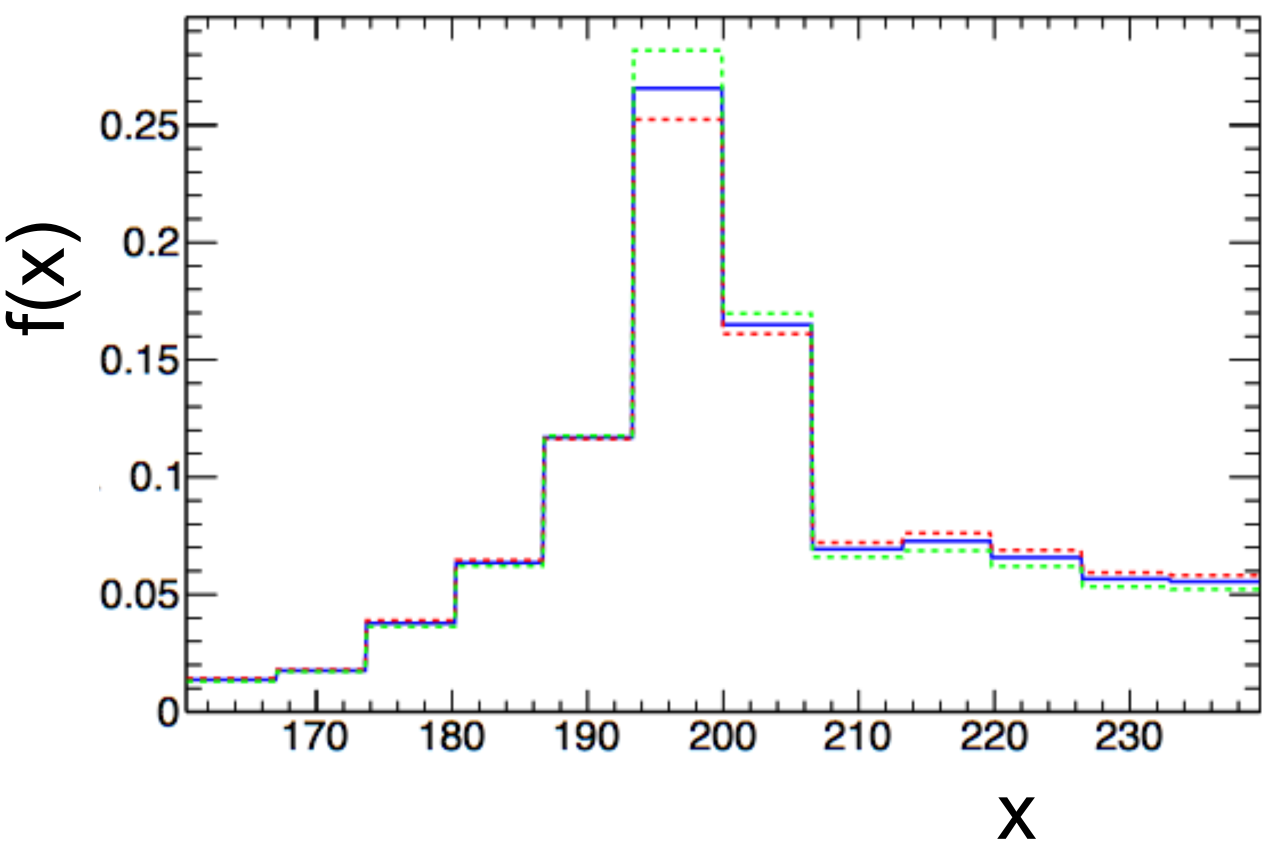


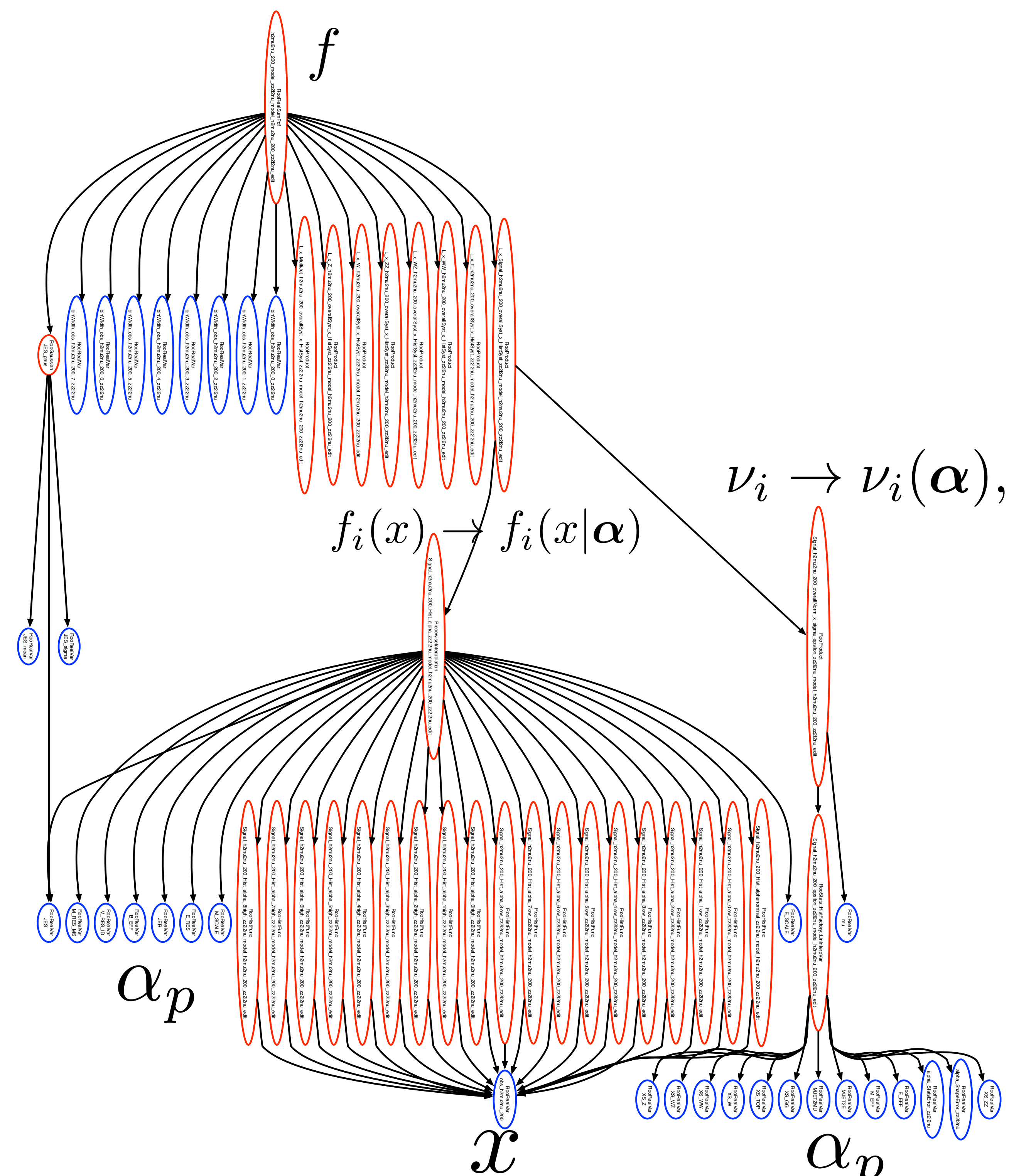


$$\mathbf{f}(\mathcal{D}|\boldsymbol{\alpha}) = \mathrm{Pois}(n|\nu(\boldsymbol{\alpha})) \prod_{e=1}^{n} f(x_e|\boldsymbol{\alpha})$$

VISUALIZING THE MODEL FOR ONE CHANNEL

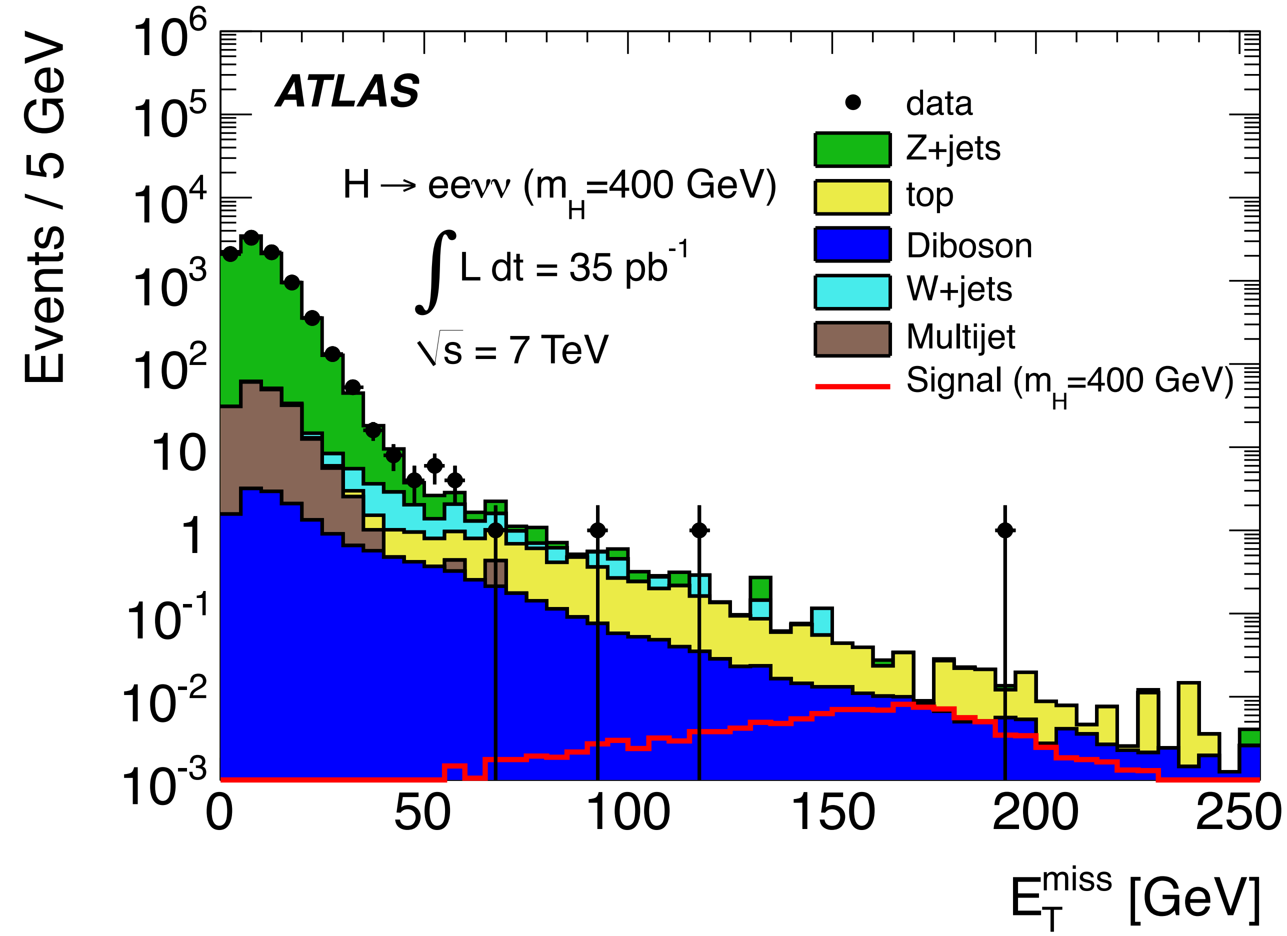


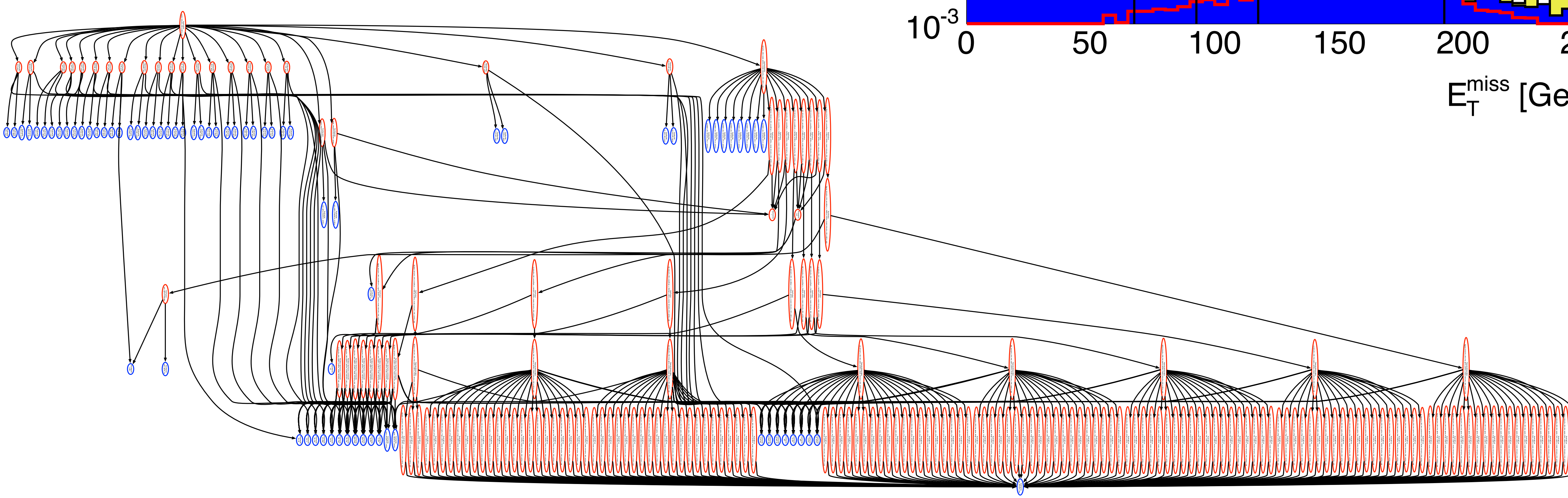




VISUALIZING THE MODEL FOR ONE CHANNEL

After parametrizing each component of the mixture model, the pdf for a single channel might look like this





Simultaneous Multi-Channel Model: Several disjoint regions of the data are modeled simultaneously. Identification of common parameters across many channels requires coordination between groups such that meaning of the parameters are really the same.

$$\mathbf{f}_{\rm sim}(\mathcal{D}_{\rm sim}|\boldsymbol{\alpha}) = \prod_{c\in\rm channels}\left[\mathrm{Pois}(n_c|\nu_c(\boldsymbol{\alpha}))\prod_{e=1}^{n_c} f_c(x_{ce}|\boldsymbol{\alpha})\right]$$

where $\mathcal{D}_{\rm sim} = \{\mathcal{D}_1, \ldots, \mathcal{D}_{c_{\rm max}}\}$

Control Regions: Some channels are not populated by signal processes, but are used to constrain the nuisance parameters

- attempt to describe systematics in a statistical language
- Prototypical Example: “on/off” problem with unknown ν_b

$$\mathbf{f}(n, m|\mu, \nu_b) = \underbrace{\mathrm{Pois}(n|\mu+\nu_b)}_{\text{signal region}} \cdot \underbrace{\mathrm{Pois}(m|\tau\nu_b)}_{\text{control region}}$$

Constraint terms

Often detailed statistical model for auxiliary measurements that measure certain nuisance parameters are not available.

- one typically has MLE for α_p, denoted a_p and standard error

Constraint Terms: are idealized pdfs for the MLE.

$$f_p(a_p|\alpha_p) \quad \text{for} \quad p \in \mathbb{S}$$

- common choices are Gaussian, Poisson, and log-normal
- New: careful to write constraint term a frequentist way
- Previously: $\pi(\alpha_p|a_p) = f_p(a_p|\alpha_p)\eta(\alpha_p)$ with uniform η

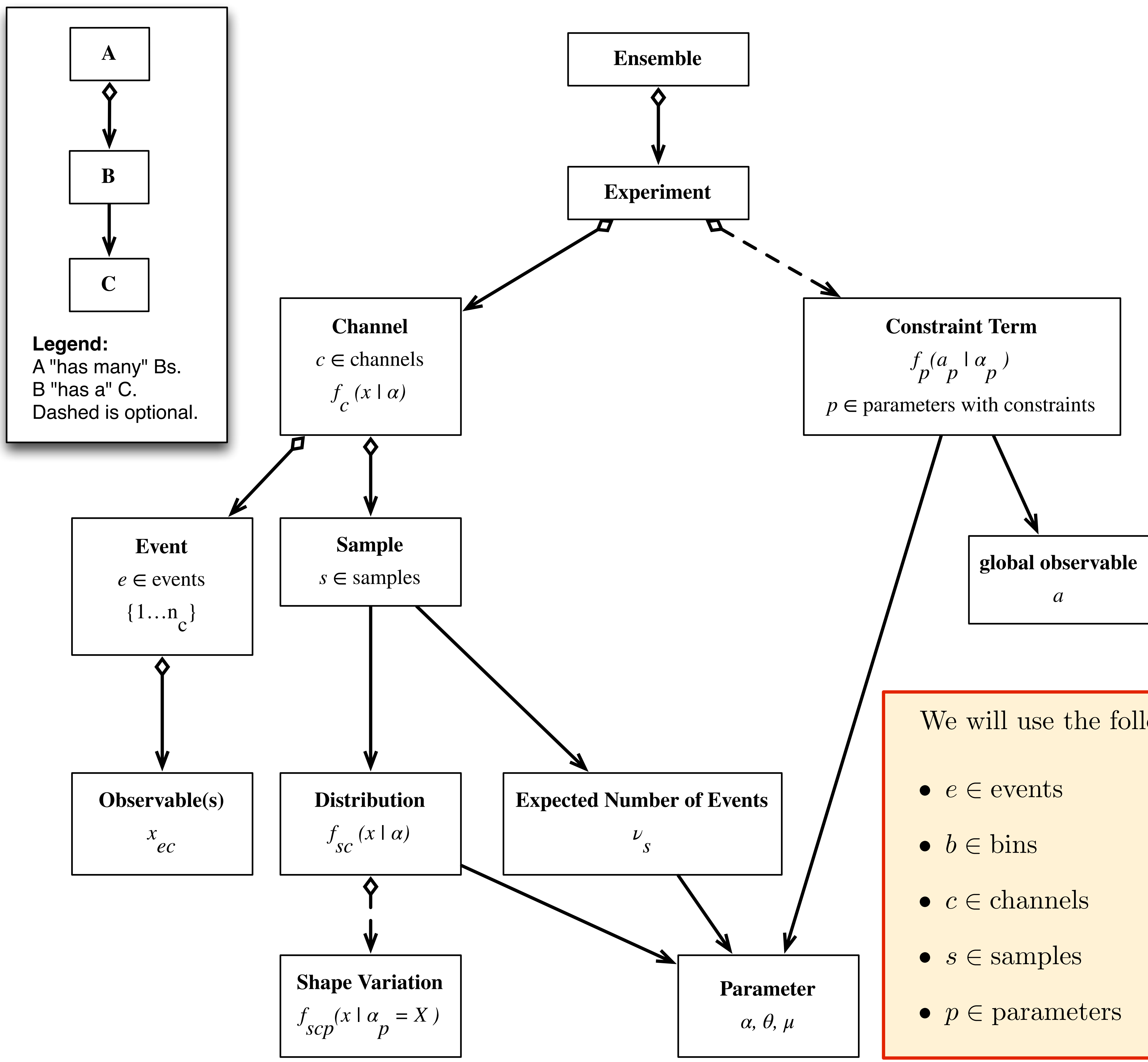
Simultaneous Multi-Channel Model with constraints:

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

where

$$\mathcal{D}_{\text{sim}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_{c_{\max}}\}, \quad \mathcal{G} = \{a_p\} \quad \text{for} \quad p \in \mathbb{S}$$

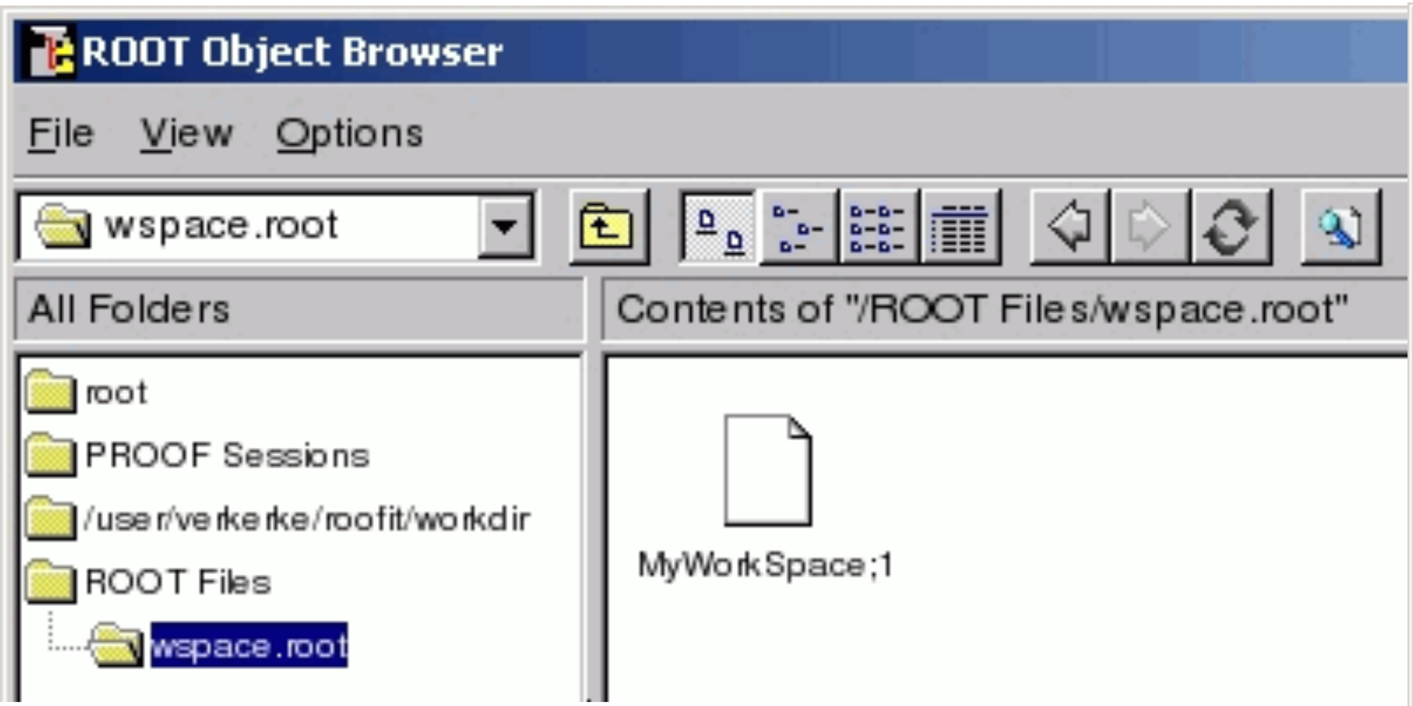
CONCEPTUAL BUILDING BLOCKS

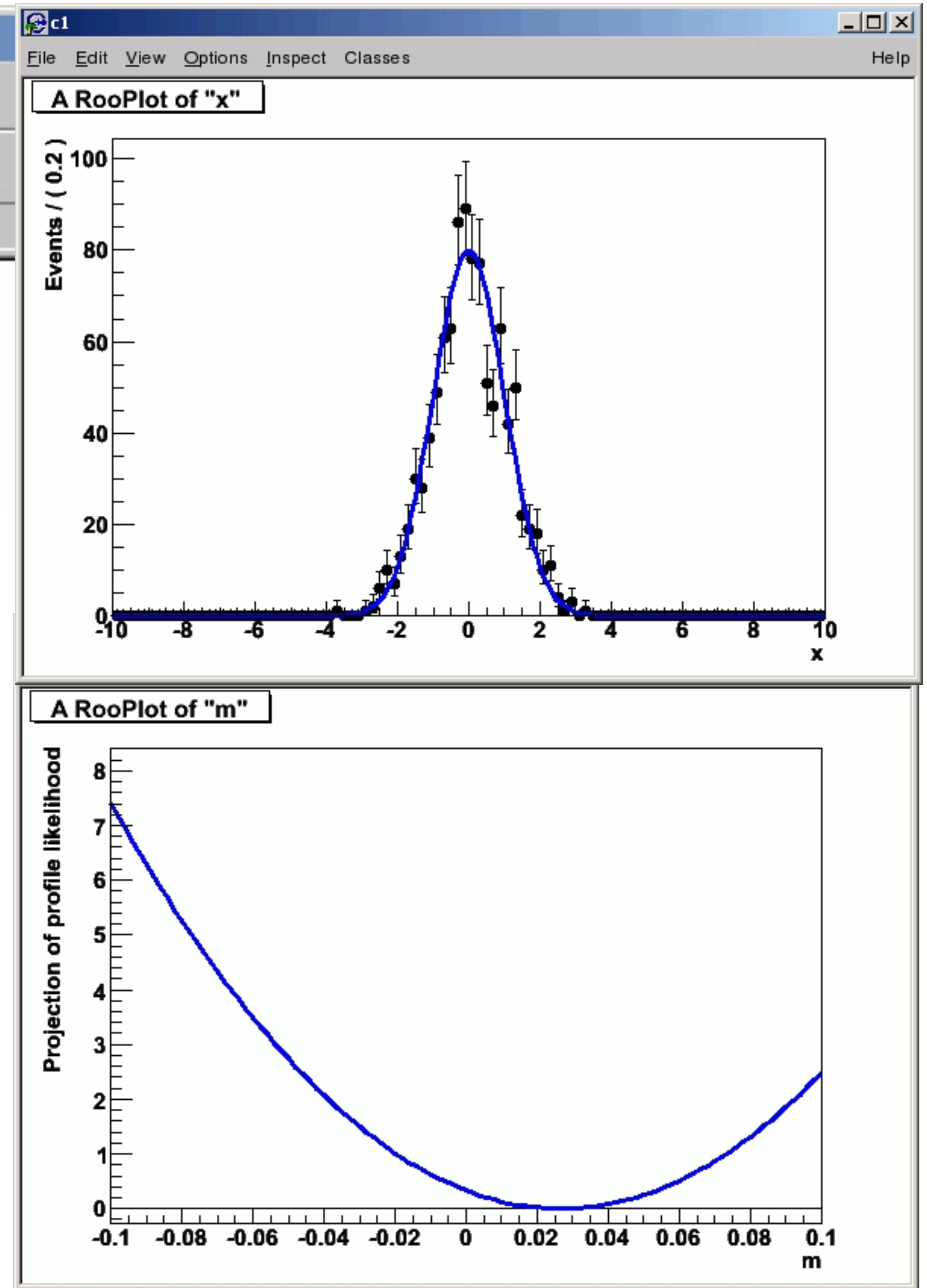


We will use the following mnemonic index conventions:

- $e \in$ events
- $b \in$ bins
- $c \in$ channels
- $s \in$ samples
- $p \in$ parameters

EXAMPLE OF DIGITAL PUBLISHING





RooFit's Workspace now provides the ability to save in a ROOT file the full likelihood model, any priors you might want, and the minimal data necessary to reproduce likelihood function.

Need this for combinations, as p-value is not sufficient information for a proper combination.

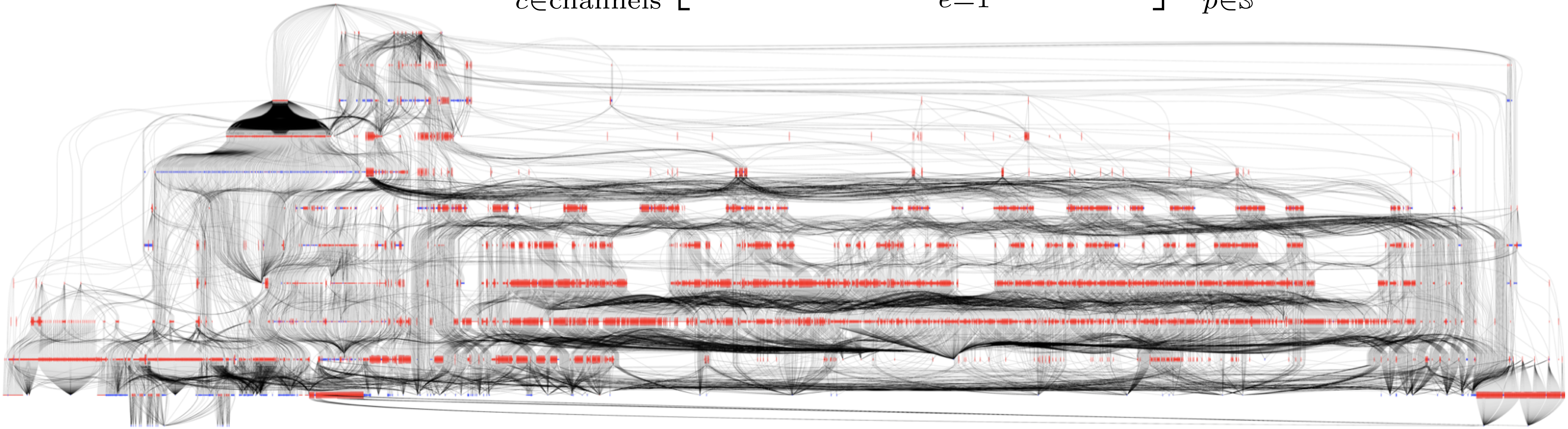
VISUALIZING THE COMBINED MODEL

State of the art: At the time of the discovery, the combined Higgs search included 100 disjoint channels and >500 nuisance parameters

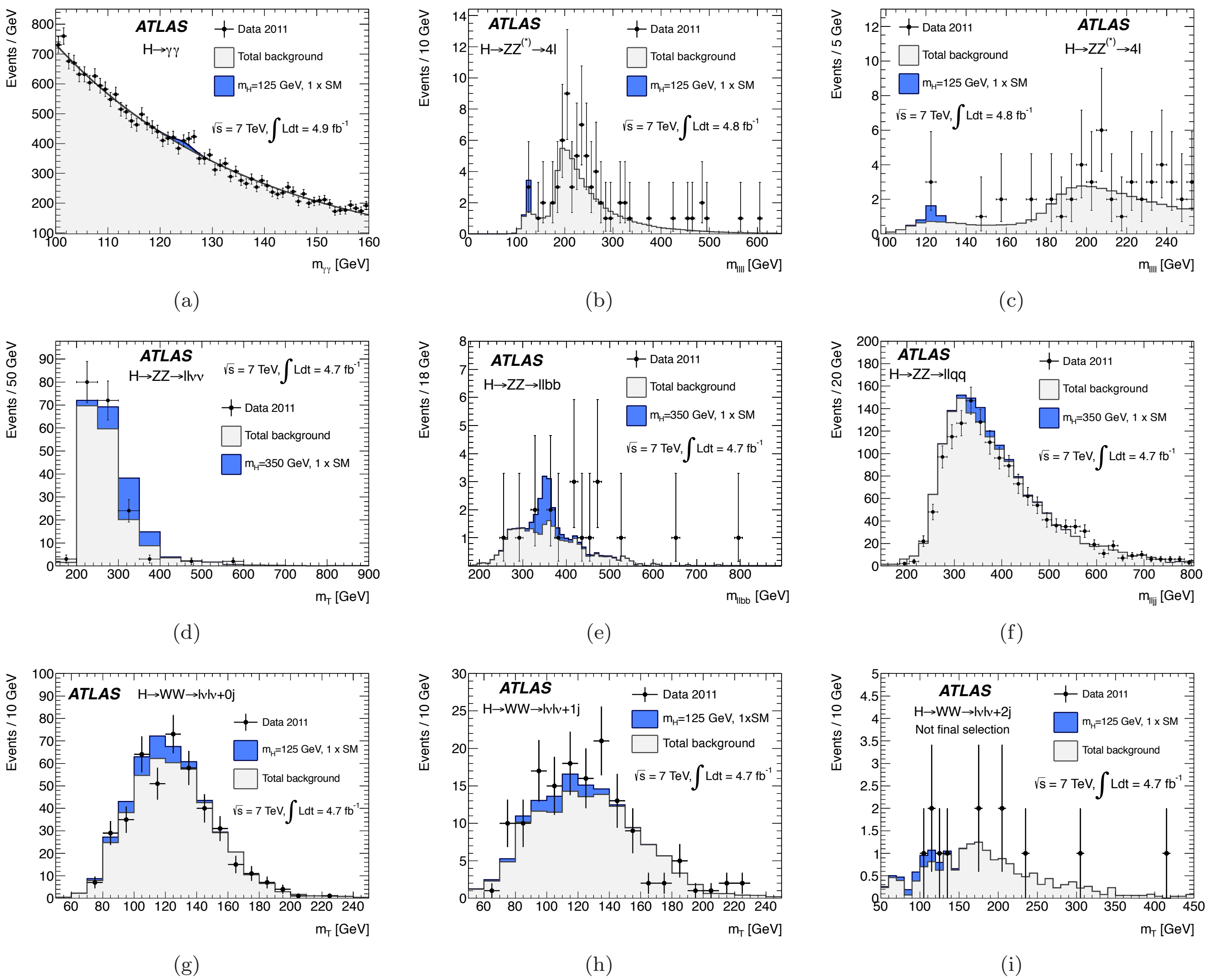
RooFit / RooStats: is the modeling language (C++) which provides technologies for collaborative modeling

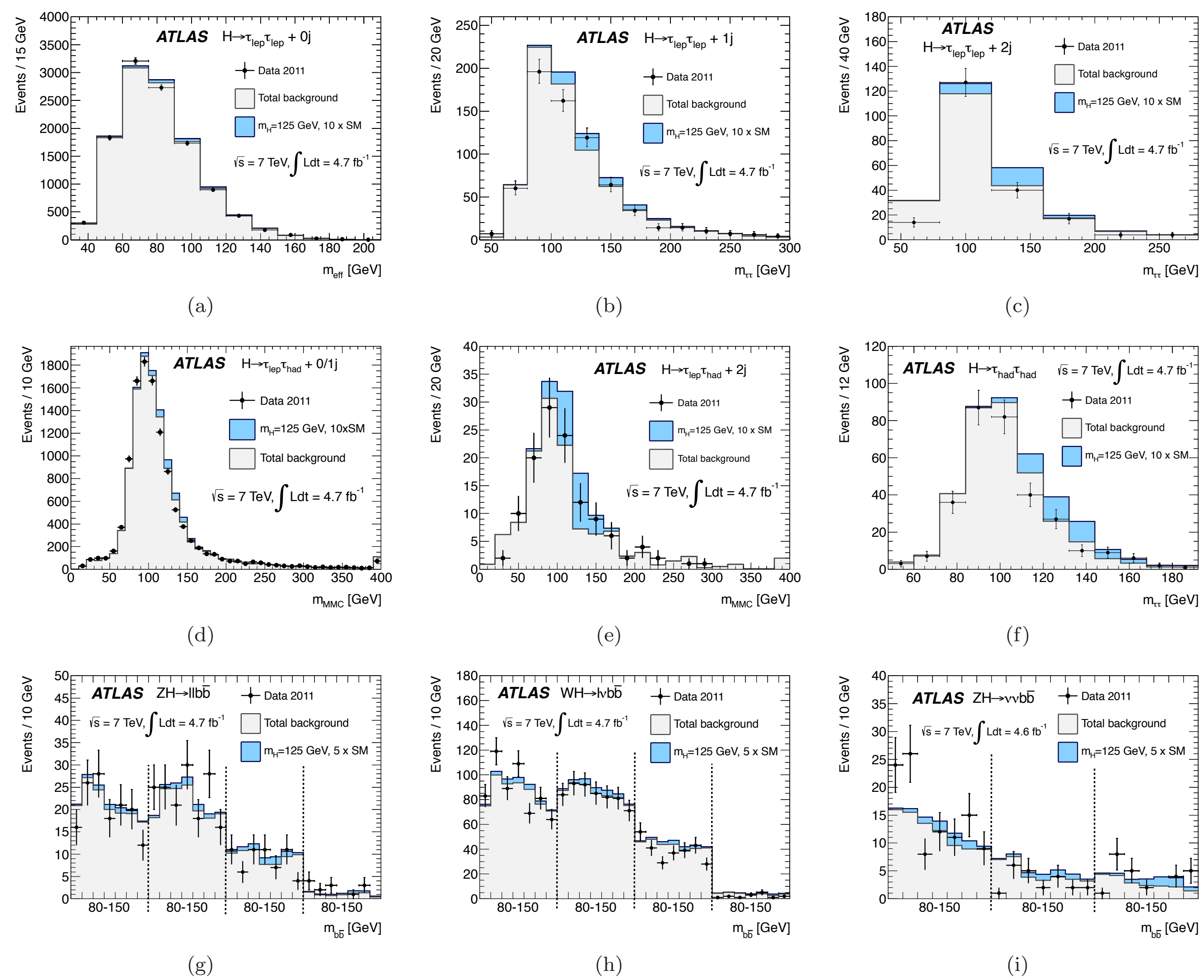
- provides technology to publish likelihood functions digitally
- and more, it's the full model so we can also generate pseudo-data

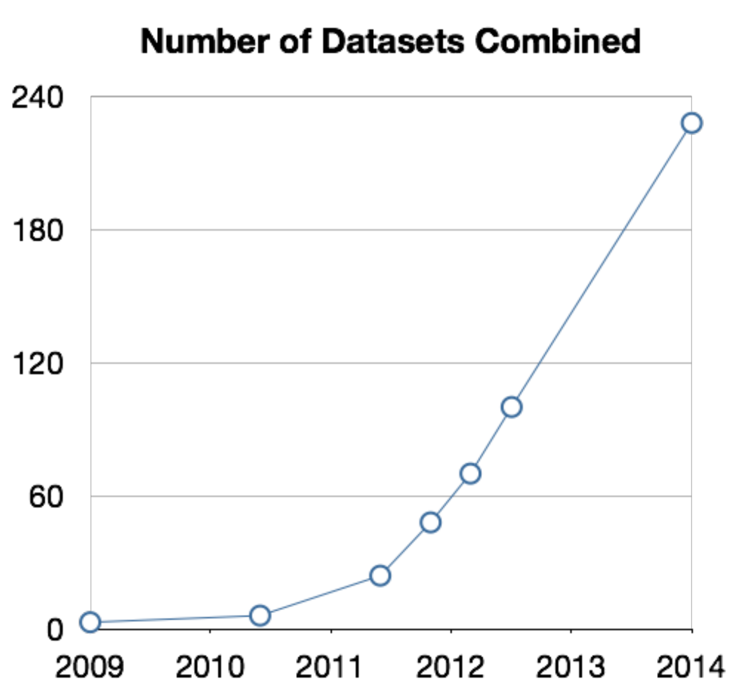
$$\mathbf{f}_{\rm tot}(\mathcal{D}_{\rm sim}, \mathcal{G}|\boldsymbol{\alpha}) = \prod_{c \in \rm channels} \left[\text{Pois}(n_c|\nu_c(\boldsymbol{\alpha})) \prod_{e=1}^{n_c} f_c(x_{ce}|\boldsymbol{\alpha}) \right] \cdot \prod_{p \in \mathbb{S}} f_p(a_p|\alpha_p)$$

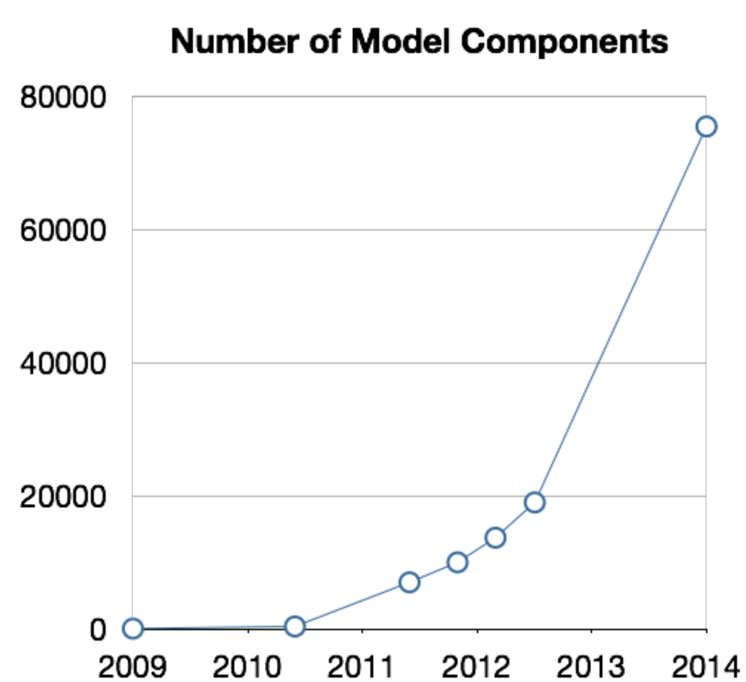


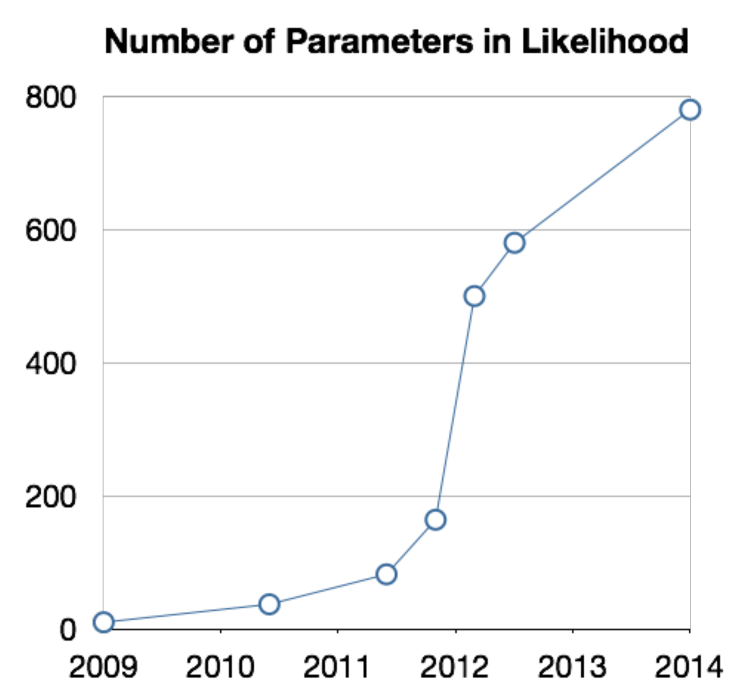
Evolution of Model Complexity



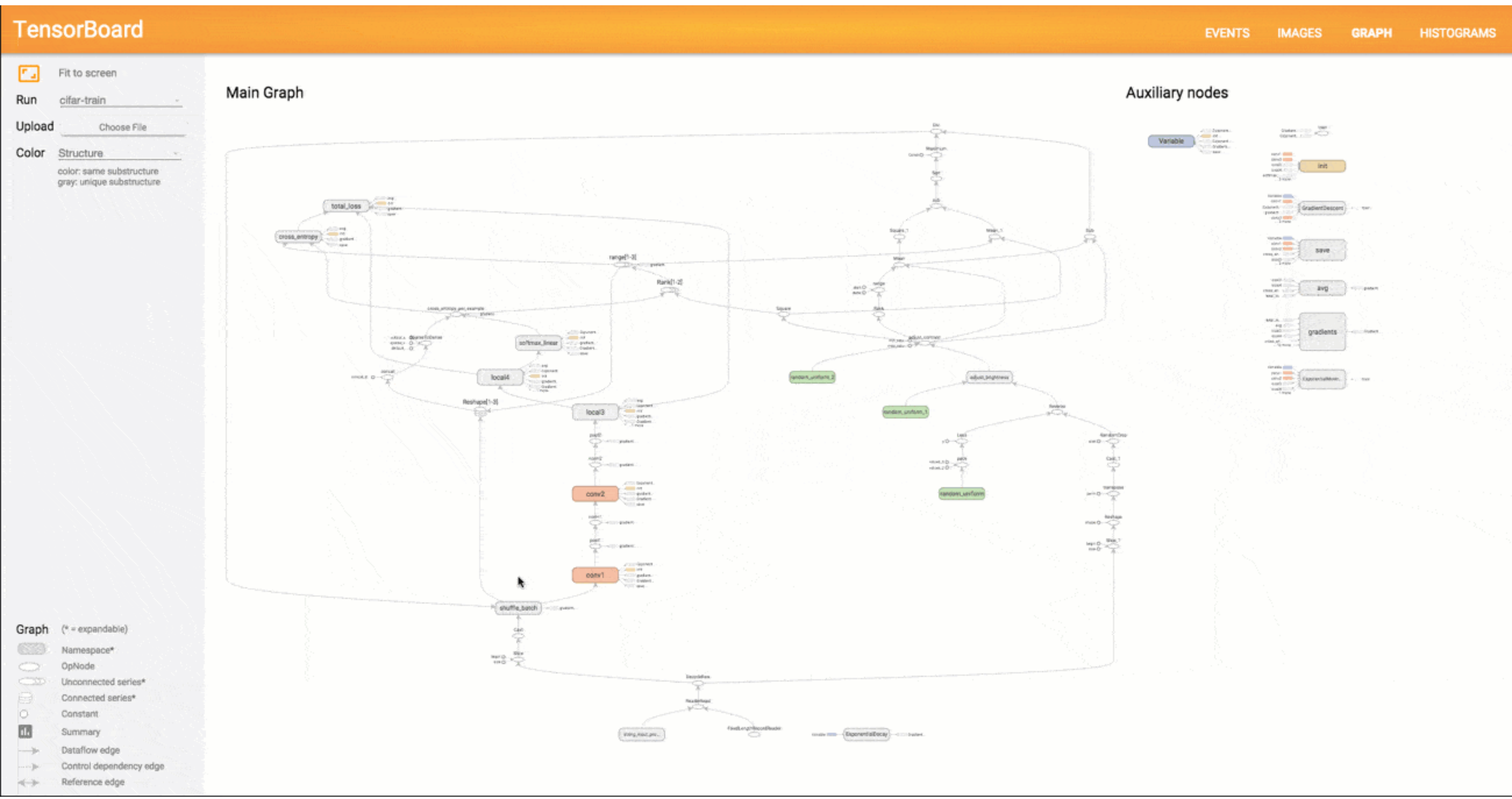




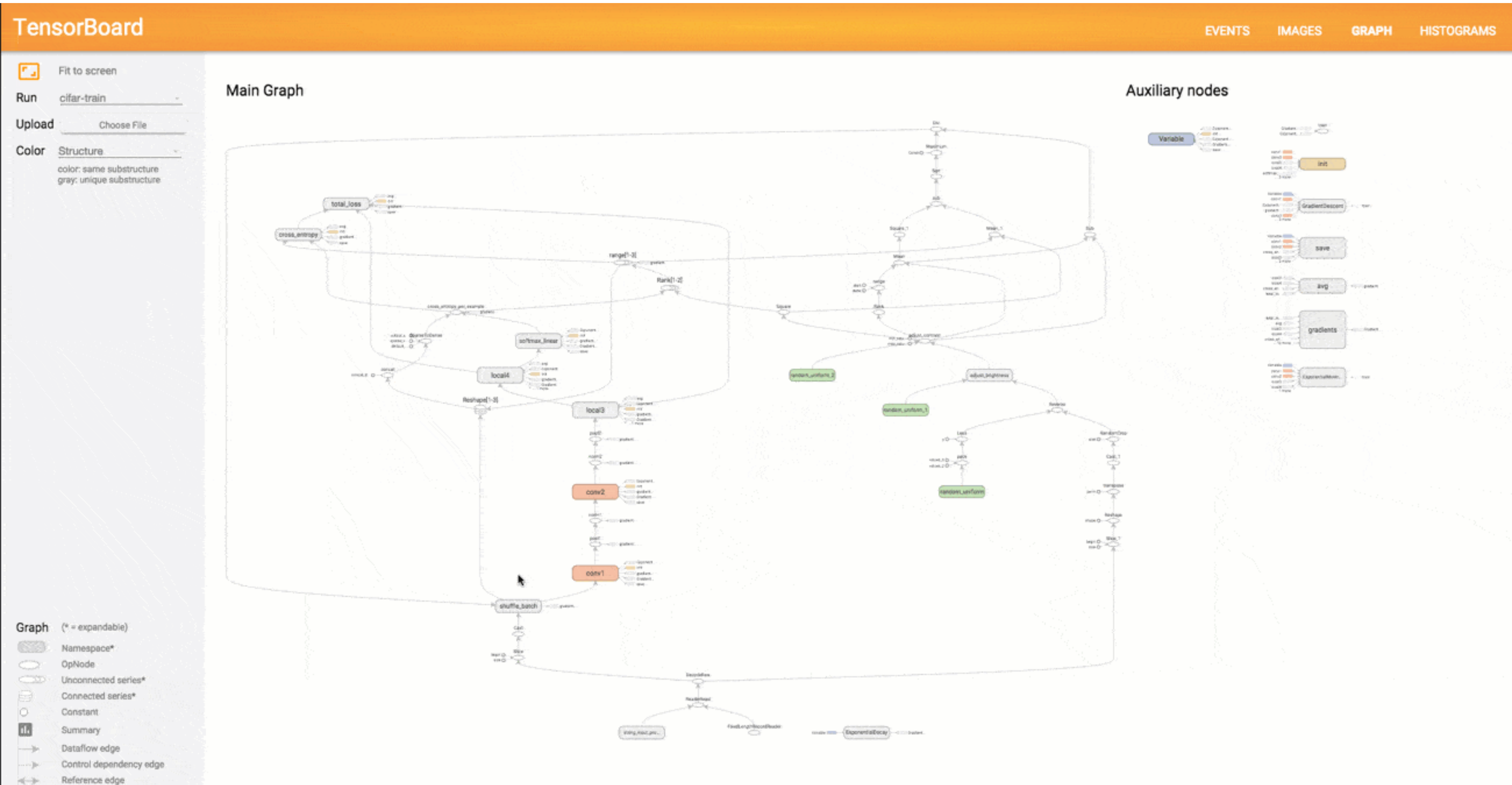




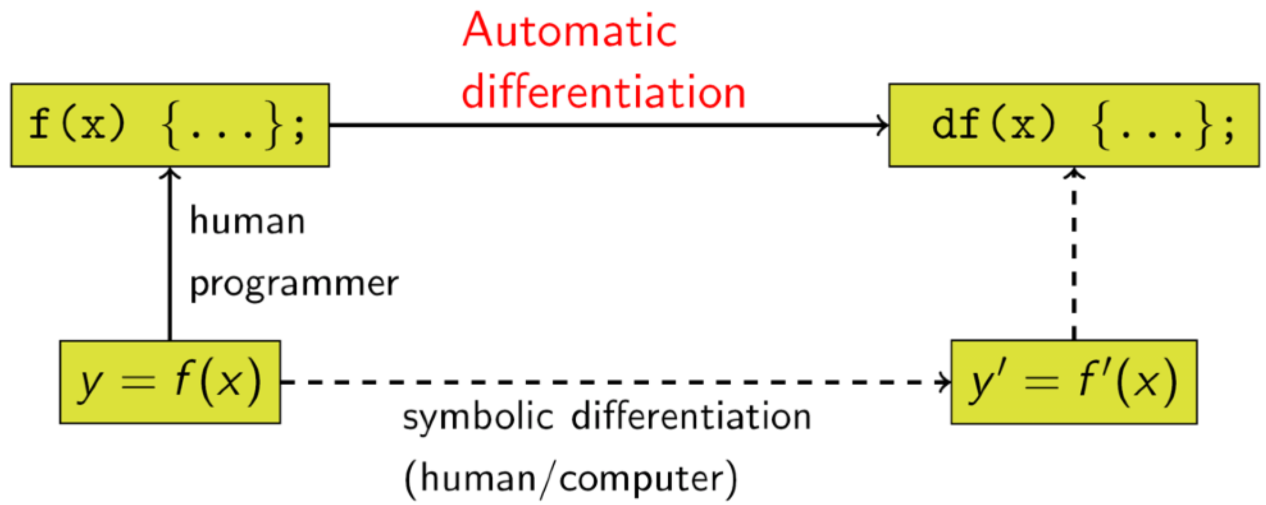
Modern Machine Learning tools like TensorFlow express the model in a similar way as a Directed Acyclic Graph (DAG)



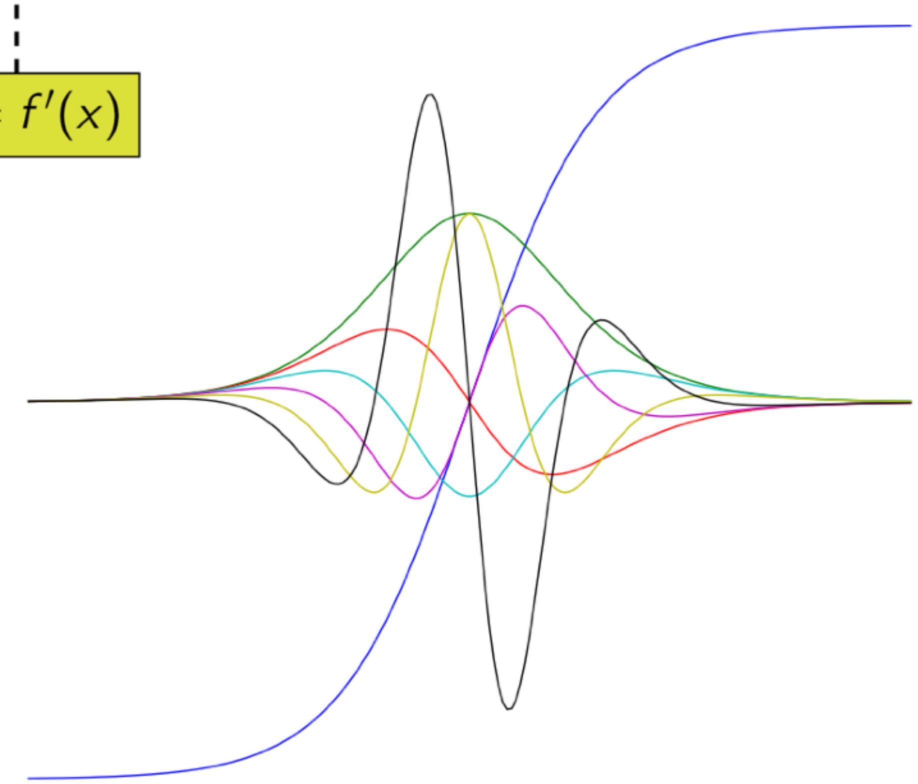
Modern Machine Learning tools like TensorFlow express the model in a similar way as a Directed Acyclic Graph (DAG)



AUTOMATIC DIFFERENTIATION




```
>>> import autograd.numpy as np  # Thinly-wrapped numpy
>>> from autograd import grad    # The only autograd function you may ever need
>>>
>>> def tanh(x):                 # Define a function
...     y = np.exp(-2.0 * x)
...     return (1.0 - y) / (1.0 + y)
...
>>> grad_tanh = grad(tanh)       # Obtain its gradient function
>>> grad_tanh(1.0)               # Evaluate the gradient at x = 1.0
0.41997434161402603
>>> (tanh(1.0001) - tanh(0.9999)) / 0.0002  # Compare to finite differences
0.41997434264973155
```

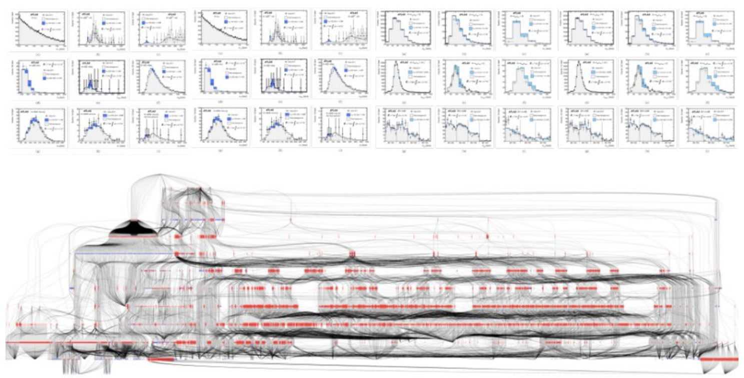


We can continue to differentiate as many times as we like, and use numpy's vectorization of scalar-valued functions across many different input values:

```
>>> from autograd import elementwise_grad as egrad  # for functions that vectorize over inputs
>>> import matplotlib.pyplot as plt
>>> x = np.linspace(-7, 7, 200)
>>> plt.plot(x, tanh(x),
...          x, egrad(tanh)(x),                                     # first  derivative
...          x, egrad(egrad(tanh))(x),                              # second derivative
...          x, egrad(egrad(egrad(tanh)))(x),                       # third  derivative
...          x, egrad(egrad(egrad(egrad(tanh))))(x),                # fourth derivative
...          x, egrad(egrad(egrad(egrad(egrad(tanh)))))(x),         # fifth  derivative
...          x, egrad(egrad(egrad(egrad(egrad(egrad(tanh))))))(x))  # sixth  derivative
>>> plt.show()
```

https://en.wikipedia.org/wiki/Automatic_differentiation

Probabilistic programming frameworks

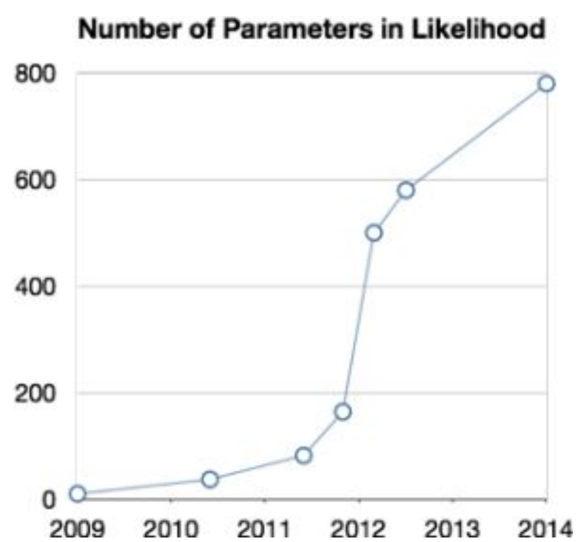


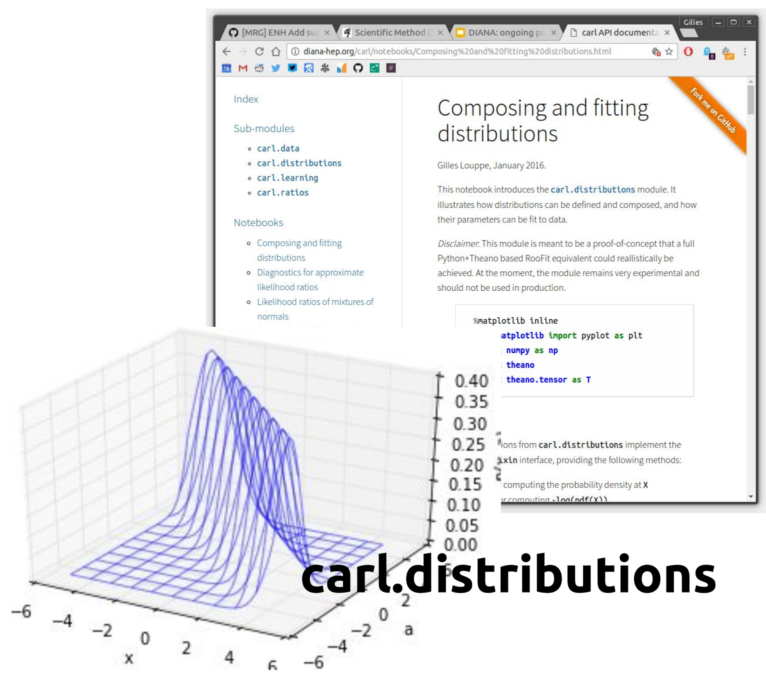
$$f_{tot}(\mathcal{D}_{sim}, \mathcal{G}|\boldsymbol{\alpha}) = \prod_{c \in \text{channels}} \left[\text{Pois}(n_c|\nu_c(\boldsymbol{\alpha})) \prod_{e=1}^{n_c} f_c(x_{ce}|\boldsymbol{\alpha}) \right] \cdot \prod_{p \in \mathbb{S}} f_p(a_p|\alpha_p)$$

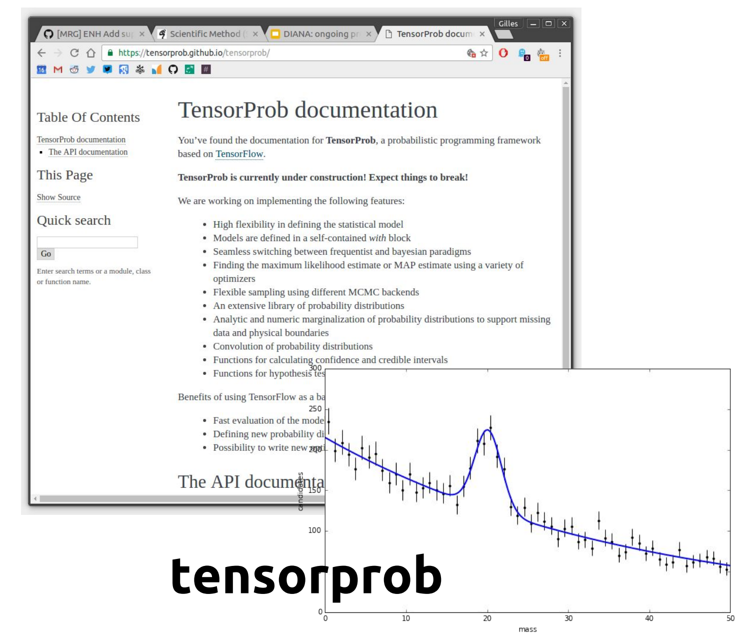
RooFit serves us well, but shows limits in terms of **scalability**.

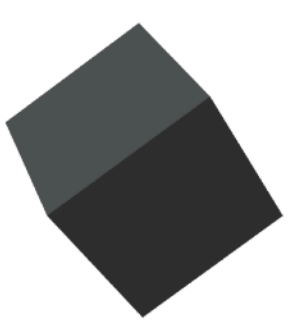
Using a data flow graph framework, RooFit would be **distributed**, **GPU-enabled** and automatically **differentiable**.

Feasibility? Certainly **within reach!** As illustrated by our tentative proof-of-concepts carl.distributions [Gilles Louppe] and tensorprob [Igor Babuschkin, now at DeepMind]. See also Edward.









Edward

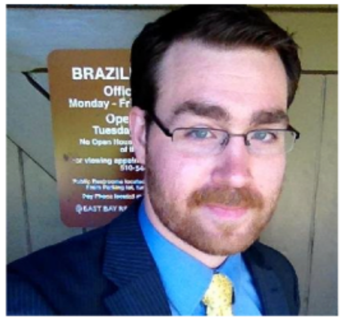
A library for probabilistic modeling, inference, and criticism.

Edward is a Python library for probabilistic modeling, inference, and criticism. It is a testbed for fast experimentation and research with probabilistic models, ranging from classical hierarchical models on small data sets to complex deep probabilistic models on large data sets. Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.

It supports **modeling** with

Dustin Tran

Ph.D. Student
Columbia University
dustin@cs.columbia.edu (@dustinvtran, http://dustintran.com



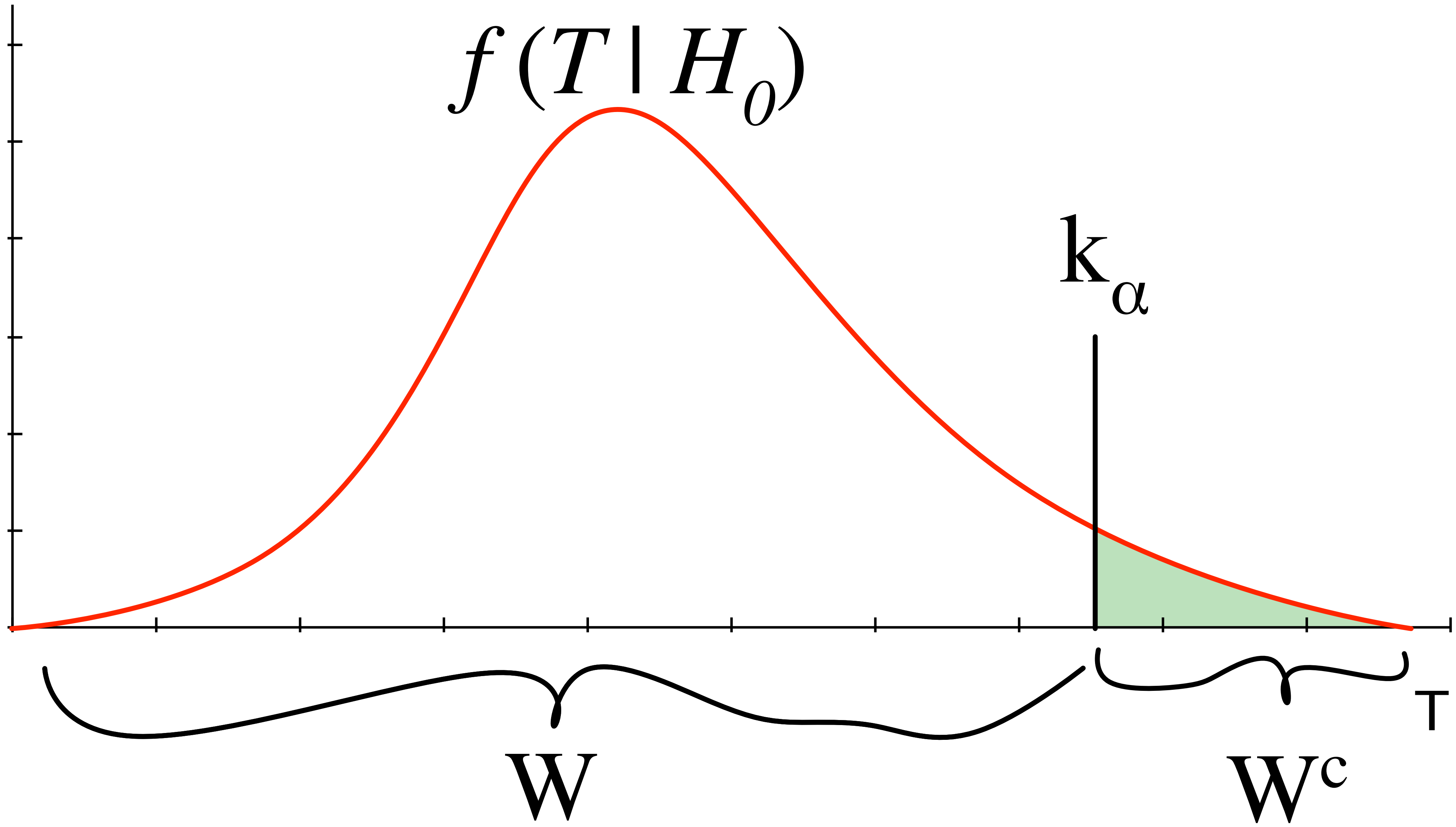
Matthew Feickert

High Energy Physics Ph.D. Candidate
Southern Methodist University
matthew.feickert@cern.ch or mfeickert@smu.edu
GitHub: matthewfeickert @HEPfeickert

Profile Likelihood Ratio

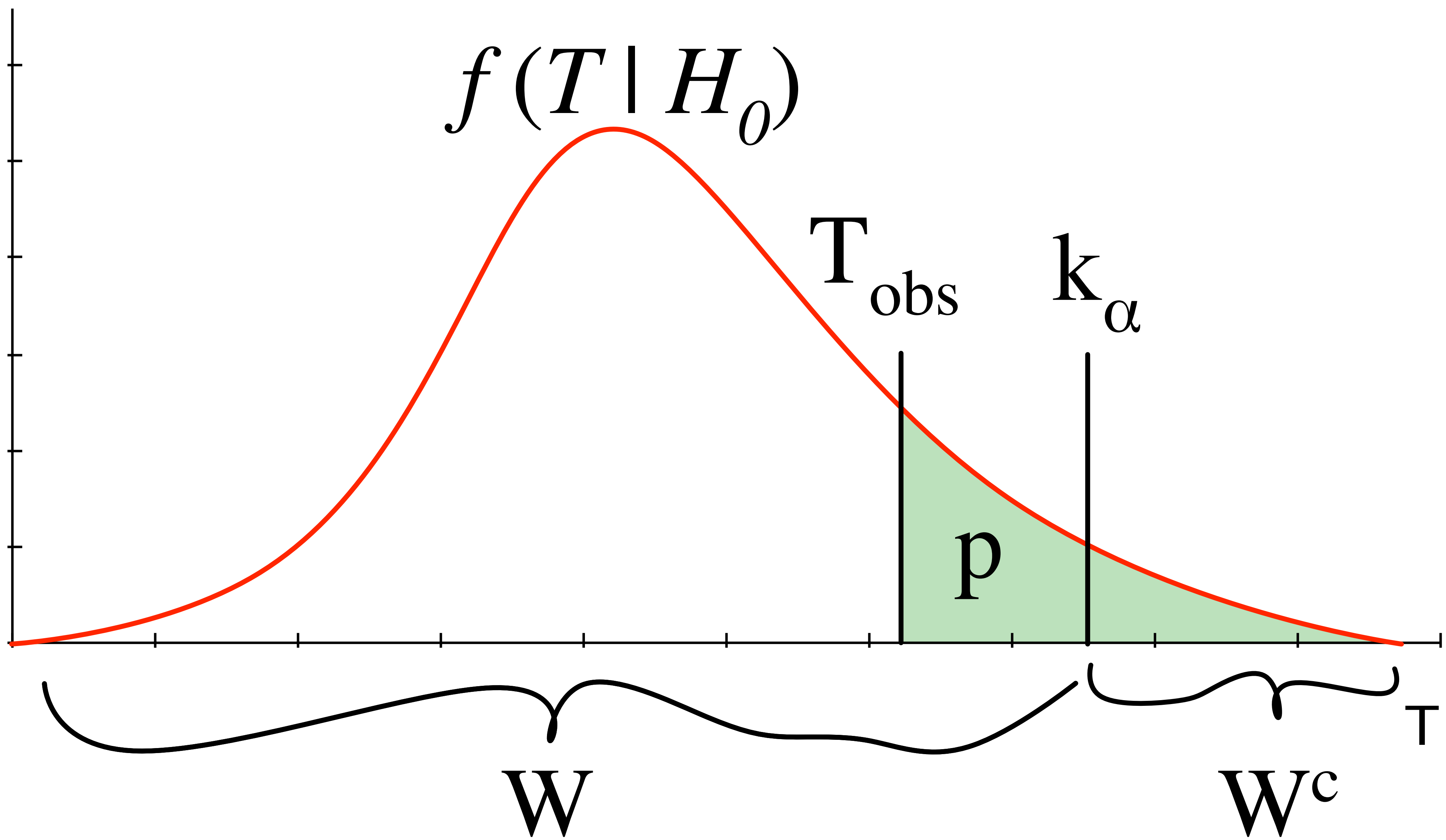
Instead of choosing to accept/reject H_0
one can compute the p-value

$$p = \int_{T_o}^{\infty} f(T|H_0)$$



Instead of choosing to accept/reject H_0
one can compute the p-value

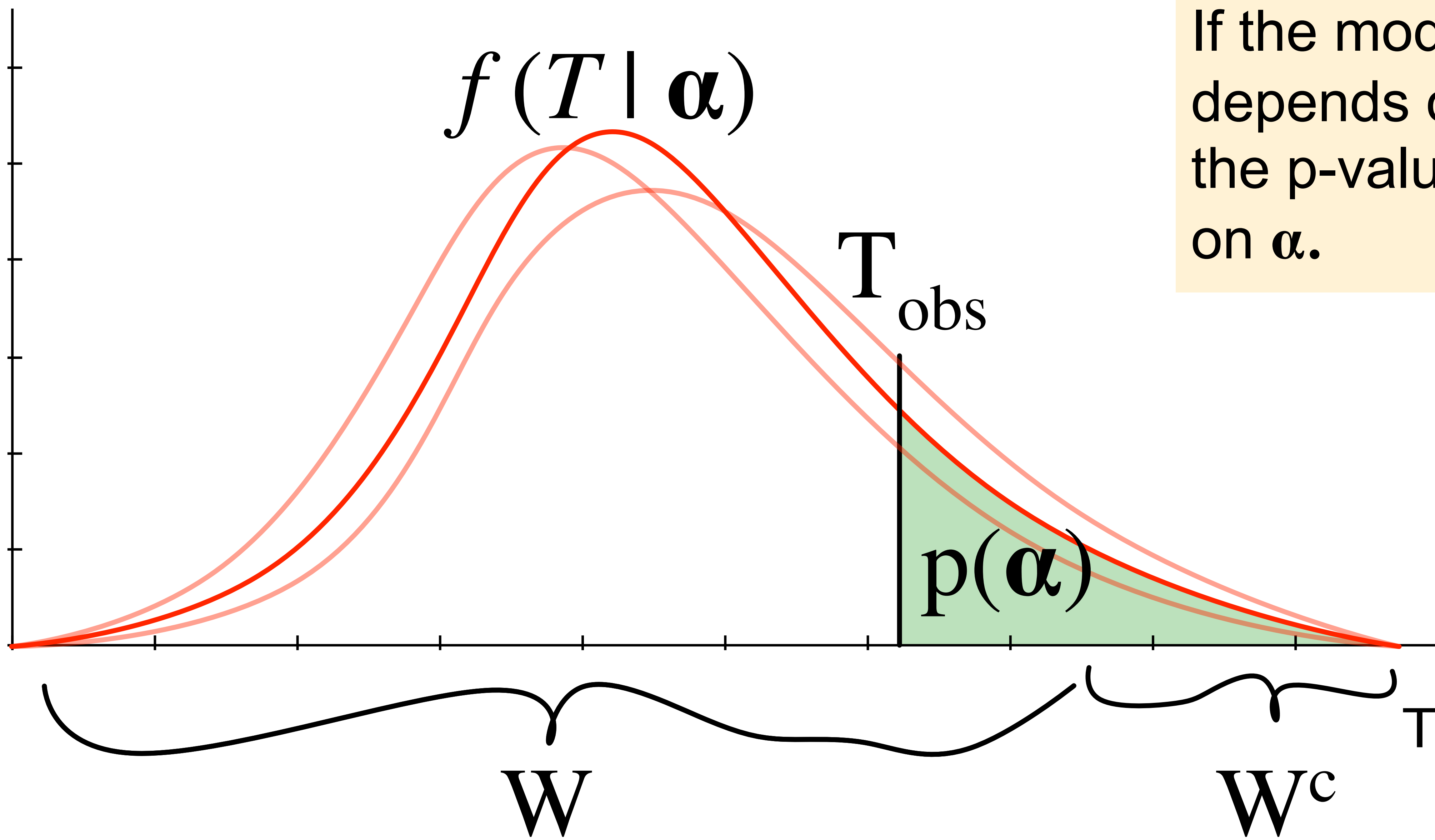
$$p = \int_{T_o}^{\infty} f(T|H_0)$$



Instead of choosing to accept/reject H_0 one can compute the p-value

$$p = \int_{T_o}^{\infty} f(T|H_0)$$

If the model for the data depends on parameters **α** the p-value also depends on **α**.

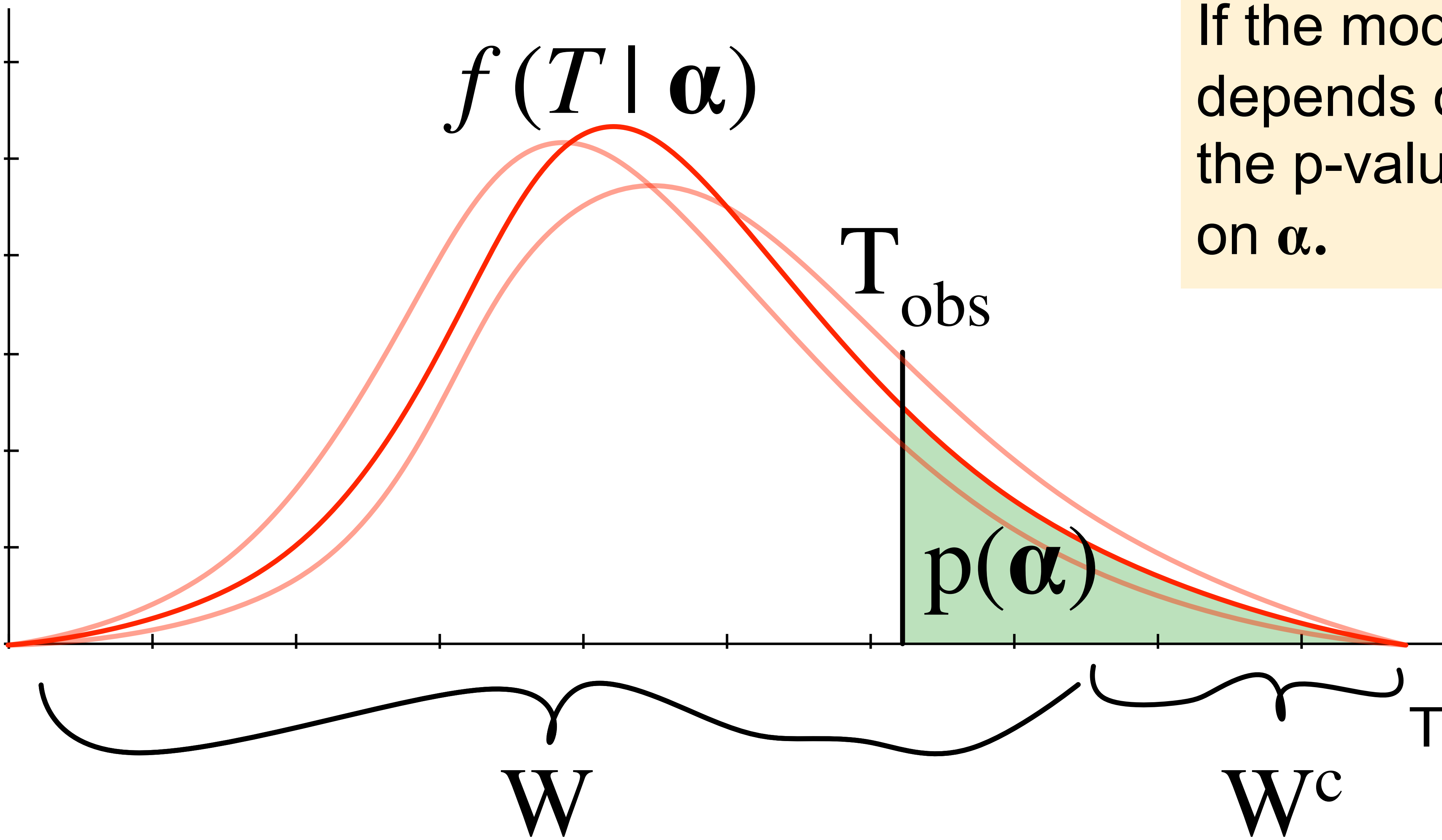


$$p(\boldsymbol{\alpha}) = \int_{T_0}^{\infty} f(T|\boldsymbol{\alpha})dT = \int \mathbf{f}(\mathcal{D}|\boldsymbol{\alpha})\,\theta(T(\mathcal{D}) - T_0)\,d\mathcal{D} = P(T \geq T_0|\boldsymbol{\alpha})$$

P-VALUES

When the model has nuisance parameters, only reject the null if $p(\boldsymbol{\alpha})$ sufficiently small **for all values** of the nuisance parameters.

If the model for the data depends on parameters $\boldsymbol{\alpha}$ the p-value also depends on $\boldsymbol{\alpha}$.



$$p(\boldsymbol{\alpha}) = \int_{T_0}^{\infty} f(T|\boldsymbol{\alpha})dT = \int \mathbf{f}(\mathcal{D}|\boldsymbol{\alpha})\,\theta(T(\mathcal{D}) - T_0)\,d\mathcal{D} = P(T \geq T_0|\boldsymbol{\alpha})$$

The Profile Likelihood Ratio

Consider our general model with a single parameter of interest μ

- let μ=0 be no signal, μ=1 nominal signal

Define **profile likelihood ratio**

$$\lambda(\mu) = \frac{L(\mu, \hat{\hat{\theta}}(\mu))}{L(\hat{\mu}, \hat{\theta})} = \frac{f(\mathcal{D}, \mathcal{G} | \mu, \hat{\hat{\theta}}(\mu; \mathcal{D}, \mathcal{G}))}{f(\mathcal{D}, \mathcal{G} | \hat{\mu}, \hat{\theta})}$$

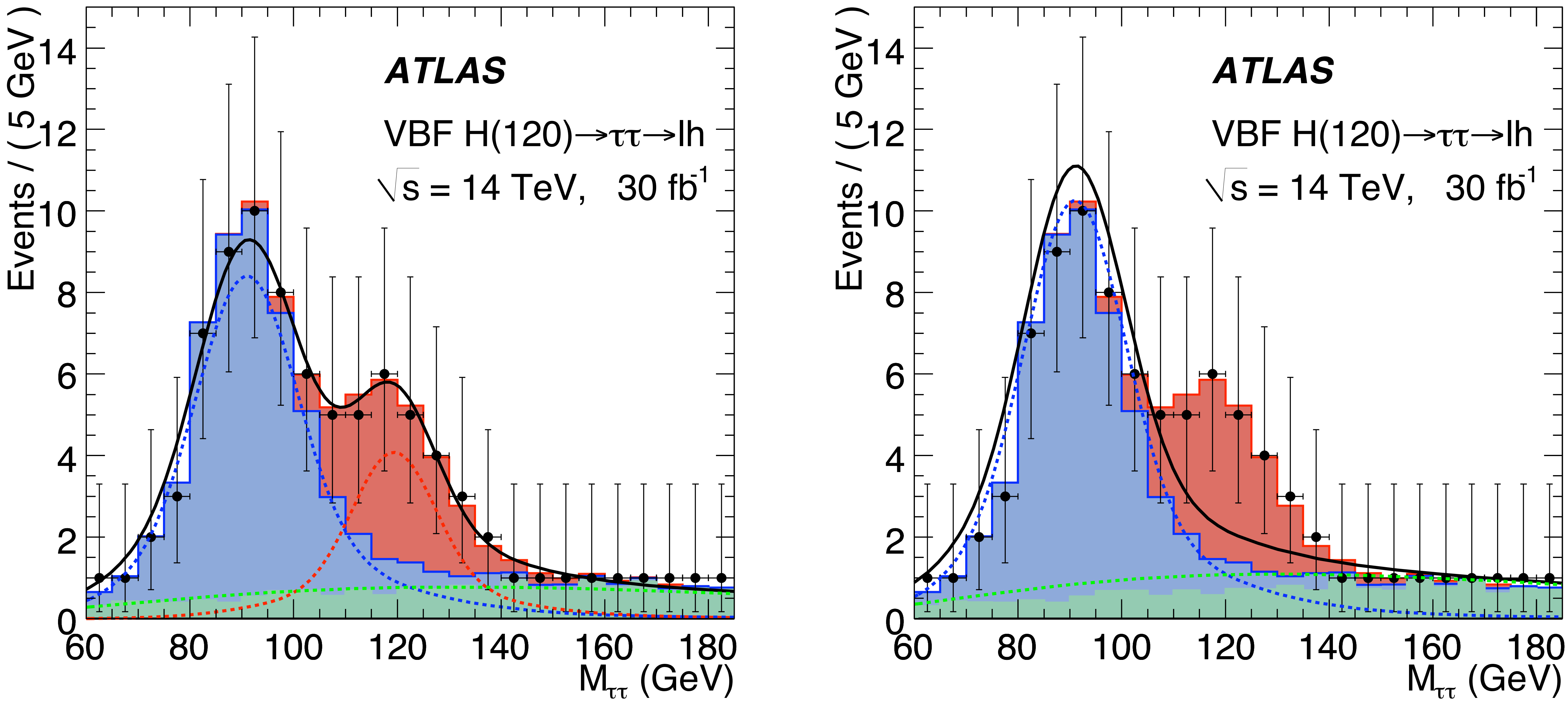
- where $\hat{\hat{\theta}}(\mu; \mathcal{D}, \mathcal{G})$ is best fit with μ fixed (the constrained maximum likelihood estimator, depends on data)
- and $\hat{\theta}$ and $\hat{\mu}$ are best fit with both left floating (unconstrained)
- Tevatron used $Q_{Tev} = \lambda(\mu=1)/\lambda(\mu=0)$ as generalization of Q_{LEP}

AN EXAMPLE

Essentially, you need to fit your model to the data twice:
once with everything floating, and once with signal fixed to 0

$$\lambda(\mu=0) = \frac{L(\mu=0, \hat{\hat{\theta}}(\mu=0))}{L(\hat{\mu}, \hat{\theta})} = \frac{f(\mathcal{D}, \mathcal{G}|\mu=0, \hat{\hat{\theta}}(\mu=0; \mathcal{D}, \mathcal{G})\,)}{f(\mathcal{D}, \mathcal{G}|\hat{\mu}, \hat{\theta})}$$

$f(\mathcal{D}, \mathcal{G}|\hat{\mu}, \hat{\theta})$ $\qquad$ $f(\mathcal{D}, \mathcal{G}|\mu=0, \hat{\hat{\theta}}(\mu=0; \mathcal{D}, \mathcal{G})\,)$



PROPERTIES OF THE PROFILE LIKELIHOOD RATIO

After a close look at the profile likelihood ratio

$$\lambda(\mu) = \frac{L(\mu, \hat{\hat{\theta}}(\mu))}{L(\hat{\mu}, \hat{\theta})} = \frac{f(\mathcal{D}, \mathcal{G}|\mu, \hat{\hat{\theta}}(\mu; \mathcal{D}, \mathcal{G})\,)}{f(\mathcal{D}, \mathcal{G}|\hat{\mu}, \hat{\theta})}$$

one can see the function is independent of true values of θ

- though its distribution might depend indirectly

Wilks's theorem states that under certain conditions the distribution of $-2 \ln \lambda\ (\mu=\mu_0)$ given that the true value of μ is μ_0 converges to a chi-square distribution

- more on this later, but the important points are:
- "asymptotic distribution" is known and it is independent of θ !
  - a quantity whose distribution is independent of θ is called a **pivot**
  - more complicated if parameters have boundaries (eg. $\mu \geq 0$)

Thus, we can calculate the p-value for the background-only hypothesis without having to generate Toy Monte Carlo!

"THE ASIMOV PAPER"

Recently we showed how to generalize this asymptotic approach

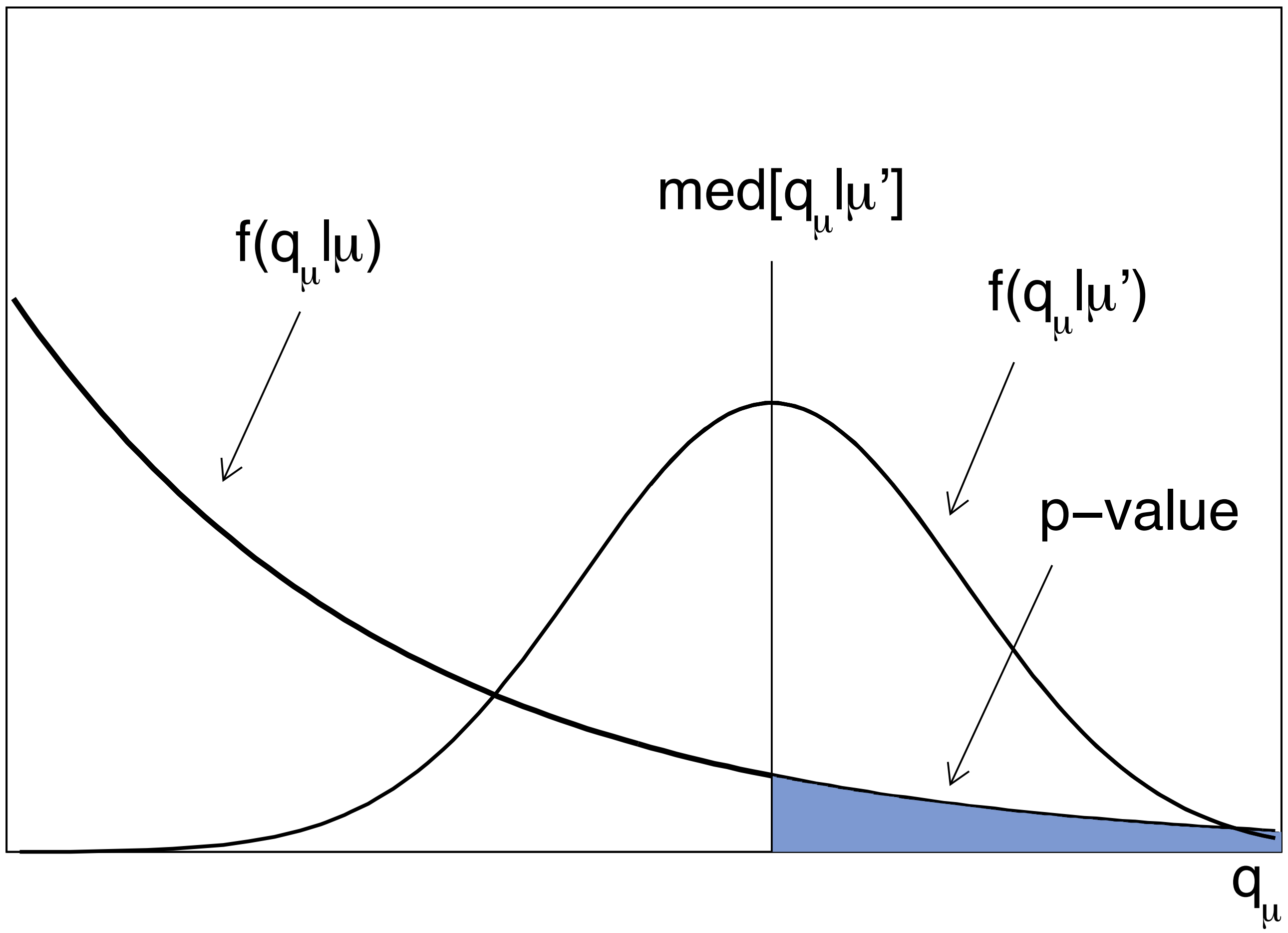
- generalize Wilks's theorem when boundaries are present
- use Wald's result for distribution for alternate hypothesis $f(-2\log\lambda(\mu) \mid \mu')$

Asymptotic formulae for likelihood-based tests of new physics

Glen Cowan, Kyle Cranmer, Eilam Gross, Ofer Vitells

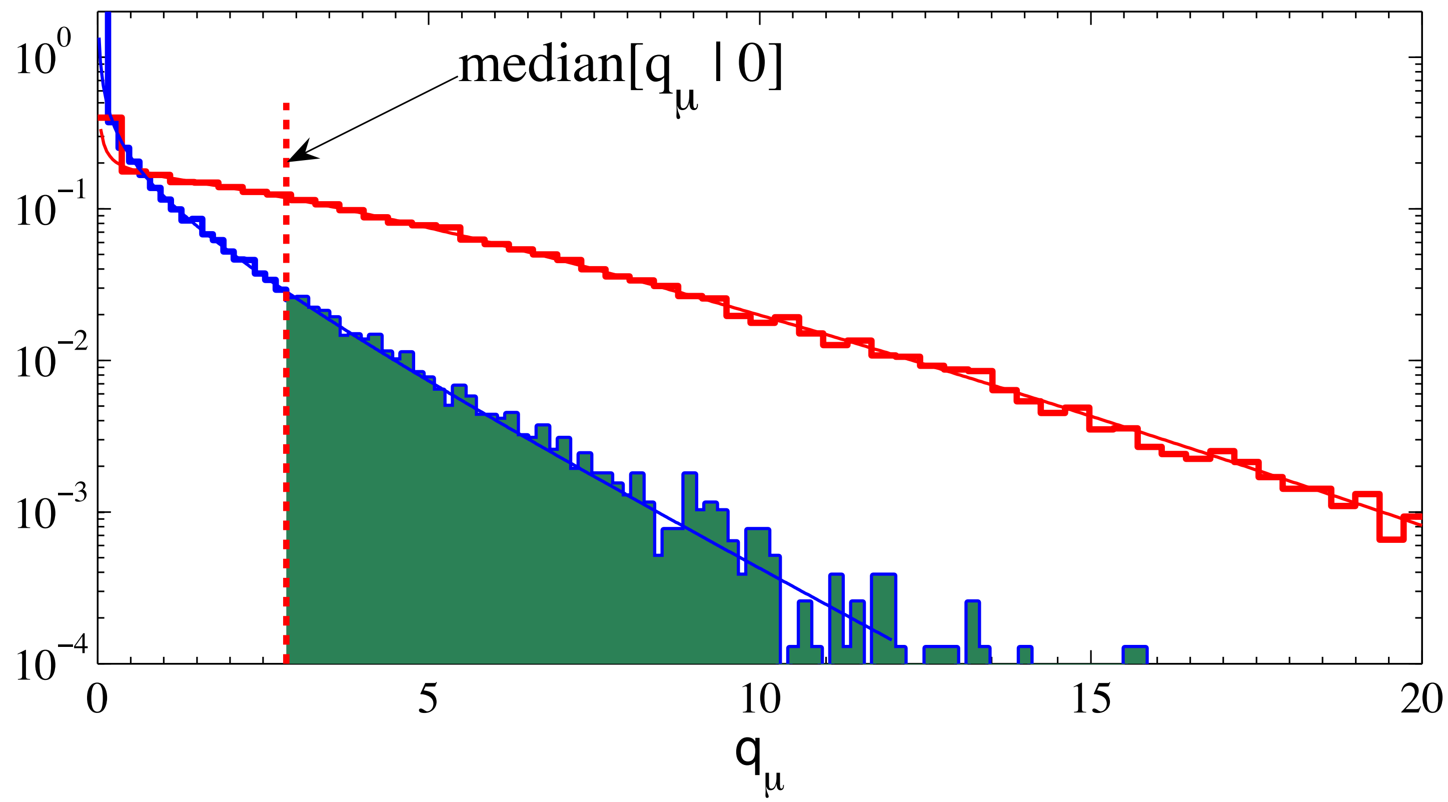
Eur.Phys.J.C71:1554,2011

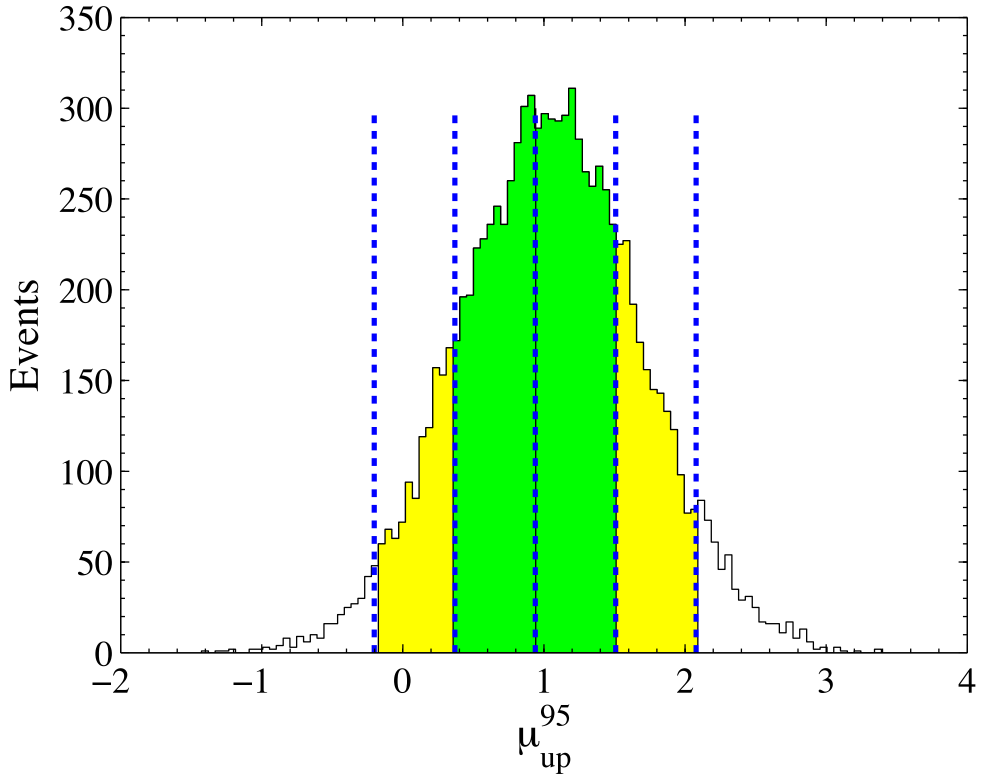
http://arxiv.org/abs/1007.1727v2



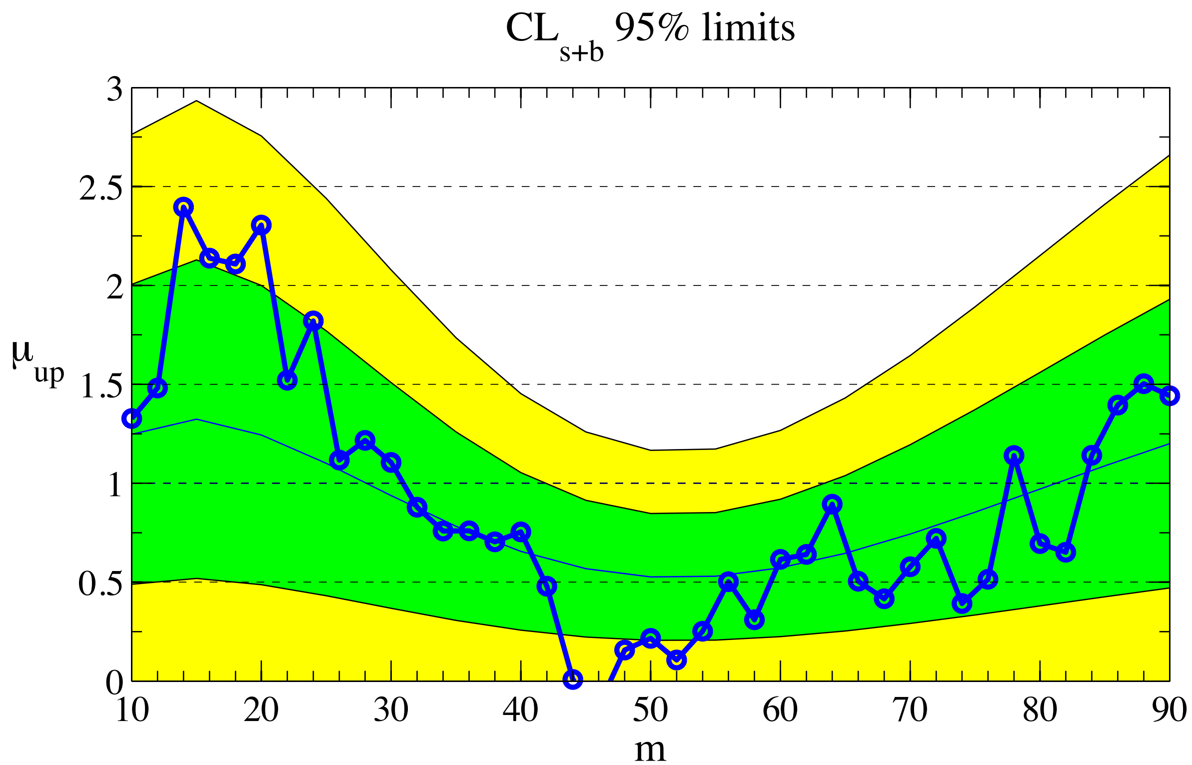
COMPARISON OF ASYMPTOTIC AND ENSEMBLES

Compare asymptotic distributions to distributions obtained with large ensembles of pseudo-experiments generated with Monte Carlo techniques





This is a significant development as building this distribution from Monte Carlo approaches can take 100,000 CPU hours for Higgs search!

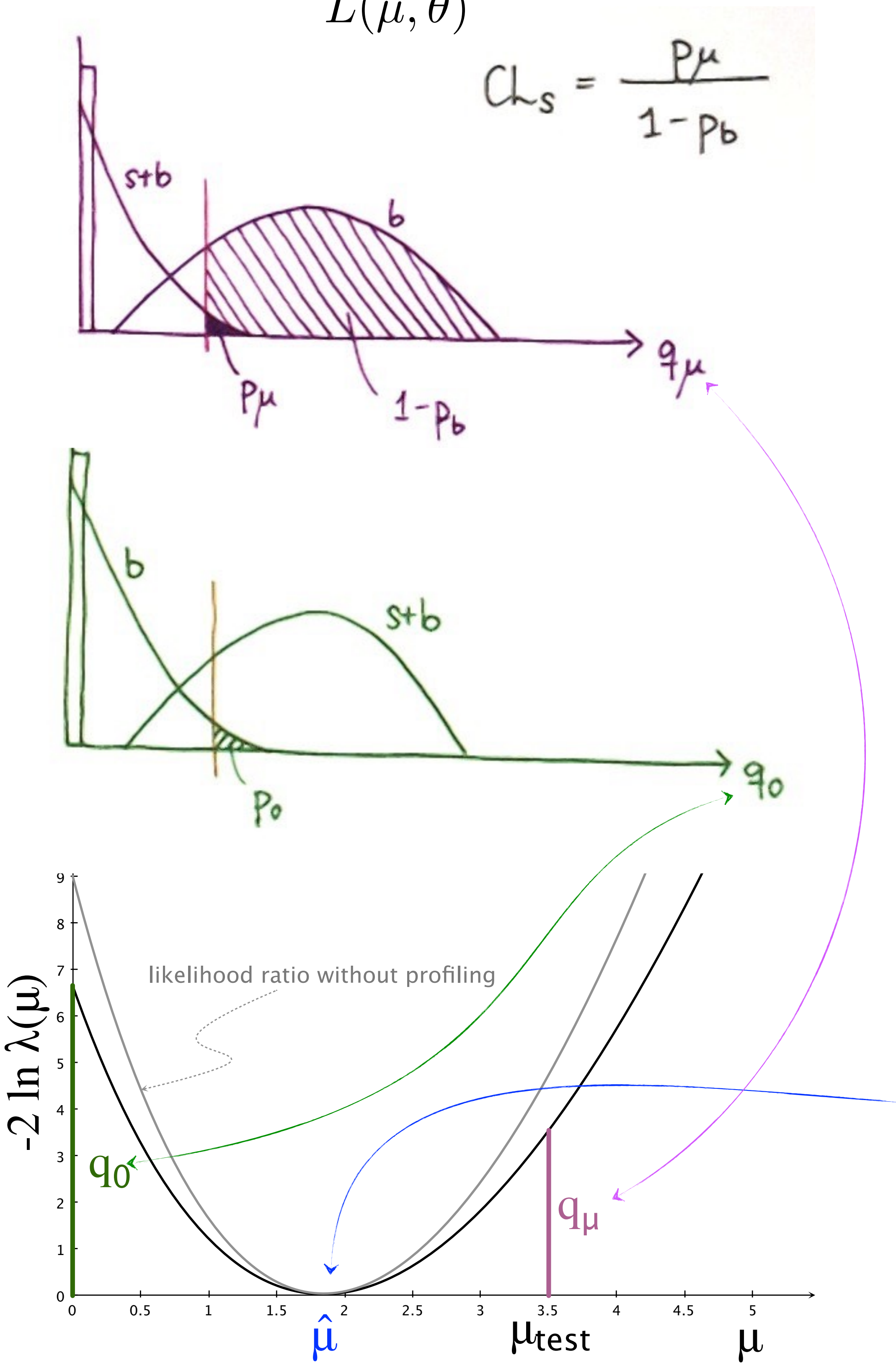


G. Cowan, KC, E. Gross, O. Vitells
Eur.Phys.J. C71 (2011) 1554
[arXiv:1007.1727]

THUMBNAIL OF THE STATISTICAL PROCEDURE

$$\lambda(\mu) = \frac{L(\mu, \hat{\hat{\theta}}(\mu))}{L(\hat{\mu}, \hat{\theta})}$$

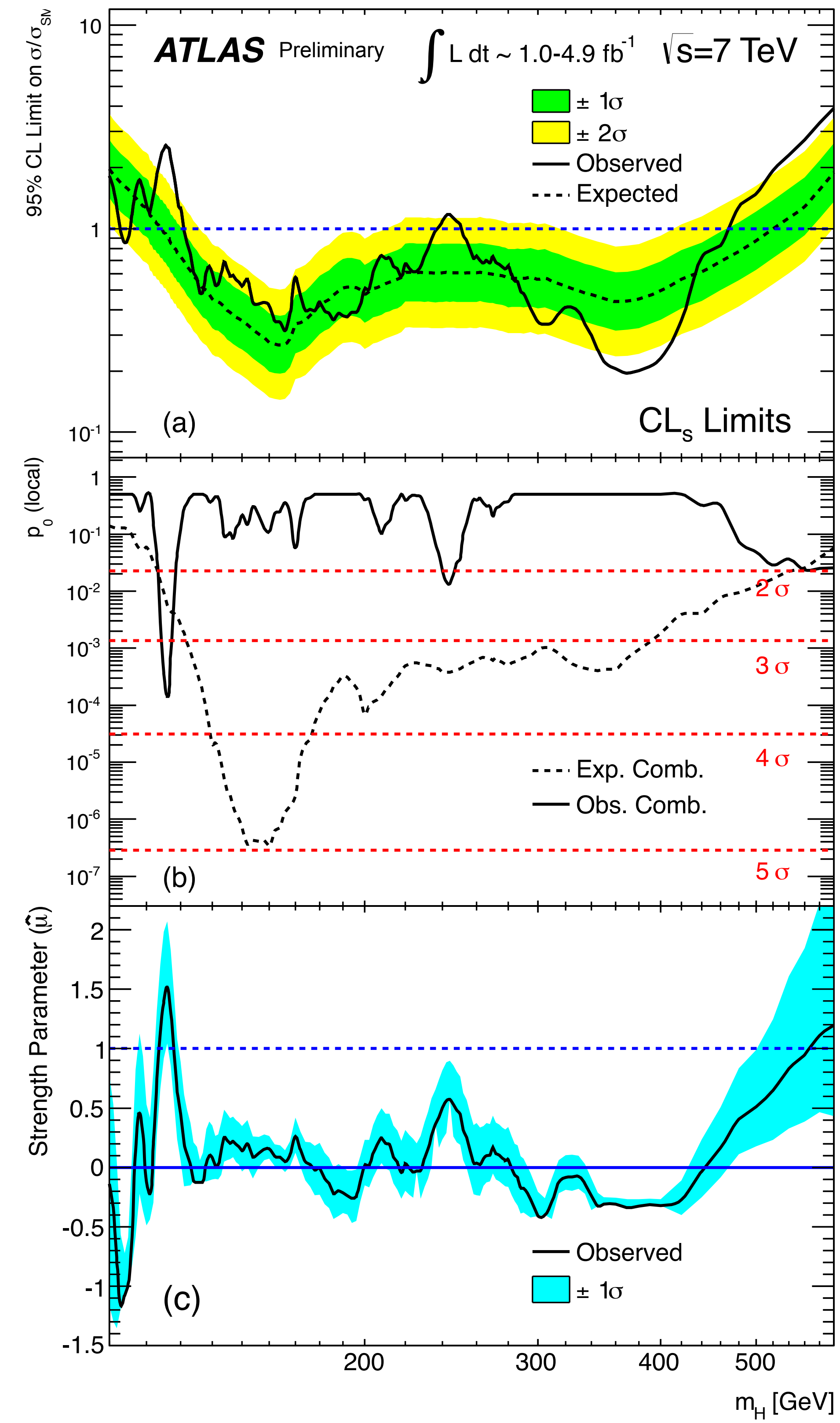
Follow LHC-HCG Combination Procedures



CL_s to test signal hypothesis

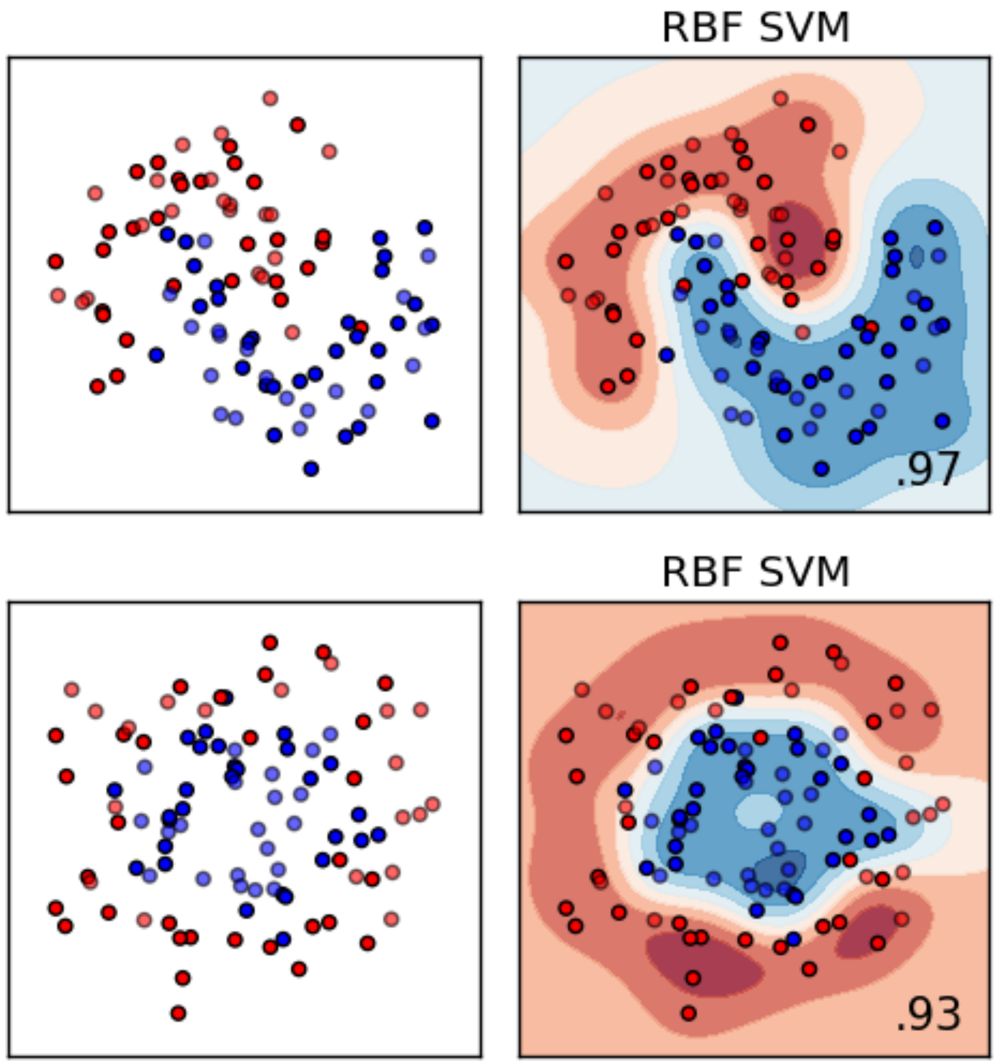
p_0 to test background hypothesis

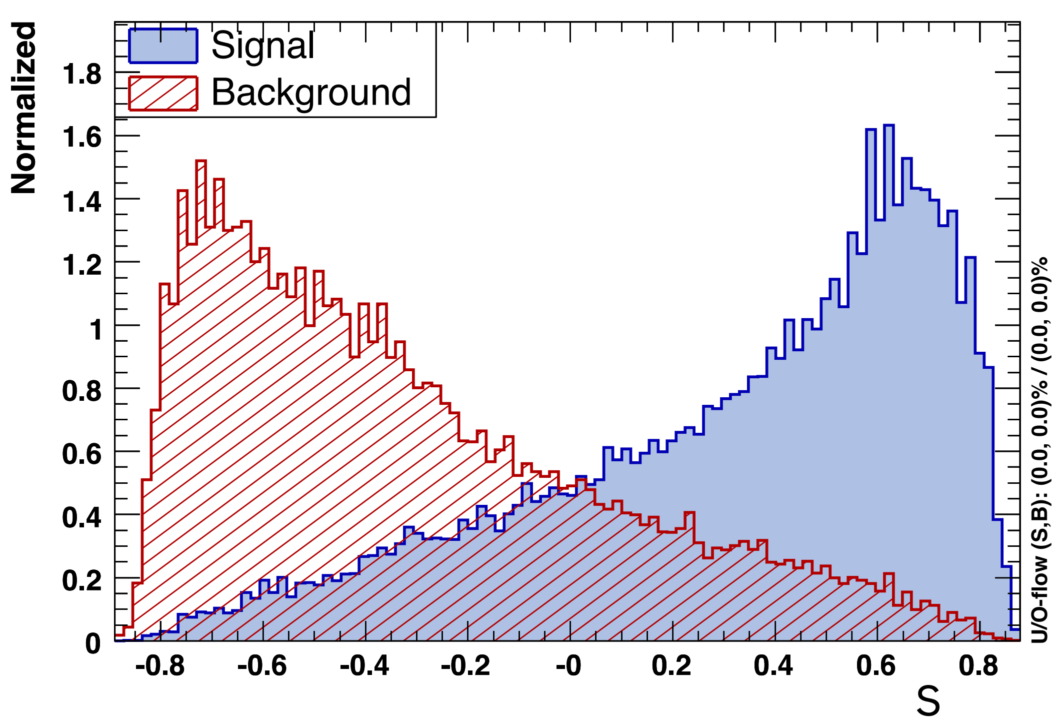
$\hat{\mu}$ to estimate signal strength



Parametrized Learning

MACHINE LEARNING: CLASSIFIERS



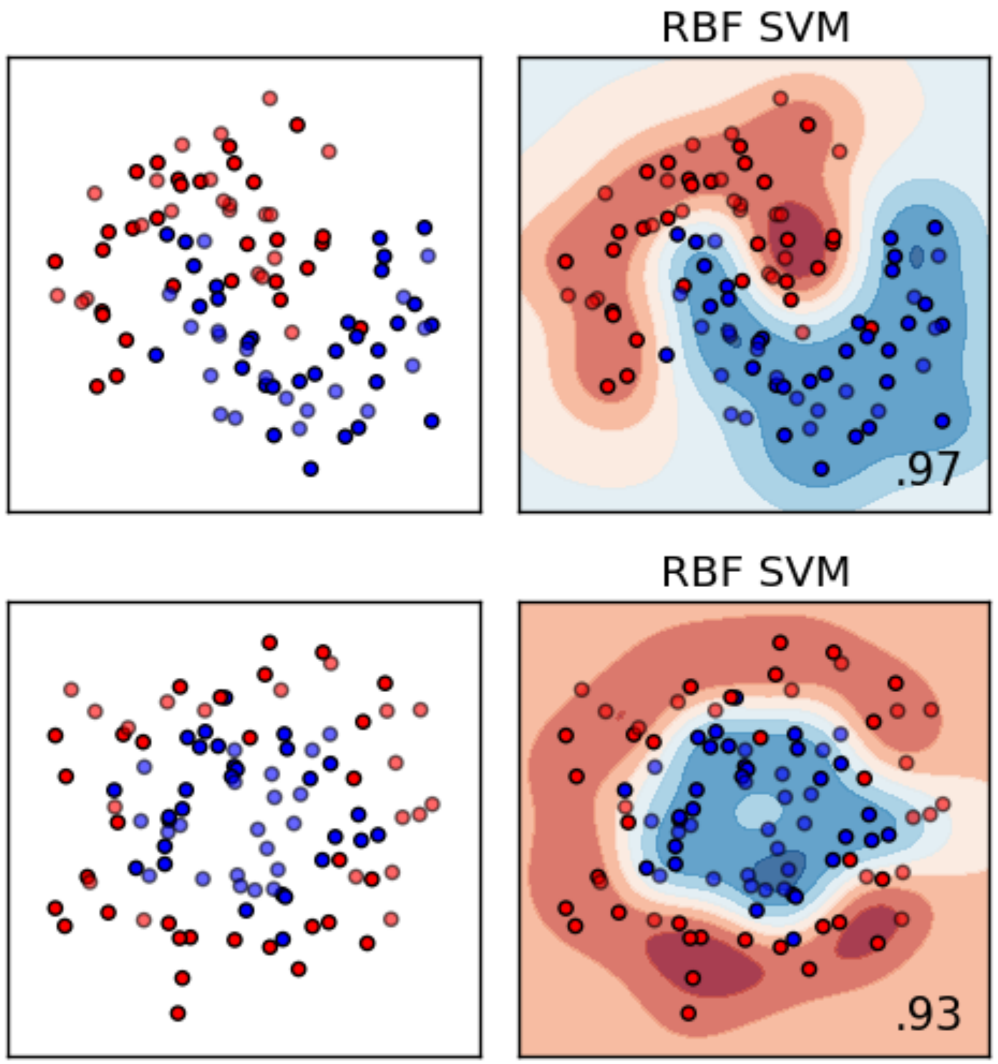


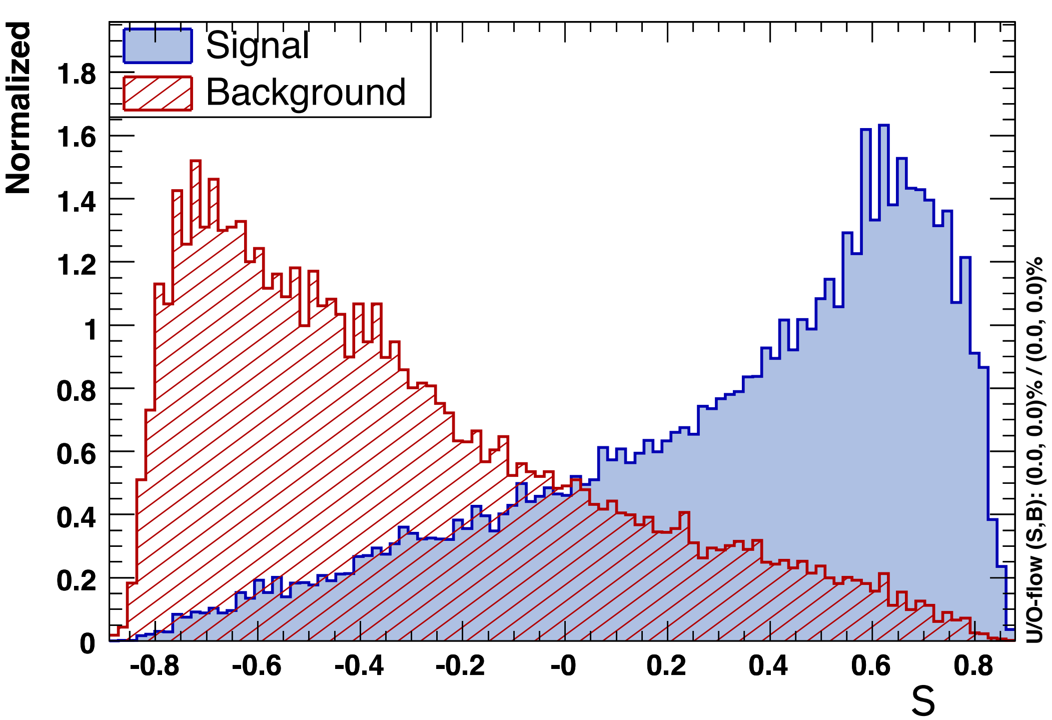
Common to use machine learning classifiers to separate signal (H_1) vs. background (H_0)

- want a function s: X→ Y that maps signal to y=1 and background to y=0
- **calculus of variations**: find function s(x) that minimizes ***loss***:

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

MACHINE LEARNING: CLASSIFIERS





- **applied calculus of variations**: find function s(x) that minimizes ***loss***:

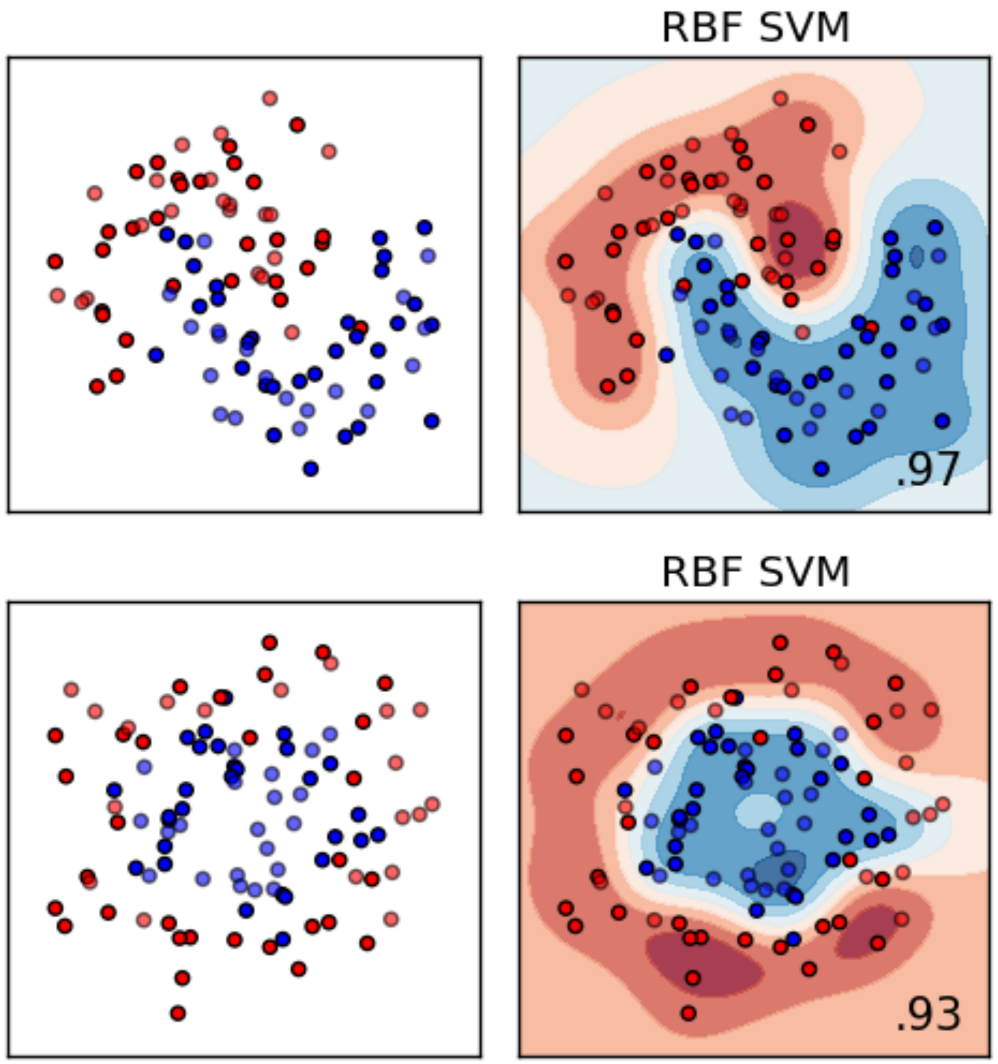
$$L[s] = \int p(x|H_0)\,(0 - s(x))^2\,dx + \int p(x|H_1)\,(1 - s(x))^2 dx$$

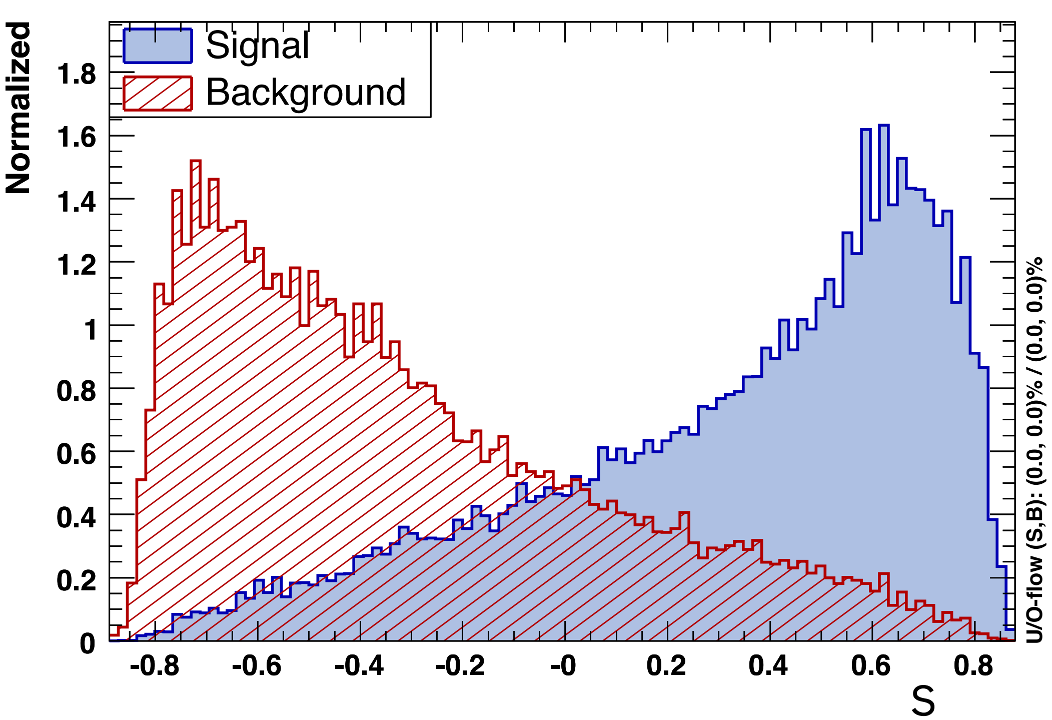
- i.e. approximate the optimal classifier

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

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

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





- **applied calculus of variations**: 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

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

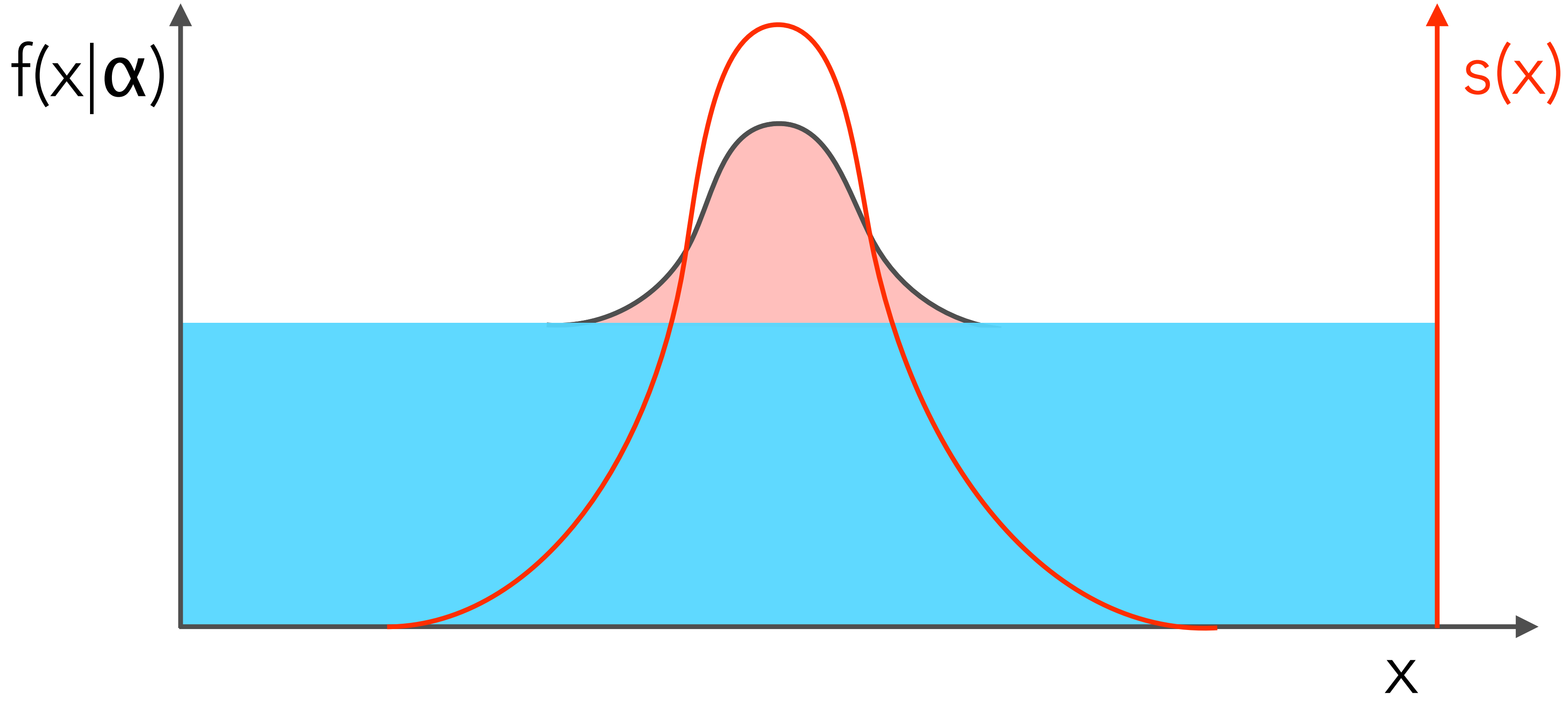
- which is 1-to-1 with the likelihood ratio

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

FIXED CLASSIFIER IS NOT OPTIMAL

Imagine a simple example of bump on flat background

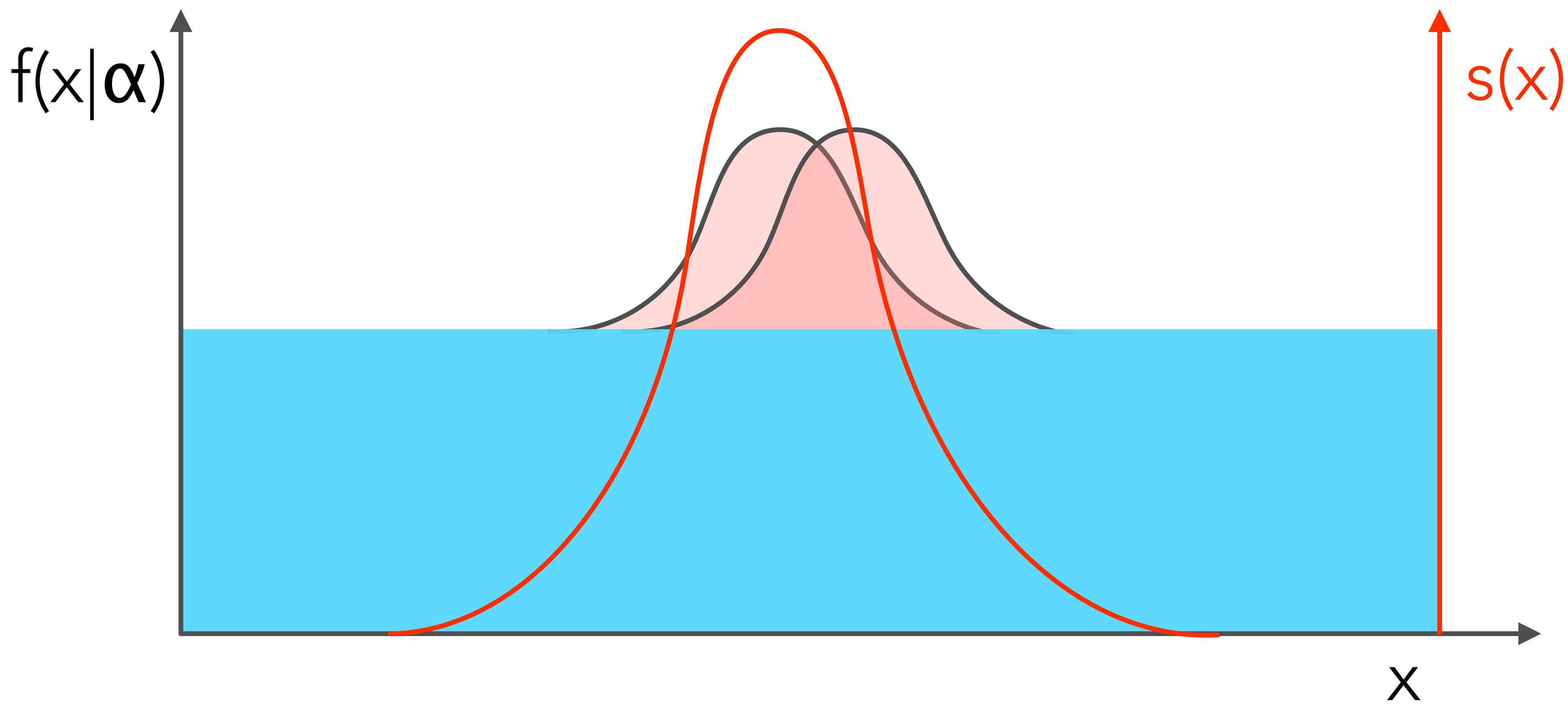
- train on samples with $\alpha=\alpha_0$ to obtain fixed classifier s(x)
- uncertainty in α modifies location and width of peak
- we can propagate the fixed learner, but classifier not optimal for $\alpha\neq\alpha_0$



FIXED CLASSIFIER IS NOT OPTIMAL

Imagine a simple example of bump on flat background

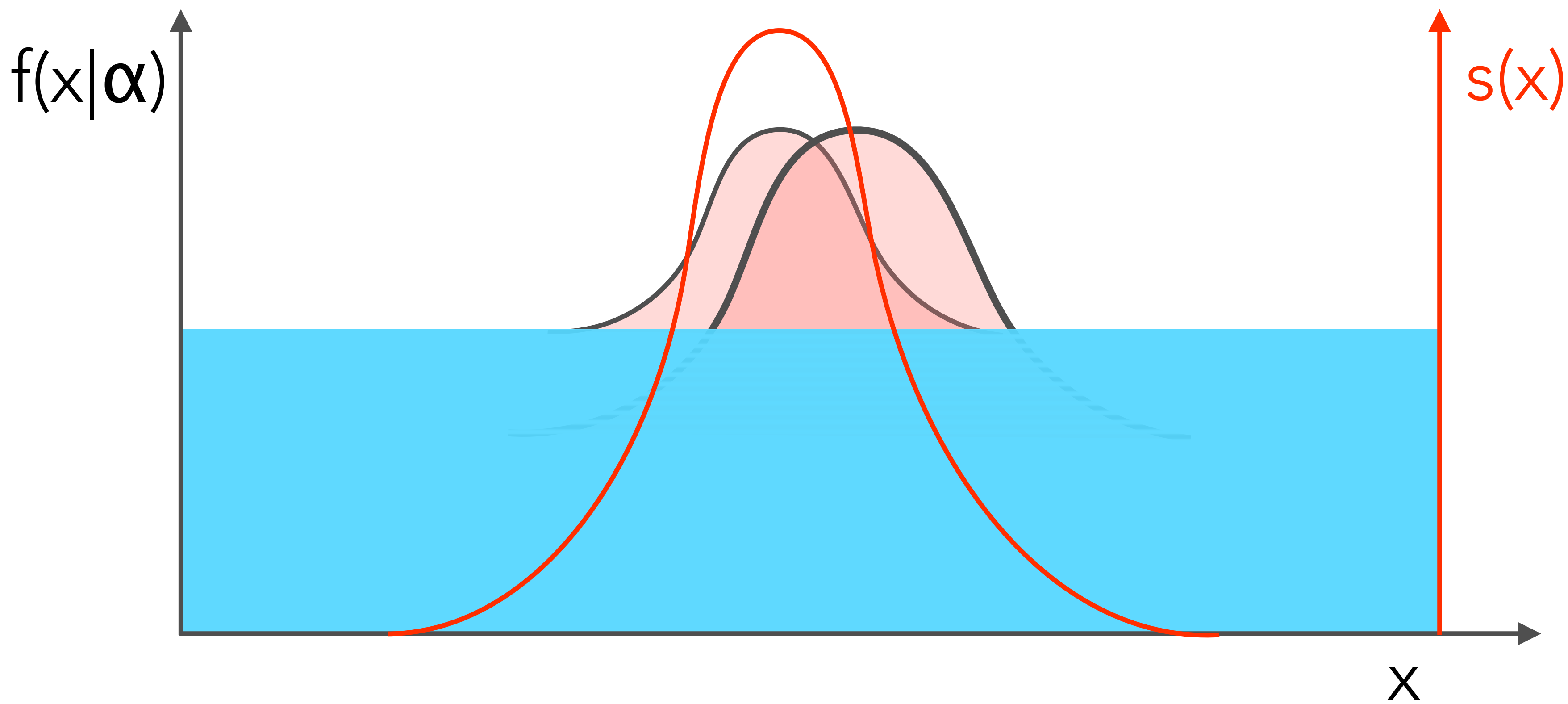
- train on samples with $\alpha=\alpha_0$ to obtain fixed classifier s(x)
- uncertainty in α modifies location and width of peak
- we can propagate the fixed learner, but classifier not optimal for $\alpha\neq\alpha_0$



FIXED CLASSIFIER IS NOT OPTIMAL

Imagine a simple example of bump on flat background

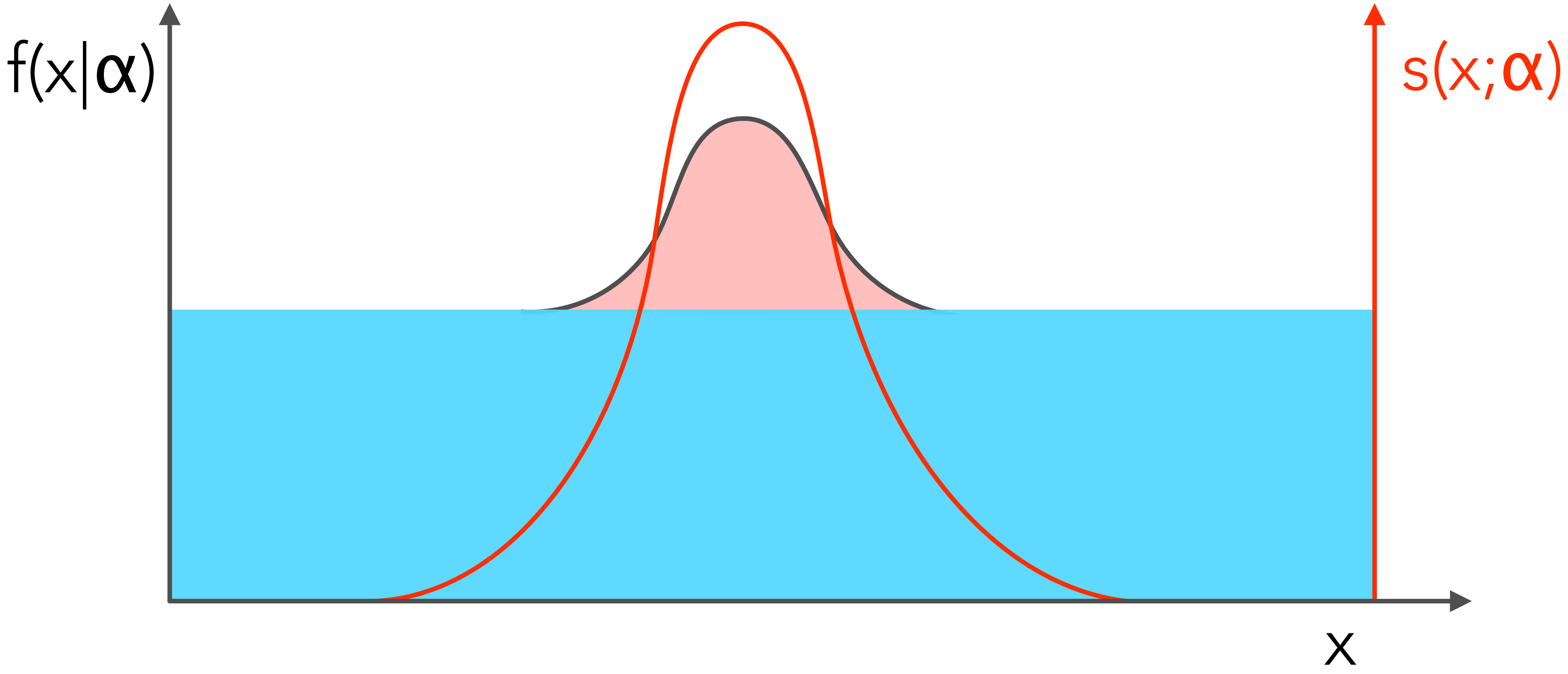
- train on samples with $\alpha=\alpha_0$ to obtain fixed classifier s(x)
- uncertainty in α modifies location and width of peak
- we can propagate the fixed learner, but classifier not optimal for $\alpha\neq\alpha_0$



A PARAMETRIZED LEARNER

We want a learner parametrized by **α**

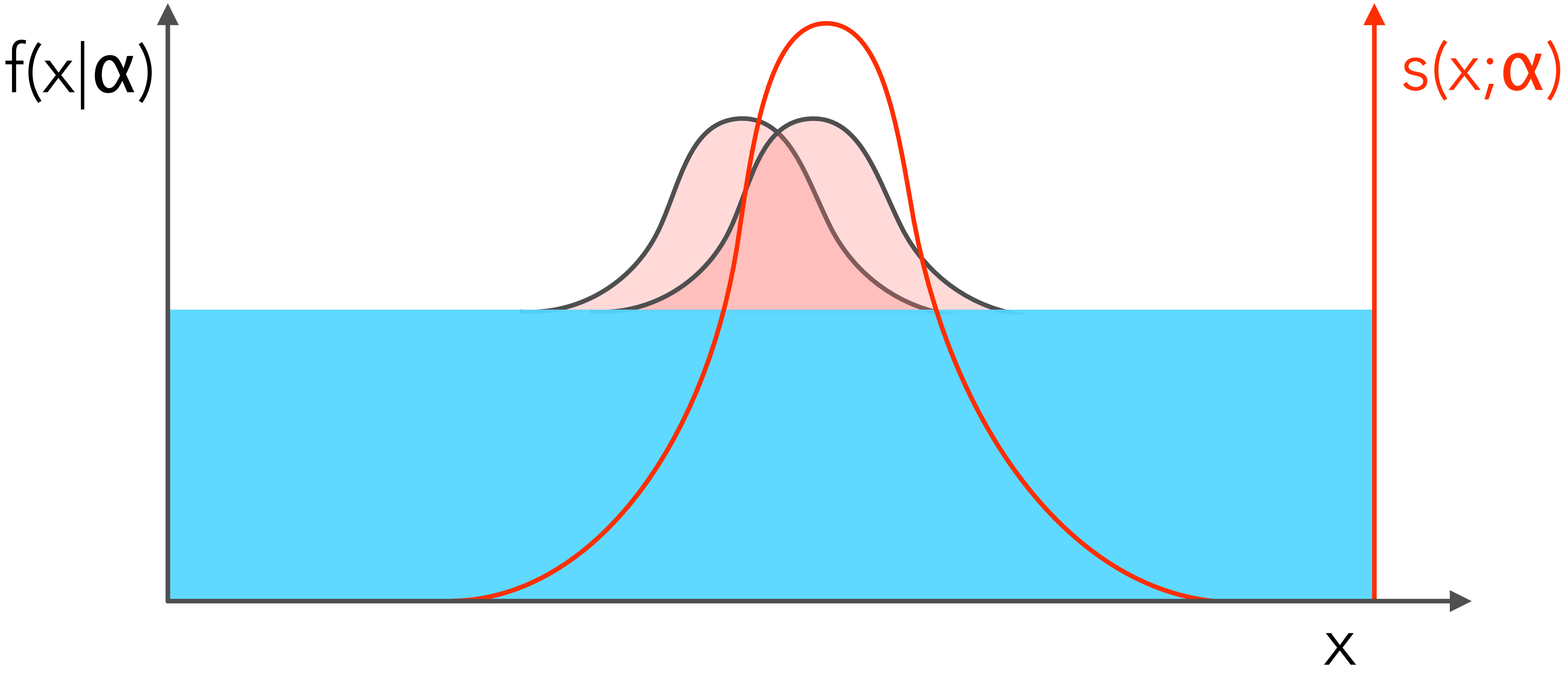
- augment training data (x,c) → (x,**α**,c) to obtain s(x;**α**)



- **problem**: how do we evaluate on testing data when **α** is unknown?

We want a learner parametrized by **α**

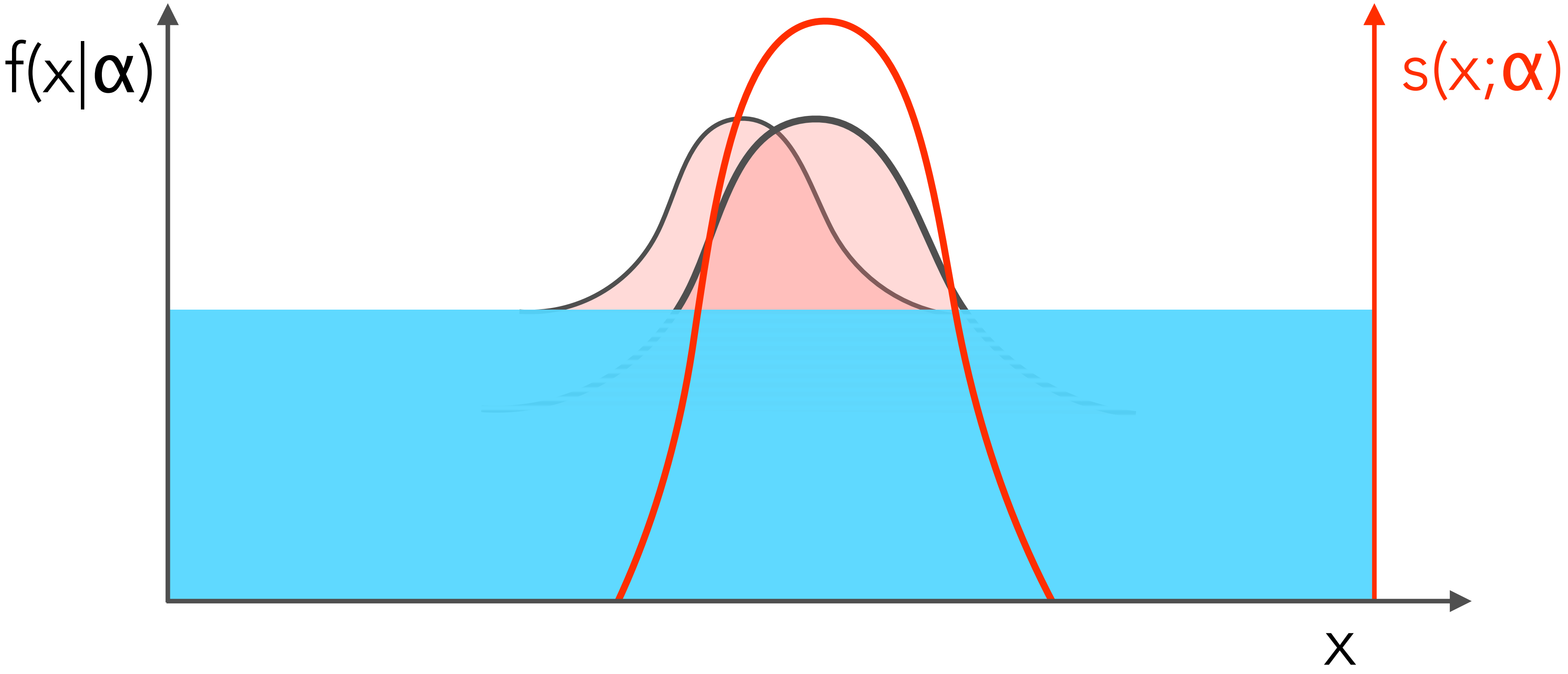
- augment training data (x,c) → (x,**α**,c) to obtain s(x;**α**)



- **problem**: how do we evaluate on testing data when **α** is unknown?

We want a learner parametrized by **α**

- augment training data (x,c) → (x,**α**,c) to obtain s(x;**α**)



- **problem**: how do we evaluate on testing data when **α** is unknown?

We started with a classifier that was learning

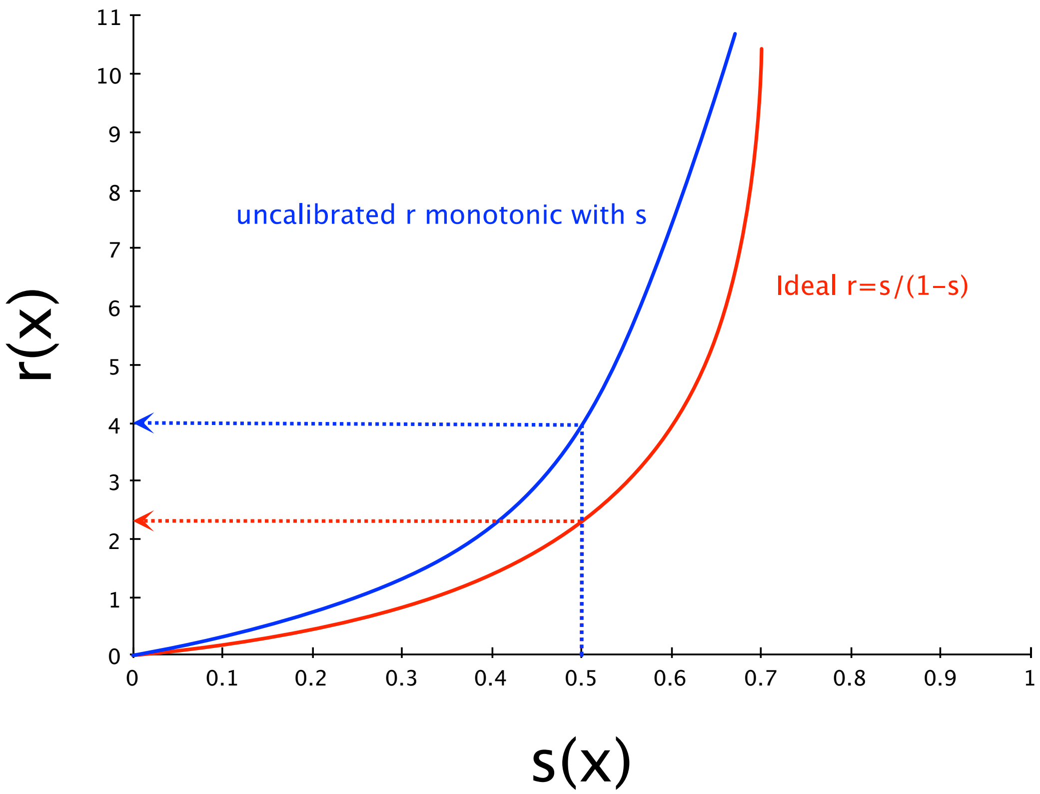
$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

Implicitly that classifier depends on H_0 and H_1 used to generate the training data. Make that explicit

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

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

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



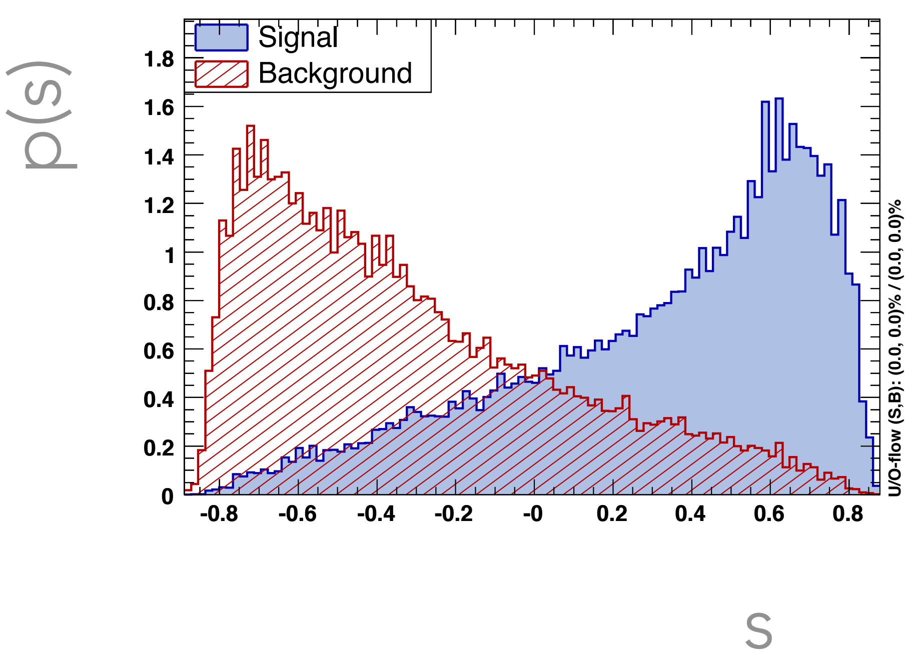
Ideally classifier will learn

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

which is 1-to-1 with the likelihood ratio

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

but often inverting s(x)→ r(x) typically doesn't work well because the classifier isn't well calibrated and learns something monotonic in r(x).



Still ok, just need to calibrate it

Theorem if s monotonic with r —>

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

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

If s(x) is monotonic with $p_1(x)/p_0(x)$, then we have

Theorem 1: We have the following equality

$$\frac{p_1(s(x))}{p_0(s(x))} = \frac{p_1(x)}{p_0(x)} \,. \tag{2.6}$$

Proof For $x \in \Omega_{s^*}$, we can factor out of the integral the constant $p_1(x)/p_0(x)$. Thus

$$p_1(s^*) = \int d\Omega_{s^*} p_1(x)/|\hat{n} \cdot \nabla s| = \frac{p_1(x)}{p_0(x)} \int d\Omega_{s^*} p_0(x)/|\hat{n} \cdot \nabla s| \,, \tag{2.7}$$

and the integrals cancel in the likelihood ratio

$$\frac{p_1(s^*)}{p_0(s^*)} = \frac{p_1(x)}{p_0(x)} \frac{\int d\Omega_{s^*} p_0(x)/|\hat{n} \cdot \nabla s|}{\int d\Omega_{s^*} p_0(x)/|\hat{n} \cdot \nabla s|} = \frac{p_1(x)}{p_0(x)} \qquad \forall x \in \Omega_{s^*} \,. \tag{2.8}$$

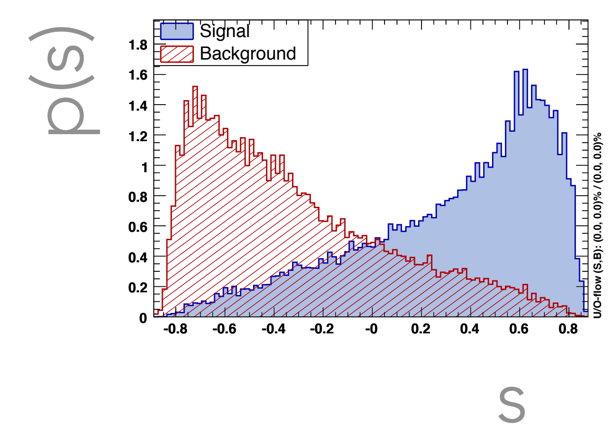
One can think of the ratio $p_1(s)/p_0(s)$ as a way of calibrating the the discriminative classifier and correcting for the monotonic transformation m of the desired likelihood ratio as in Eq. 1.3.

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

$$T(D;\theta_0,\theta_1) = \prod_{e=1}^{n} \frac{p(x_e|\theta_0)}{p(x_e|\theta_1)}$$

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

$$T(D;\theta_0,\theta_1) = \prod_{e} \frac{p(x_e|\theta_0)}{p(x_e|\theta_1)} = \prod_{e} \frac{p(s(x_e;\theta_0,\theta_1)|\theta_0)}{p(s(x_e;\theta_0,\theta_1)|\theta_1)}$$



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

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

Remember that a **classifier** that minimizes squared loss ∑ [y_i - $s(x_i)$]^2 approximates the regression function, which has the same level sets!

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

Now we can go beyond classification, and estimate parameters of theory and confidence intervals

Denote the maximum likelihood estimator

$$\hat{\theta} = \arg\max_{\theta} p(D|\theta) \tag{4.2}$$

The denominator in the likelihood ratio is just a constant

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

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.

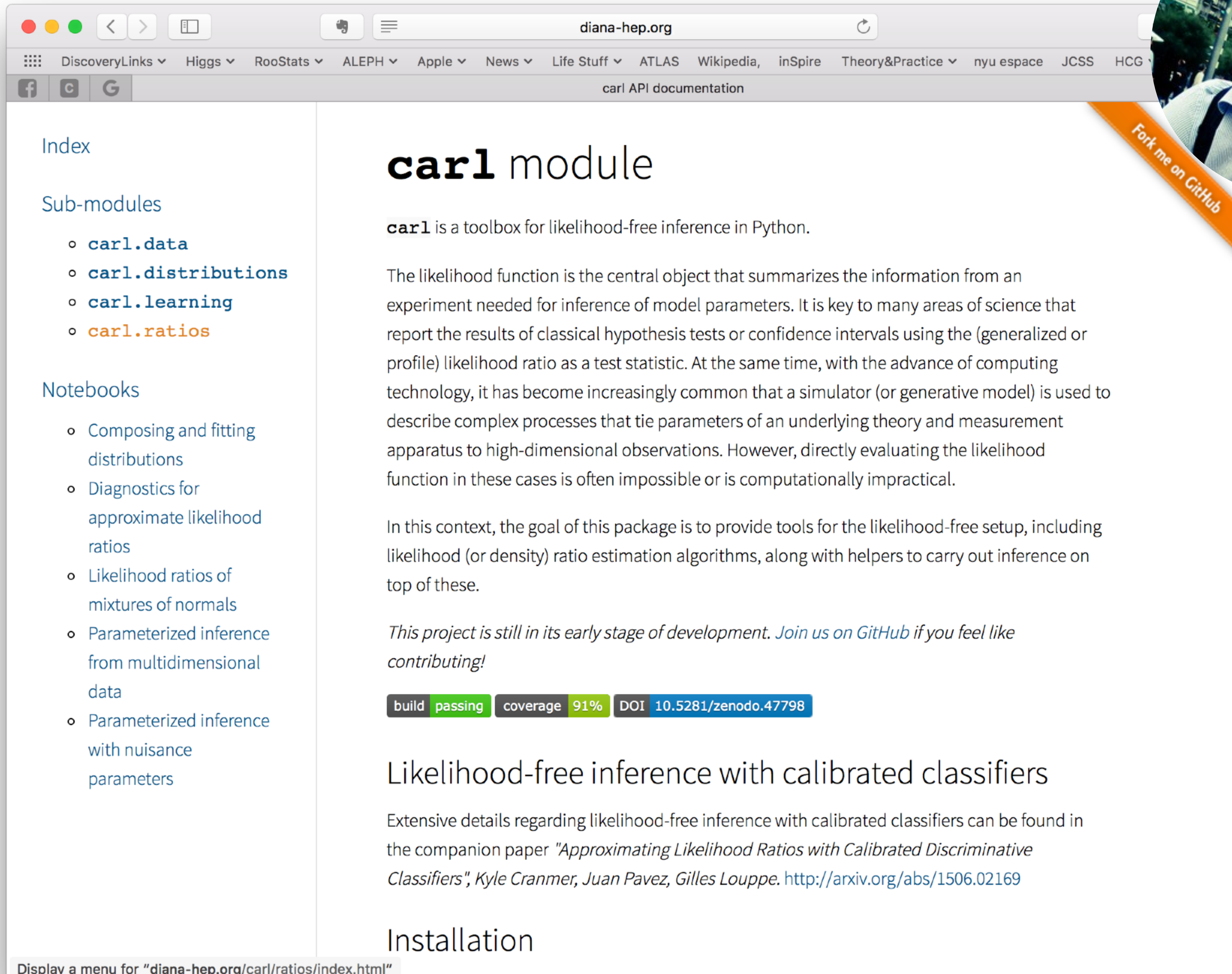
Provides a non-trivial diagnostic:

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

SOFTWARE

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

Gilles Loupper


diana-hep.org

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

carl API documentation

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

`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

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

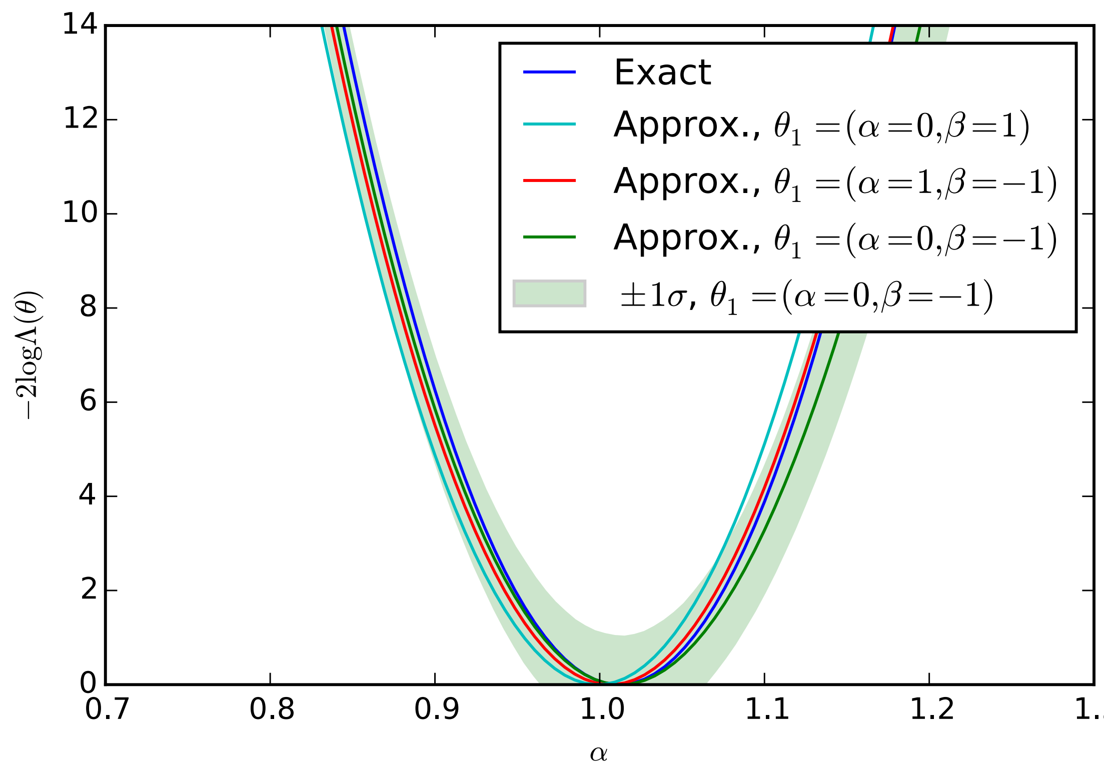
Installation

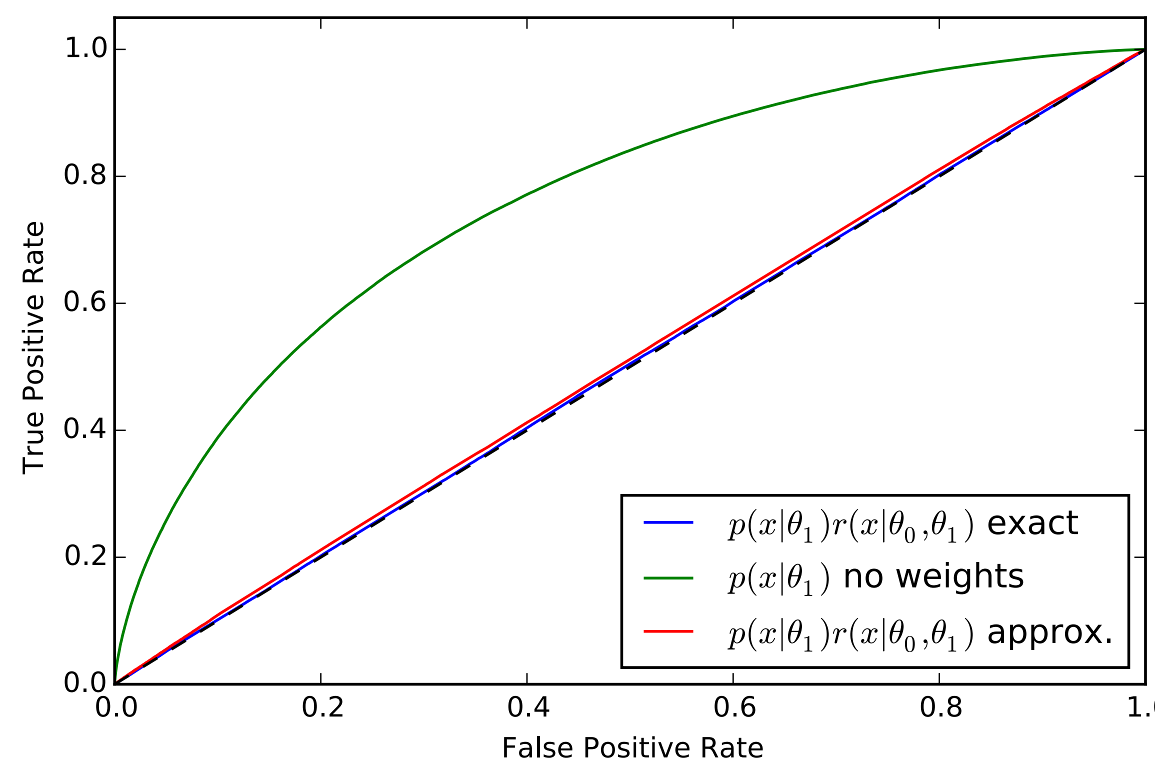
Fork me on GitHub

Display a menu for "diana-hep.org/carl/ratios/index.html"

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.

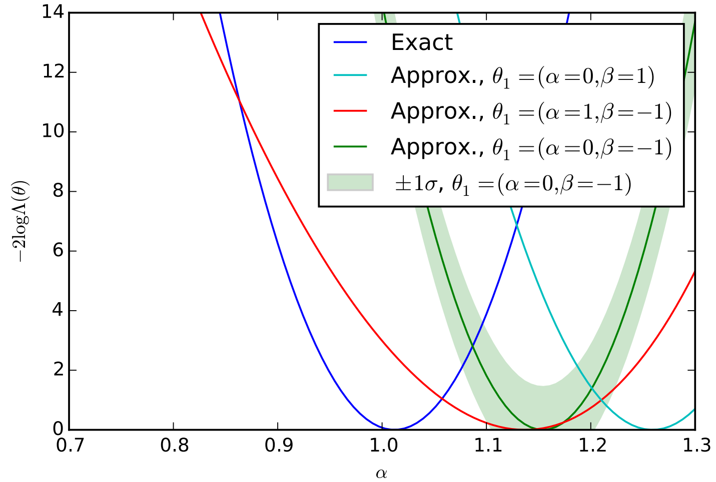
1. For inference, the value of the MLE $\hat{\theta}$ should be independent of the value of θ_1 used in the denominator of the ratio.
2. Train a classifier to distinguish between unweighted samples from $p(\mathbf{x}|\theta_0)$ and samples from $p(\mathbf{x}|\theta_1)$ weighted by $\hat{r}(\hat{s}(\mathbf{x};\theta_0,\theta_1))$.



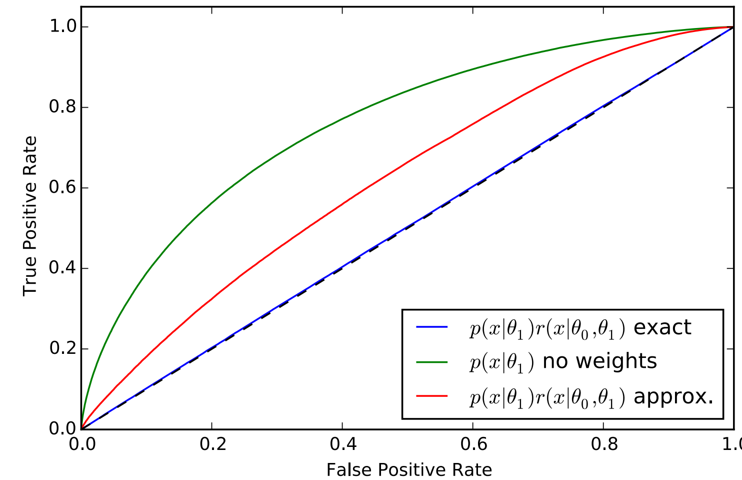


$$\frac{p_1(s^*)}{p_0(s^*)} = \frac{p_1(x)}{p_0(x)} \boxed{\frac{\int d\Omega_{s^*} p_0(x)/|\hat{n}\cdot\nabla s|}{\int d\Omega_{s^*} p_0(x)/|\hat{n}\cdot\nabla s|}} = \frac{p_1(x)}{p_0(x)} = r(x)$$

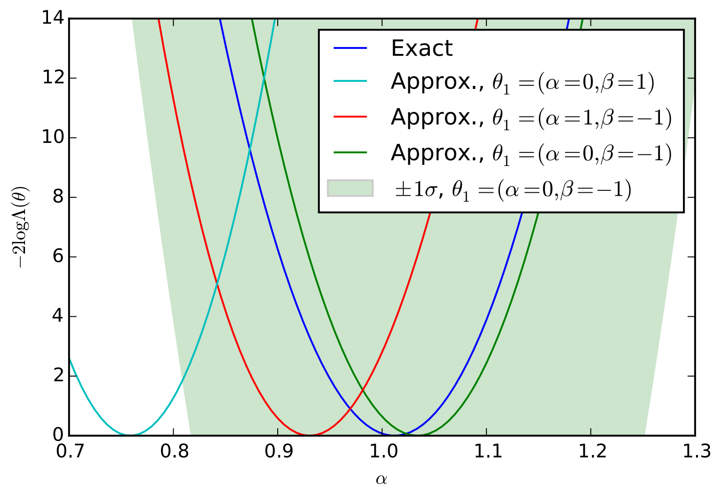
DIAGNOSTICS



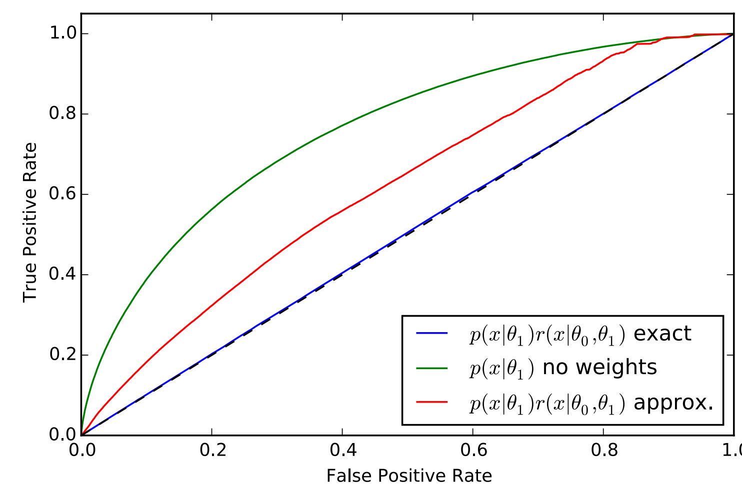
(a) Poorly trained, well calibrated.



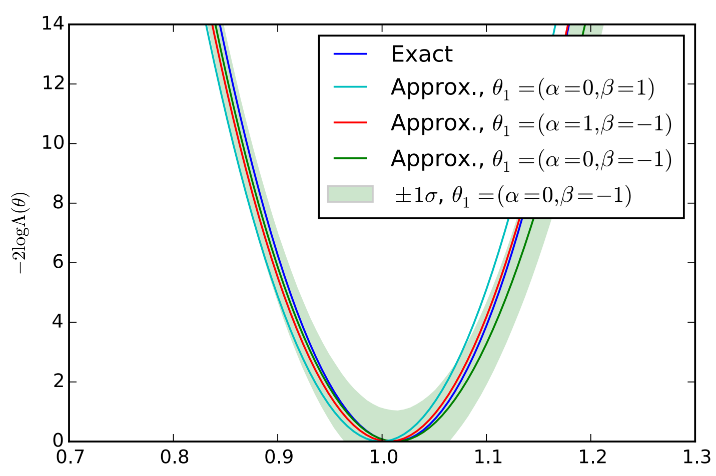
(b) Poorly trained, well calibrated.



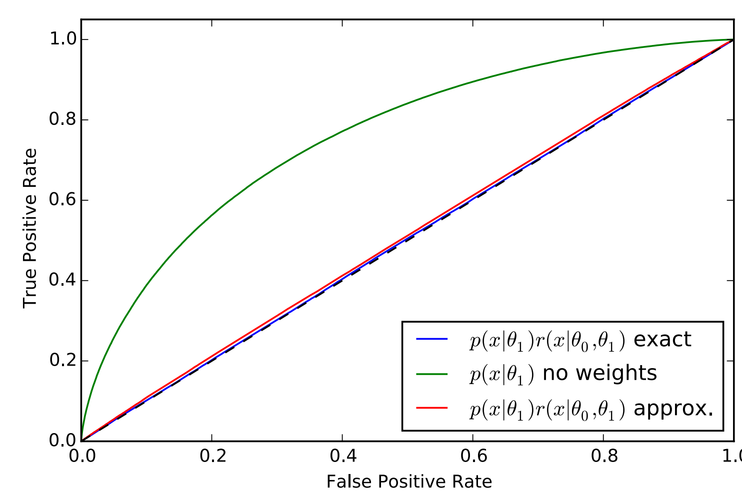
(c) Poorly calibrated, well trained.



(d) Poorly calibrated, well trained.



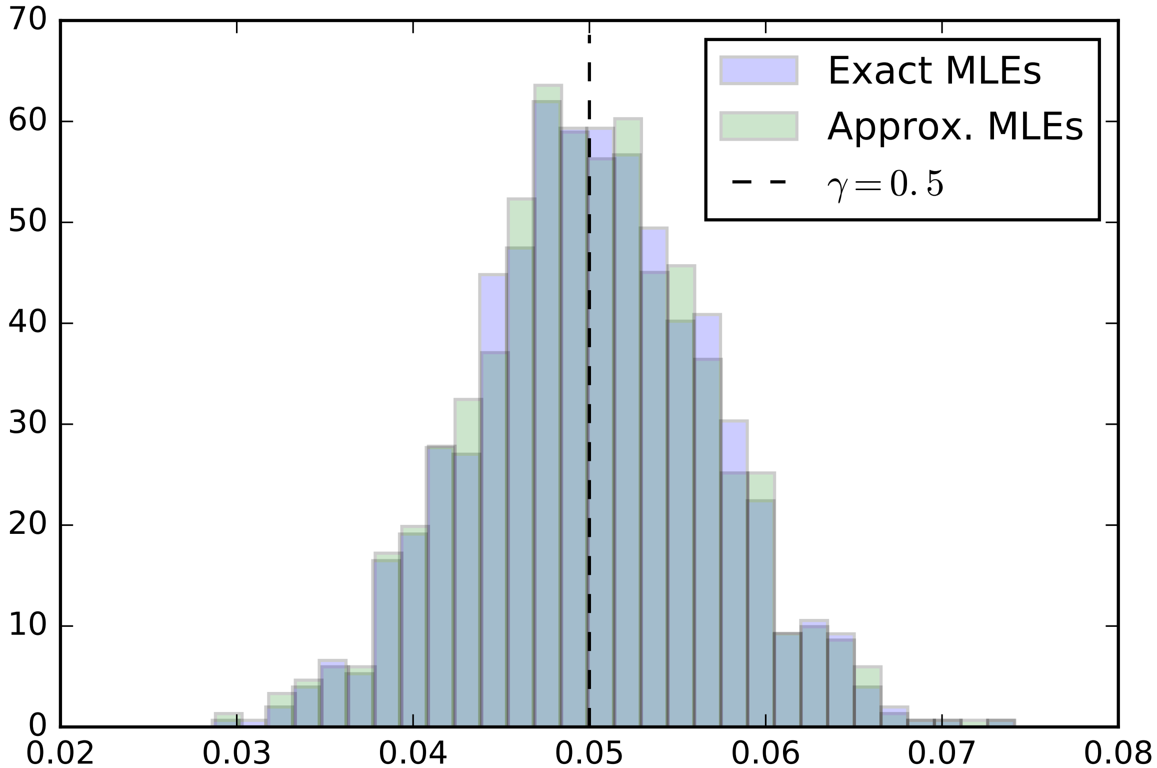
(e) Well trained, well calibrated.



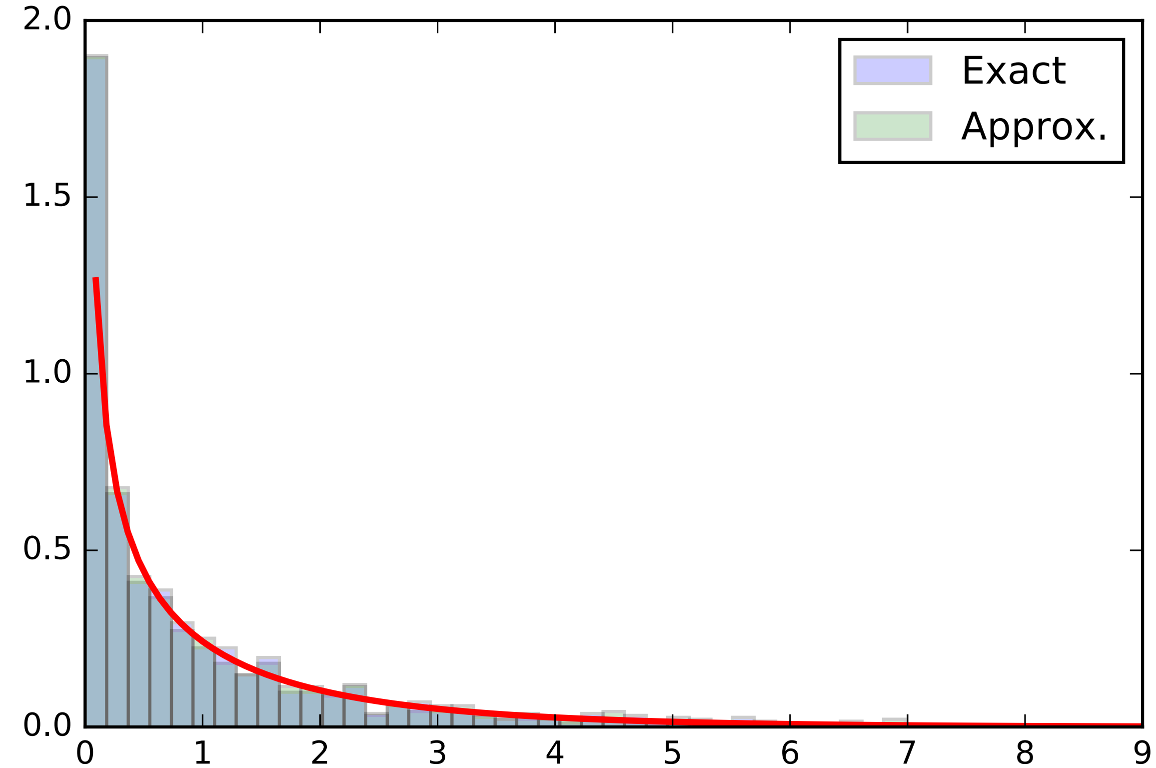
(f) Well trained, well calibrated.

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

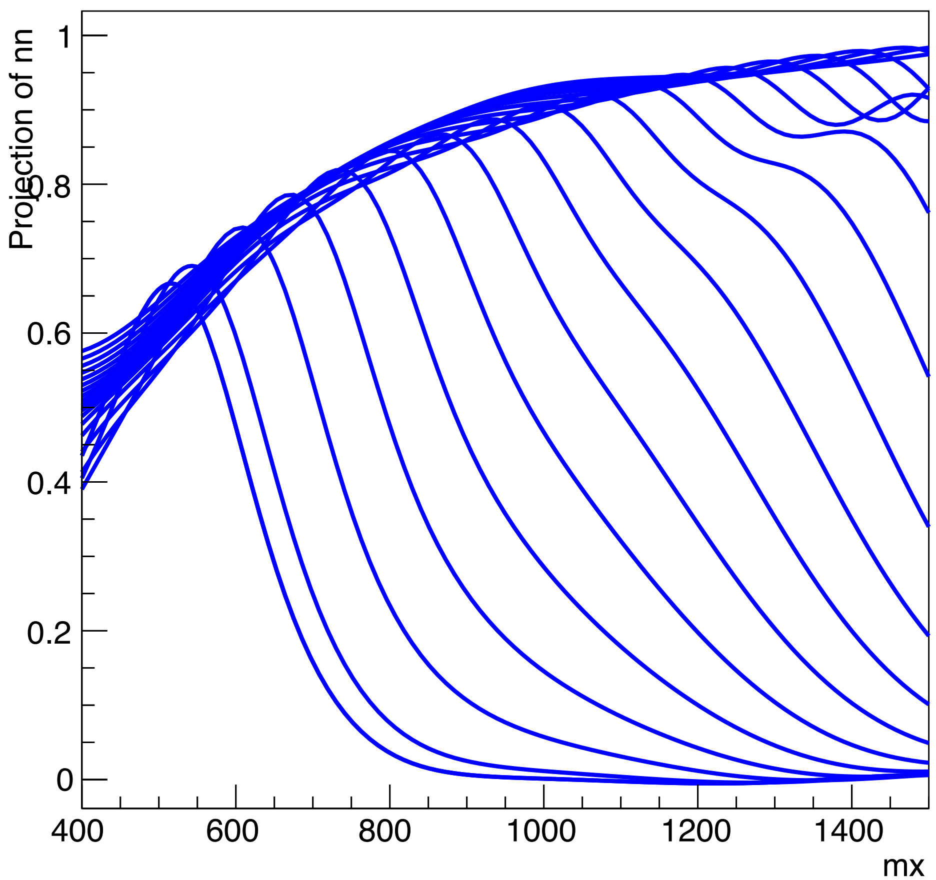
K.C., G. Louppe, J. Pavez: http://arxiv.org/abs/1506.02169

WRAPPING SKLEARN, THEANO, XGBOOST, …

https://github.com/cranmer/roofit-python-wrapper

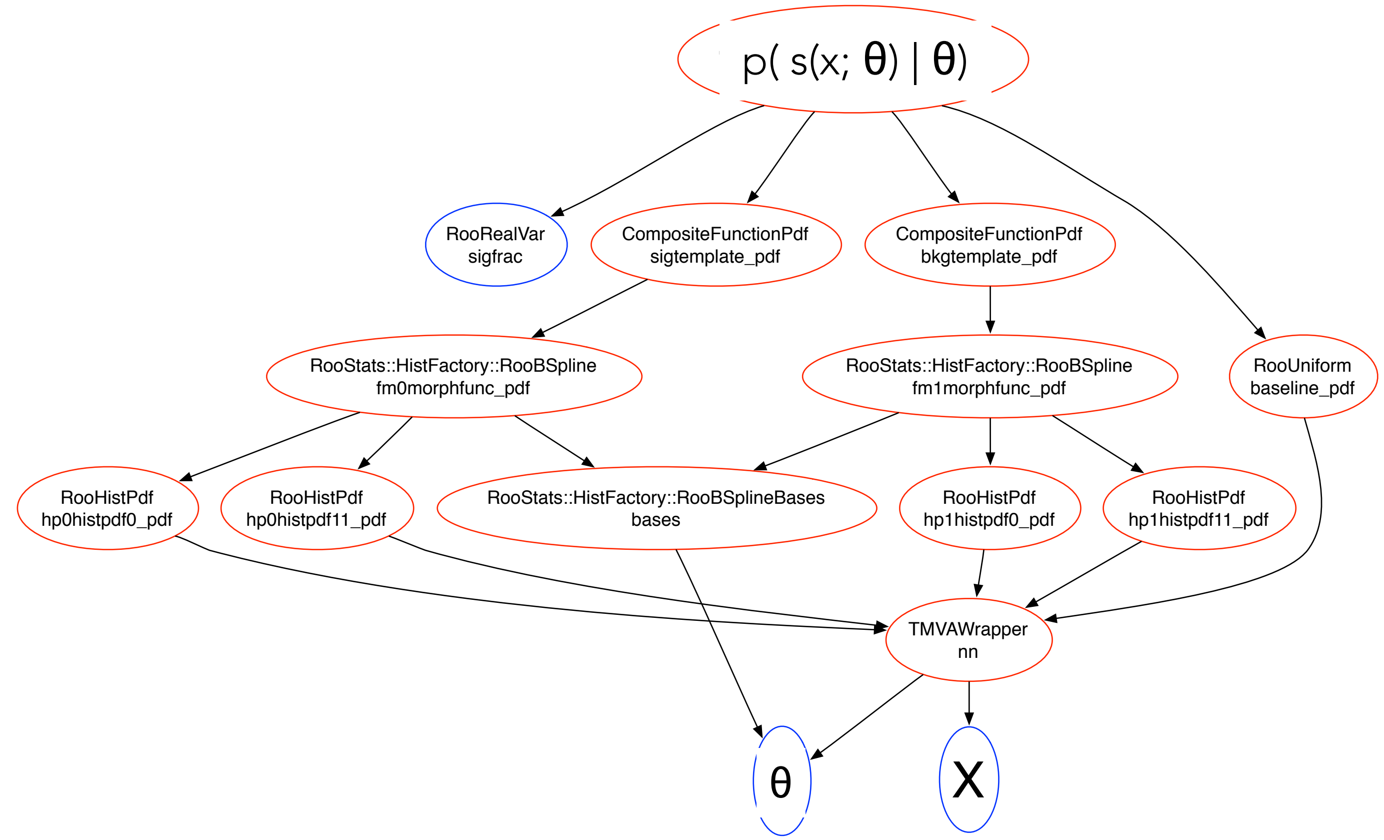
```
from ROOT import *
import numpy as np
from sklearn import svm
from sklearn.externals import joblib

def scikitlearnFunc(x=0.):
    clf = joblib.load('../adaptive.pkl')
    traindata = np.array((x,0.))
    outputs=clf.predict(traindata)
    return outputs[0]

def scikitlearnTest():
    gSystem.Load( 'libSciKitLearnWrapper' )
    x = RooRealVar('x','x',0.2,-5,5)
    s = SciKitLearnWrapper('s','s',x)
    s.RegisterCallBack( scikitlearnFunc );

    c1 = TCanvas('c1')
    frame = x.frame()
    s.plotOn(frame)
    frame.Draw()
    c1.SaveAs('scikitlearn-wrapper-plot.pdf')

if __name__ == '__main__':
    scikitlearnTest()
```

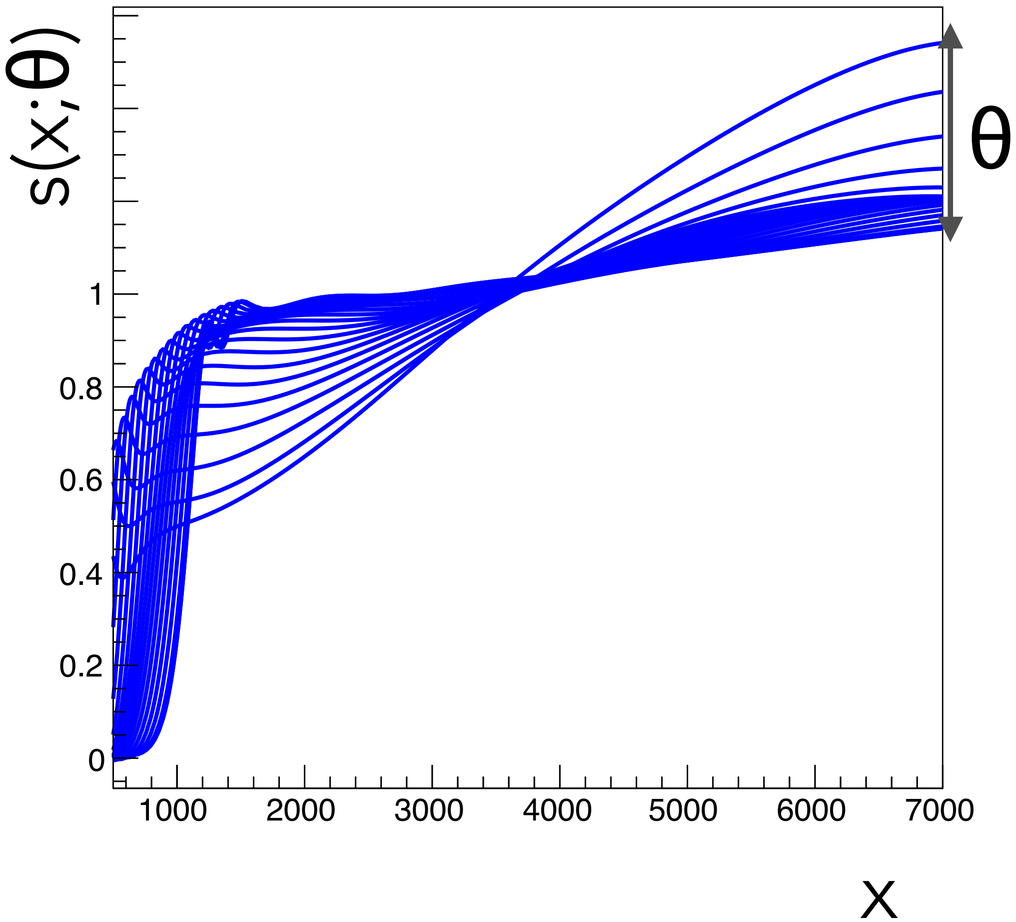
Handy utility to wrap any python function as a RooAbsReal

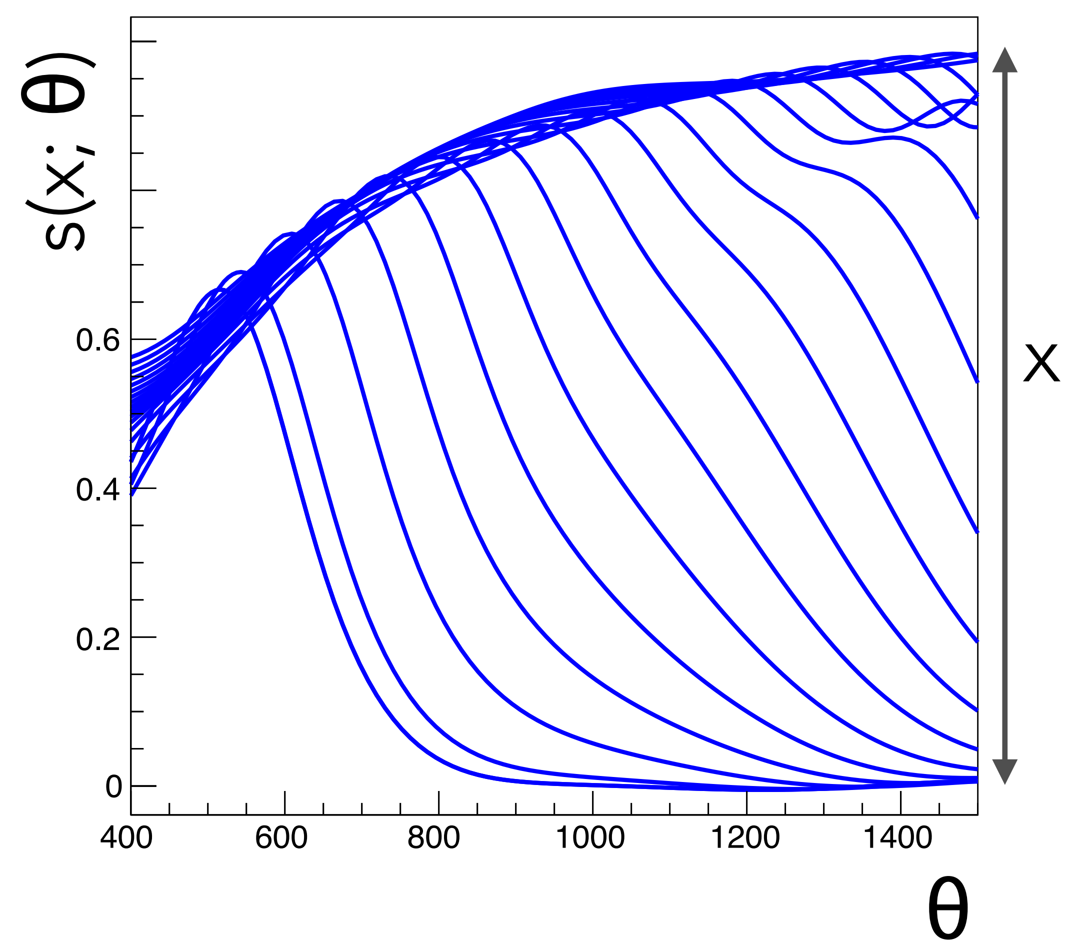


Postpone evaluation of the classifier to the time when the likelihood is evaluated and a specific value of the parameter θ is being tested

$$T(D;\theta_0,\theta_1) = \prod_e \frac{p(x_e|\theta_0)}{p(x_e|\theta_1)} = \prod_e \frac{p(s(x_e;\theta_0,\theta_1)|\theta_0)}{p(s(x_e;\theta_0,\theta_1)|\theta_1)}$$

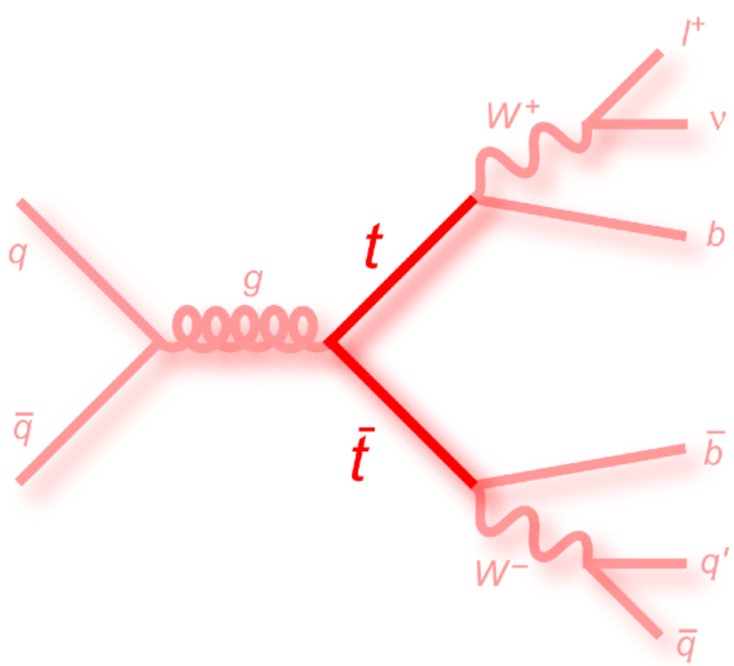






PARAMETRIZED CLASSIFIERS WITH DNN

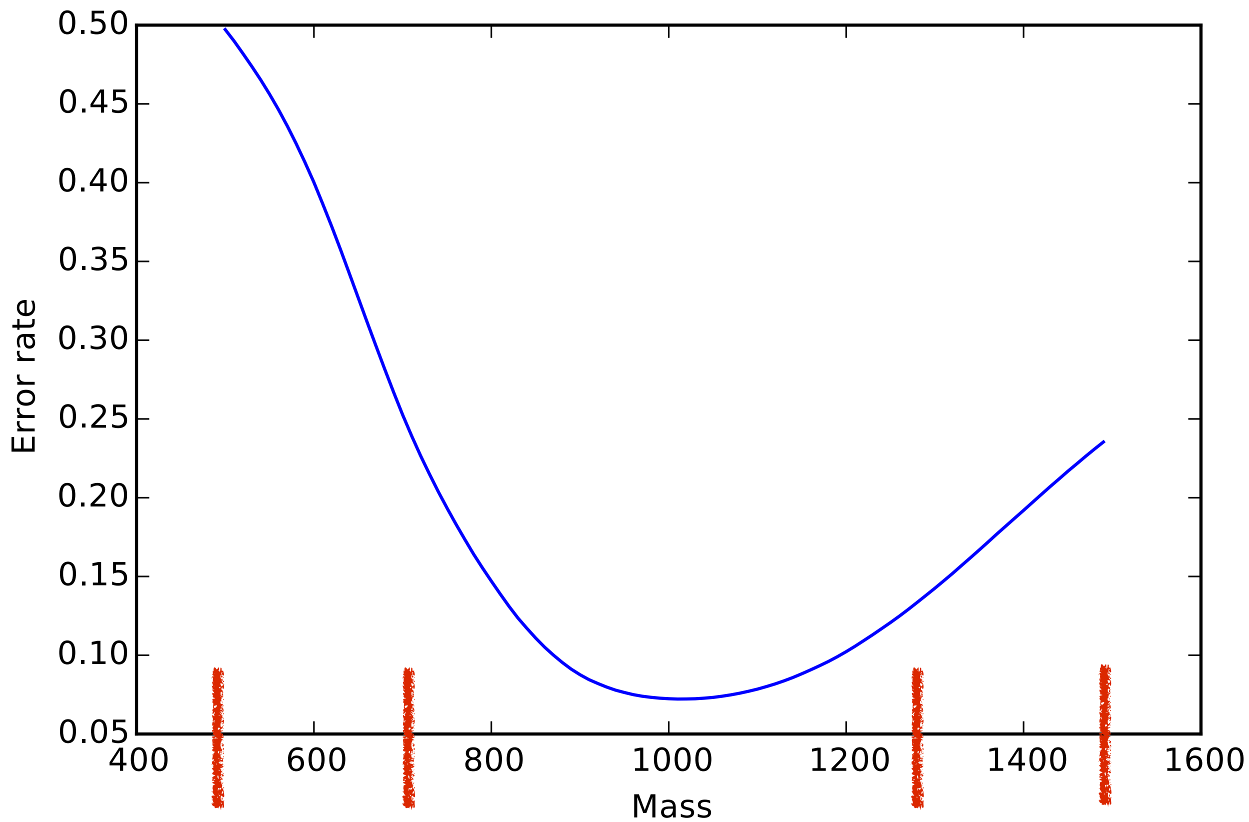
Example: $Z' \to t\bar{t}$



together with:

Peter Sadowski , Daniel Whiteson, Pierre Baldi, Taylor Faucett

The networks were trained on 28 features: 22 low-level, 5 high-level, and the mass

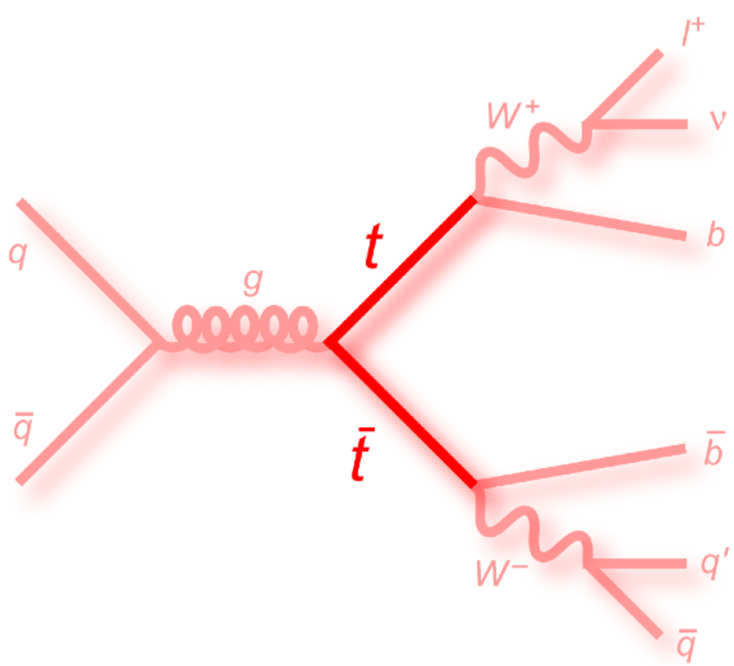


Train at $m_{Z'}$=500,750,1250,1500 GeV

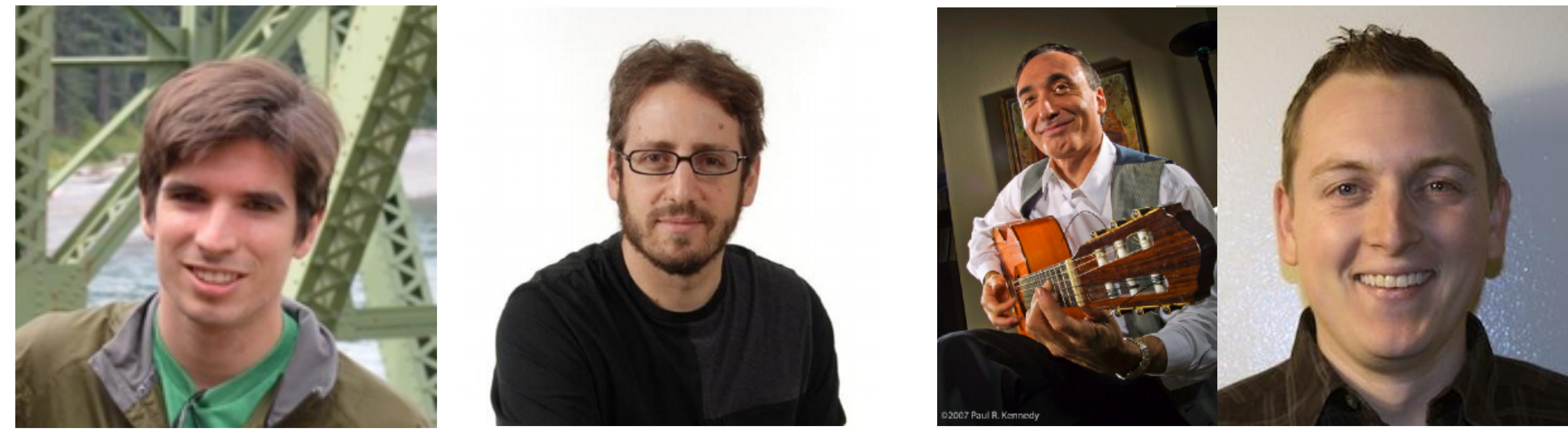
Almost identical performance to dedicated training at $m_{Z'}$=1000 GeV

PARAMETRIZED CLASSIFIERS WITH DNN

Example: $Z' \to t\bar{t}$

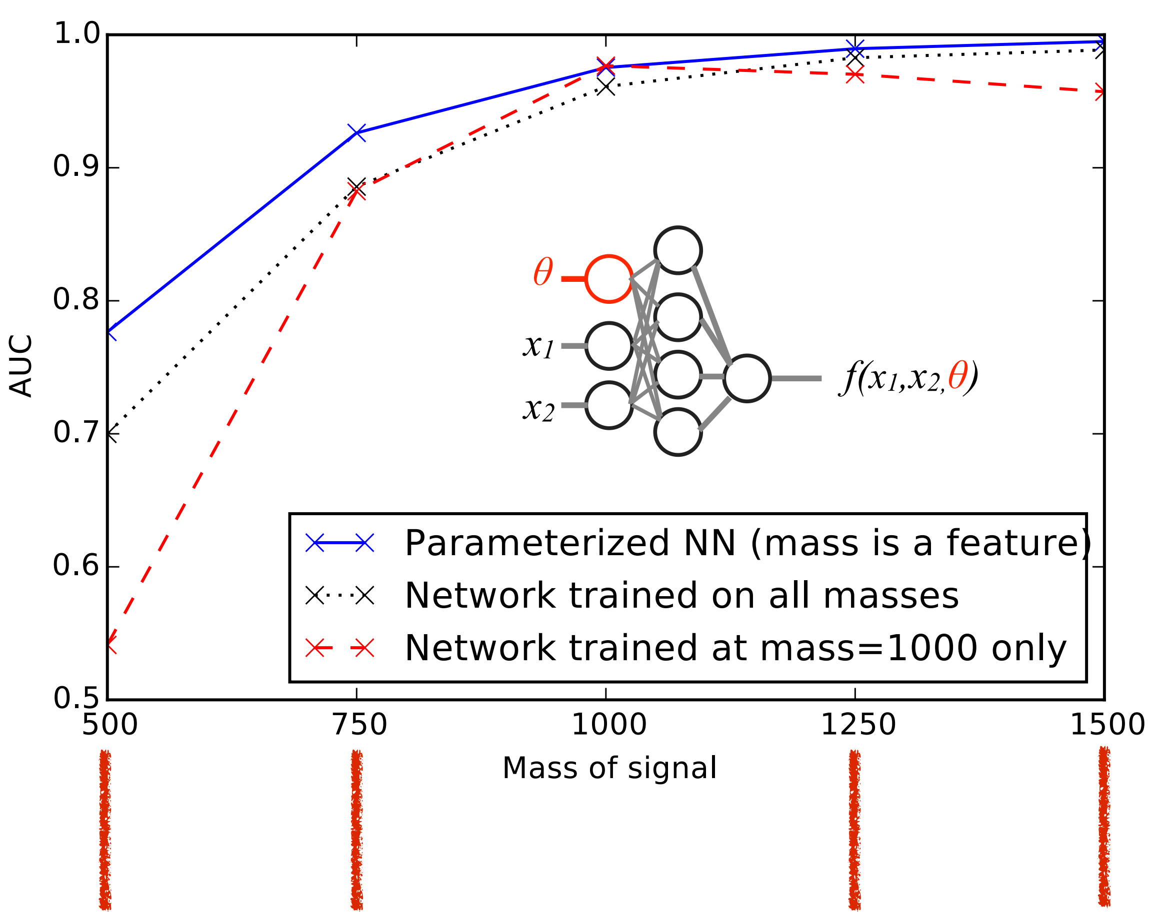


arXiv:1601.07913, together with:



Peter Sadowski , Daniel Whiteson, Pierre Baldi, Taylor Faucett

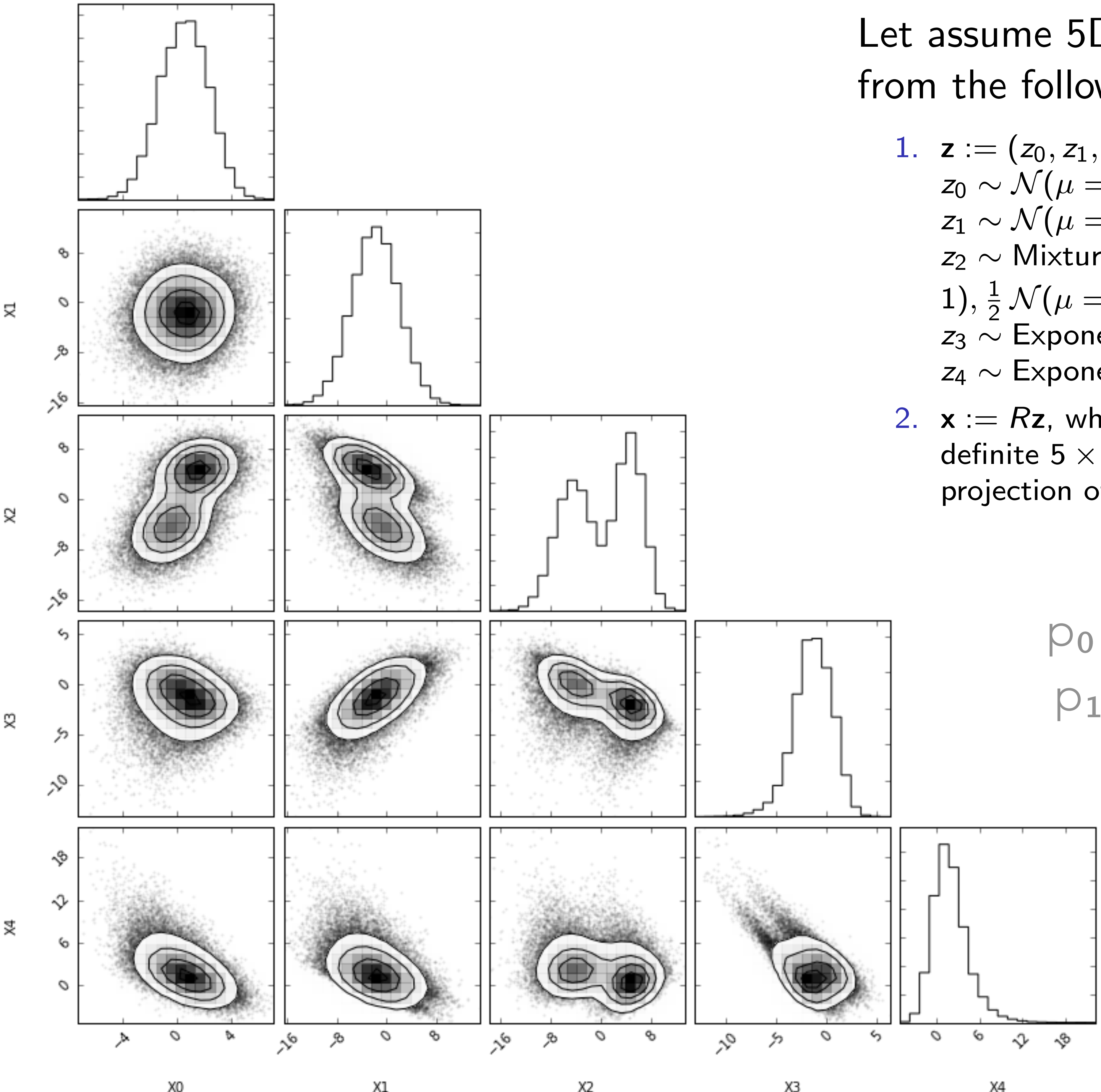
The networks were trained on 28 features: 22 low-level, 5 high-level, and the mass



Train at $m_{Z'}$=500,750,1250,1500 GeV

Almost identical performance to dedicated training at $m_{Z'}$=1000 GeV

THE DATA

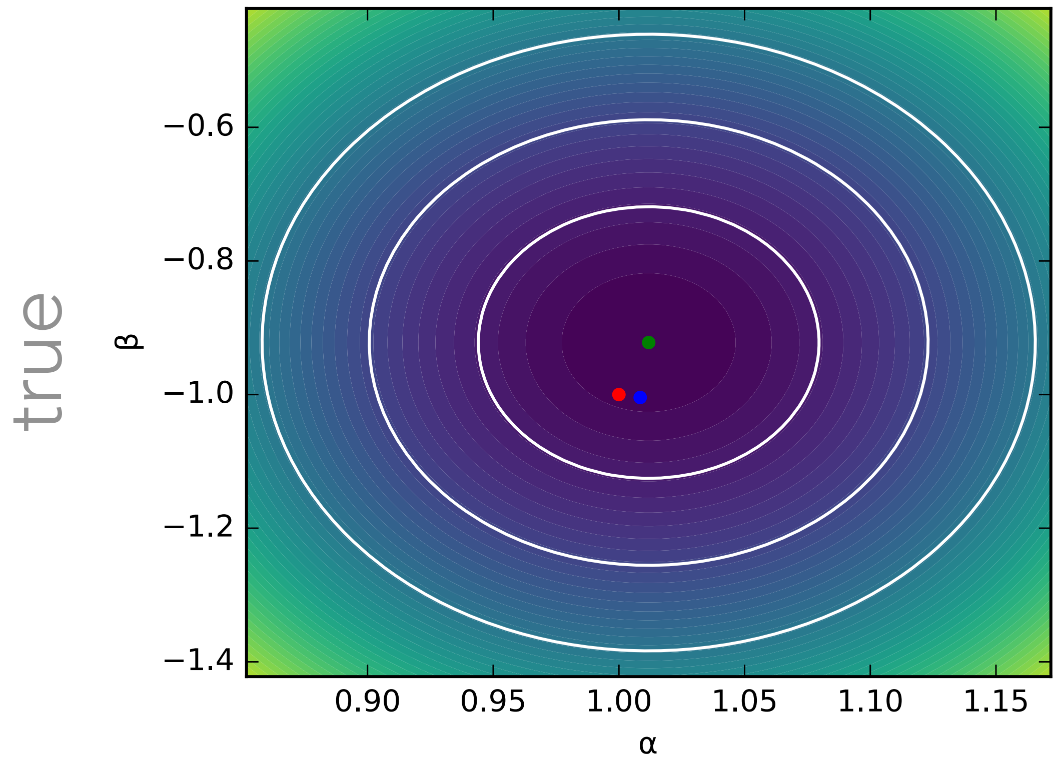


Let assume 5D data $\mathbf{x}$ generated from the following process p_0:

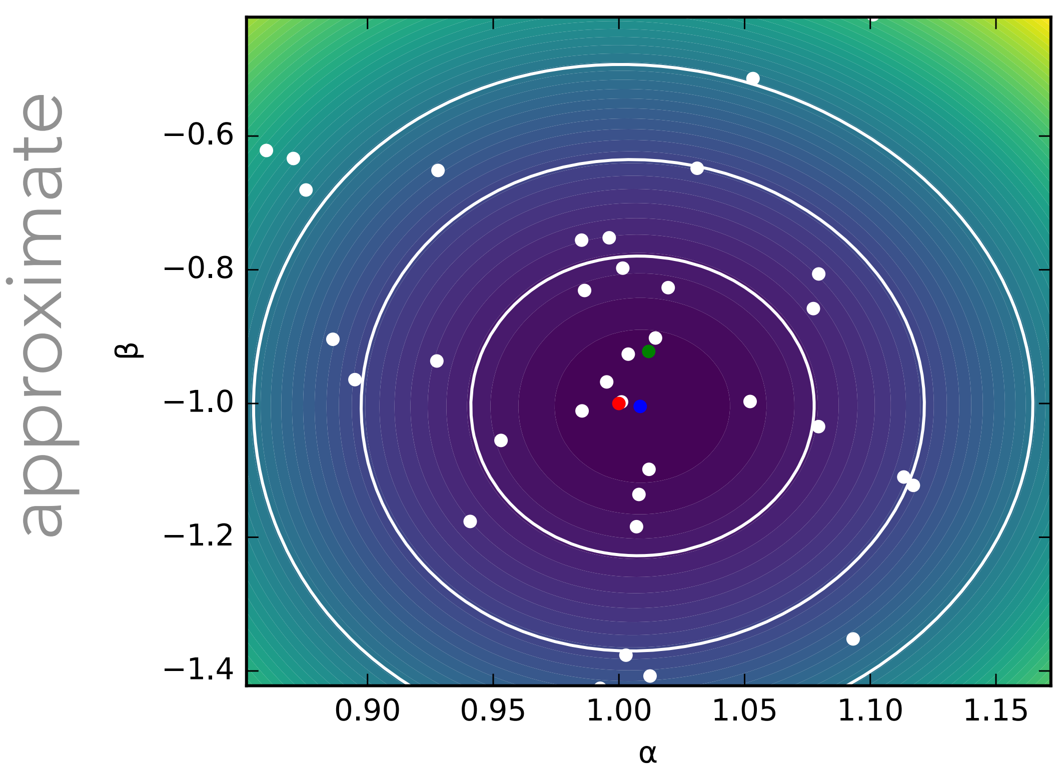
1. $\mathbf{z} := (z_0, z_1, z_2, z_3, z_4)$, such that $z_0 \sim \mathcal{N}(\mu = \alpha, \sigma = 1)$, $z_1 \sim \mathcal{N}(\mu = \beta, \sigma = 3)$, $z_2 \sim \text{Mixture}(\frac{1}{2}\mathcal{N}(\mu = -2, \sigma = 1), \frac{1}{2}\mathcal{N}(\mu = 2, \sigma = 0.5))$, $z_3 \sim \text{Exponential}(\lambda = 3)$, and $z_4 \sim \text{Exponential}(\lambda = 0.5)$;
2. $\mathbf{x} := R\mathbf{z}$, where R is a fixed semi-positive definite 5×5 matrix defining a fixed projection of $\mathbf{z}$ into the observed space.

p_0 has α=1, β=-1

p_1 has α=0, β=0



(a)



(c)

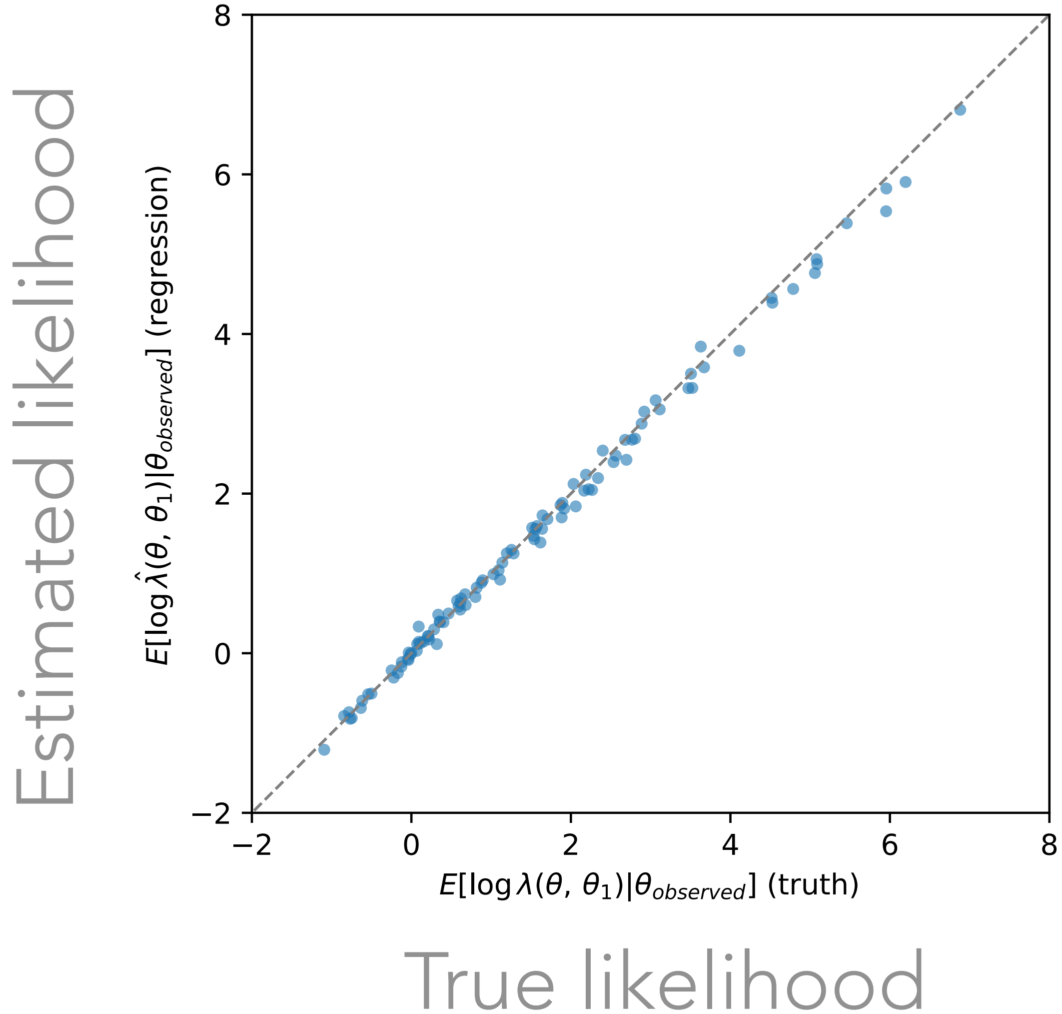
Let assume 5D data **x** generated from the following process p_0:

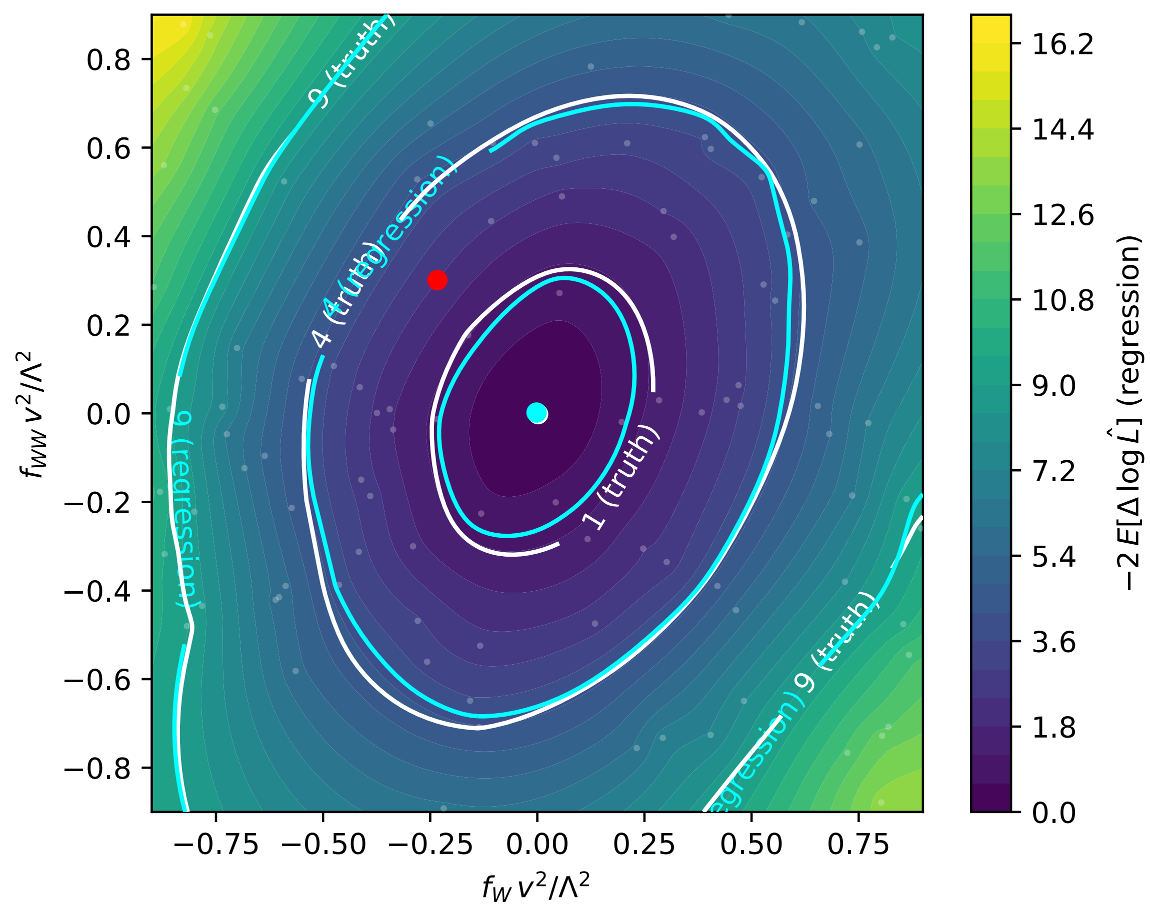
1. $\mathbf{z} := (z_0, z_1, z_2, z_3, z_4)$, such that $z_0 \sim \mathcal{N}(\mu = \alpha, \sigma = 1)$, $z_1 \sim \mathcal{N}(\mu = \beta, \sigma = 3)$, $z_2 \sim \text{Mixture}(\frac{1}{2}\mathcal{N}(\mu = -2, \sigma = 1), \frac{1}{2}\mathcal{N}(\mu = 2, \sigma = 0.5))$, $z_3 \sim \text{Exponential}(\lambda = 3)$, and $z_4 \sim \text{Exponential}(\lambda = 0.5)$;
2. $\mathbf{x} := R\mathbf{z}$, where R is a fixed semi-positive definite 5×5 matrix defining a fixed projection of **z** into the observed space.

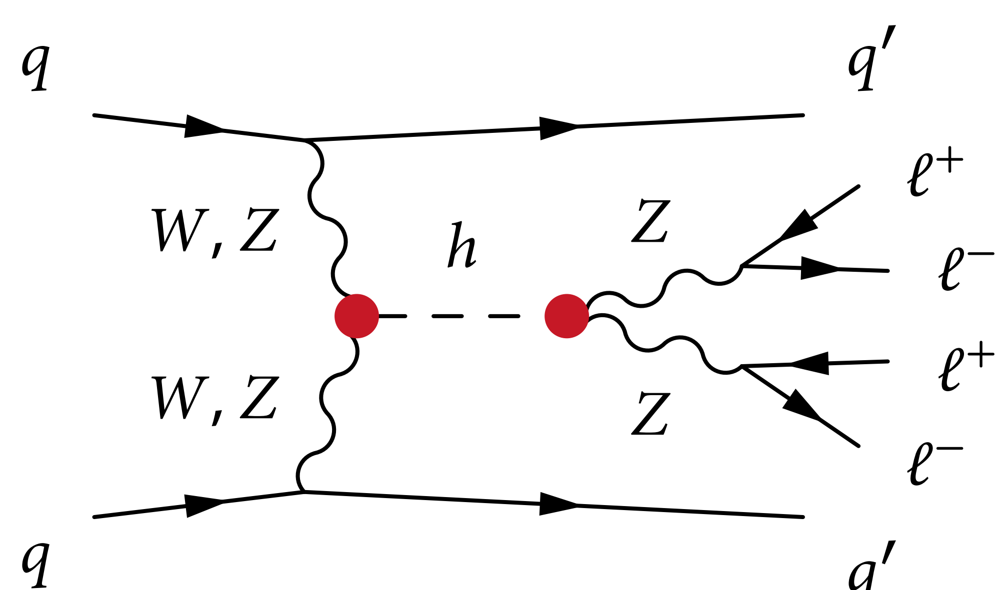
p_0 has α=1, β=-1

p_1 has α=0, β=0

LEARNING A 16 DIM LIKELIHOOD



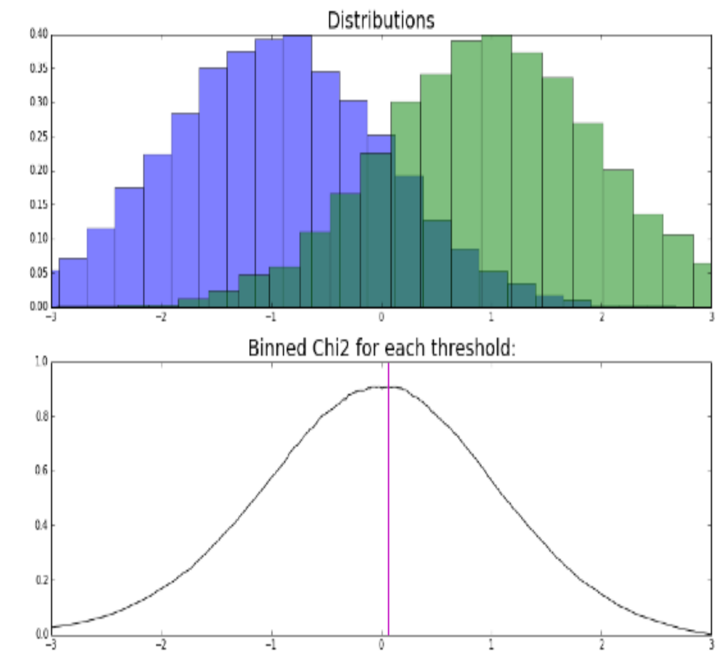




High Dimensional Reweighting

GBReweighter

Nice blog post by Alex Rogozhnikov [link] (Yandex Data Science group based at CERN contributing to **hep_ml** package). He developed **GBReweighter.**



Find decision trees that **maximize** "symmetrized χ^2"

$$\chi^2 = \sum_{\text{bin}} \frac{(w_{\text{bin, original}} - w_{\text{bin, target}})^2}{w_{\text{bin, original}} + w_{\text{bin, target}}}$$

"Note, that I want it to be as high as possible. If the weights of original and target distribution are equal, I don't need to reweight in this bin and corresponding summand is zero. If the summand is high, reweighting in bin is needed."

Then he boosts:

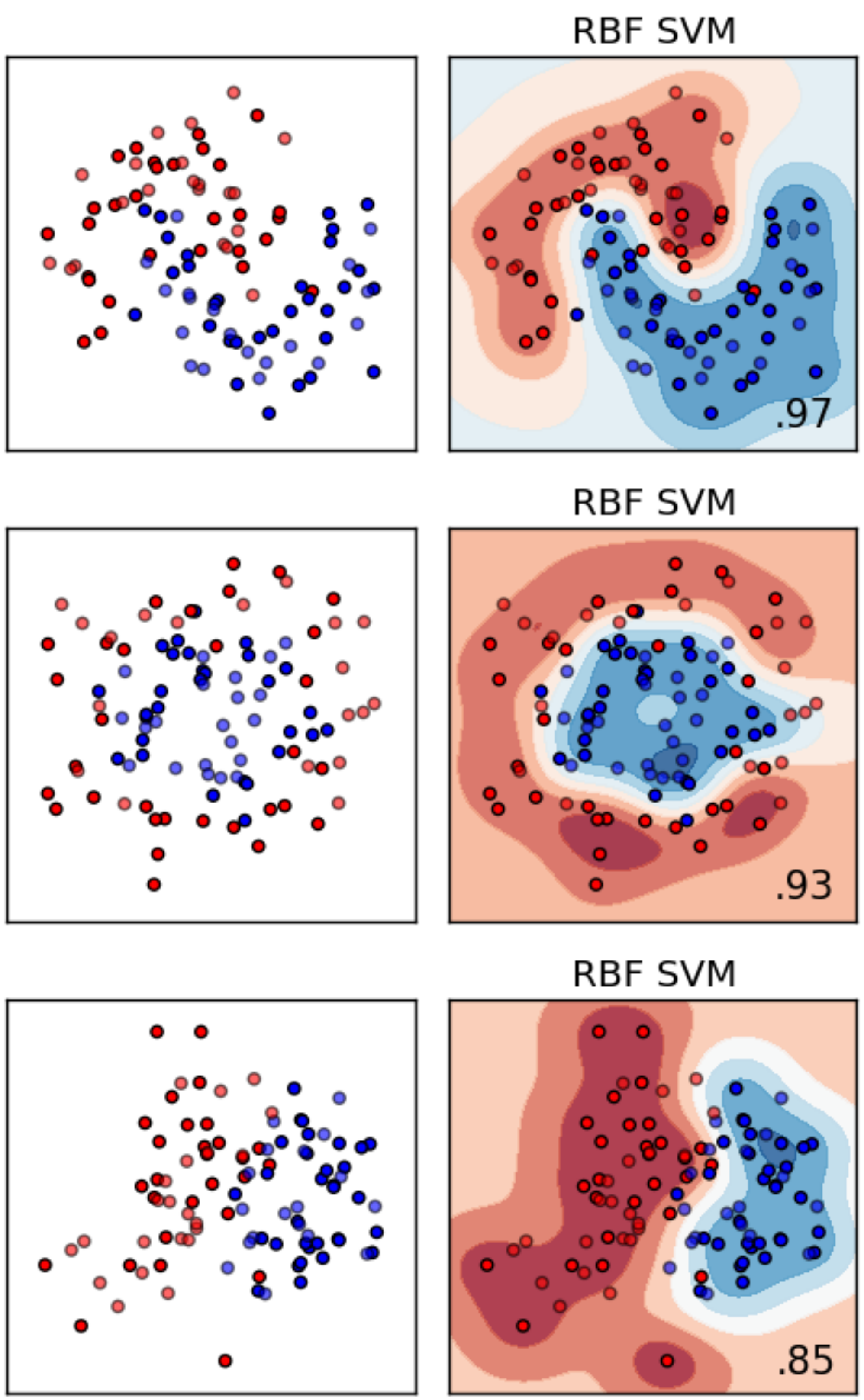
1. build a shallow tree to maximize symmetrized χ^2
2. compute predictions in leaves:

$$\text{leaf\_pred} = \ln \frac{w_{\text{leaf, target}}}{w_{\text{leaf, original}}}$$

3. reweight distributions (compare with AdaBoost):

$$w \leftarrow \begin{cases} w, & \text{if event from target (RD) distribution} \\ w \times e^{\text{pred}}, & \text{if event from original (MC) distribution} \end{cases}$$

```
from hep_ml.reweight import GBReweighter
gb = GBReweighter()
gb.fit(mc_data, real_data)
gb.predict_weights(mc_other_channel)
```



Idea is to train a classifier for signal (H_1) vs. background (H_0)

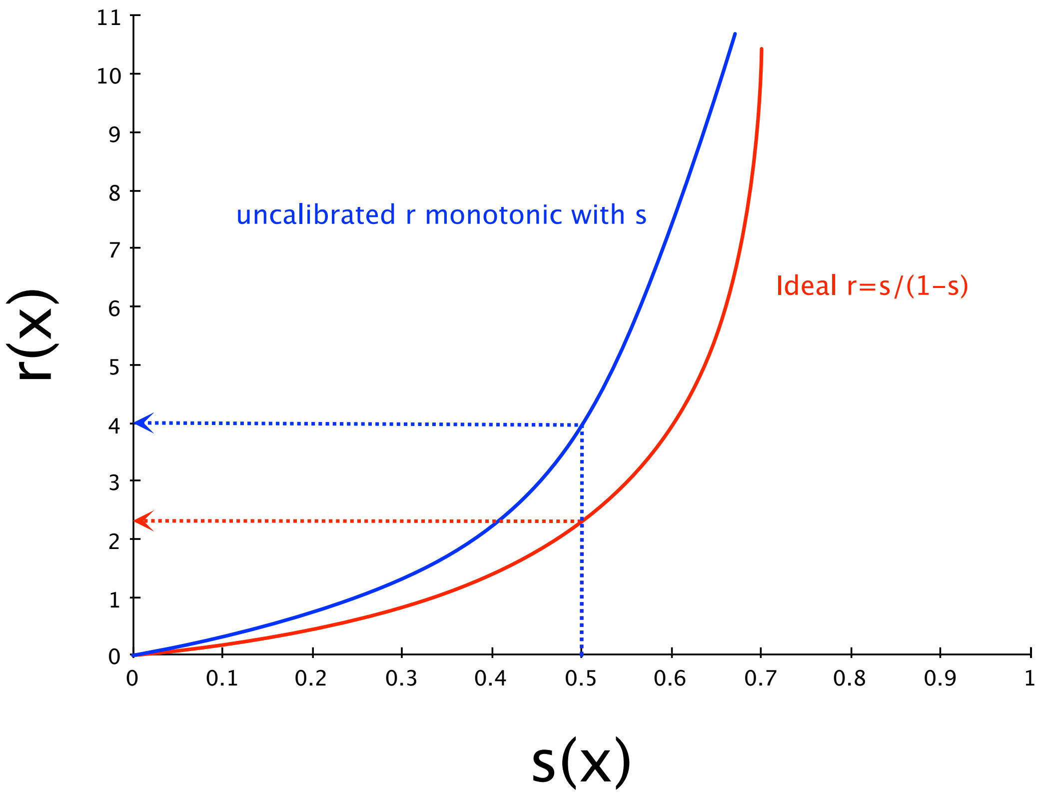
- with a balanced sample of y=0,1 labels and a squared loss the optimal classifier would learn the regression function

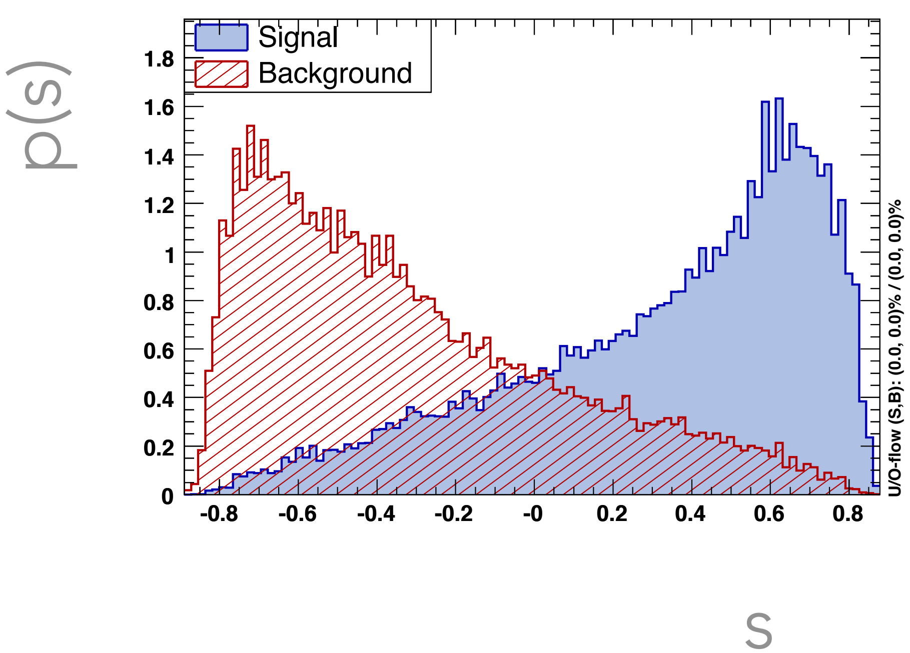
$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

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

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

IMPORTANCE OF CALIBRATION





Theorem if s monotonic with r —>

Ideally classifier will learn

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

which is 1-to-1 with the likelihood ratio

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

but often inverting s(x)→ r(x) typically doesn't work well because the classifier isn't well calibrated and learns something monotonic in r(x).

Still ok, just need to calibrate it

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

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

If s(x) is monotonic with $p_1(x)/p_0(x)$, then we have

Theorem 1: We have the following equality

$$\frac{p_1(s(x))}{p_0(s(x))} = \frac{p_1(x)}{p_0(x)} \; . \tag{2.6}$$

Proof For $x \in \Omega_{s^*}$, we can factor out of the integral the constant $p_1(x)/p_0(x)$. Thus

$$p_1(s^*) = \int d\Omega_{s^*} p_1(x)/|\hat{n} \cdot \nabla s| = \frac{p_1(x)}{p_0(x)} \int d\Omega_{s^*} p_0(x)/|\hat{n} \cdot \nabla s| \; , \tag{2.7}$$

and the integrals cancel in the likelihood ratio

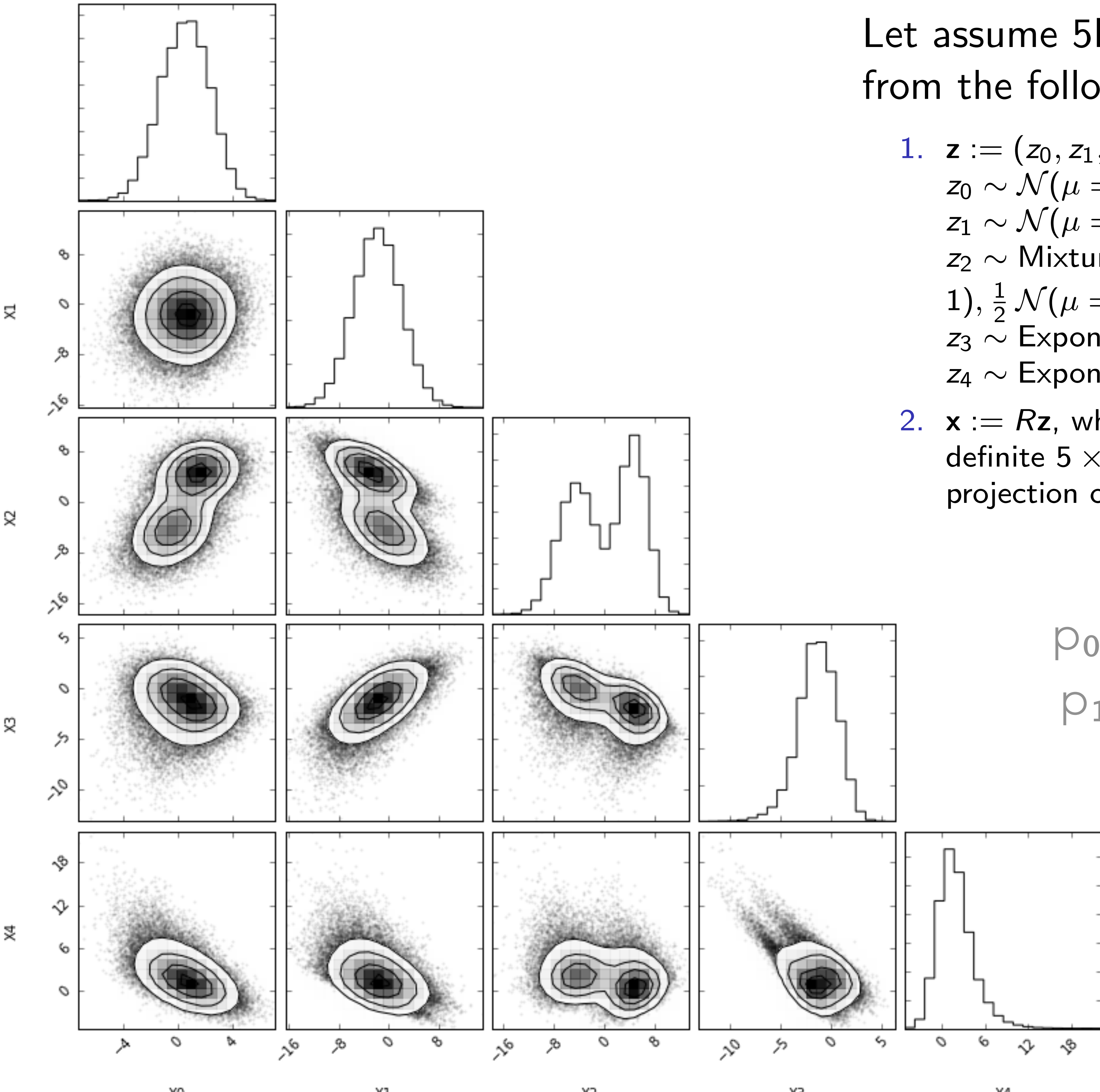
$$\frac{p_1(s^*)}{p_0(s^*)} = \frac{p_1(x)}{p_0(x)} \frac{\int d\Omega_{s^*} p_0(x)/|\hat{n} \cdot \nabla s|}{\int d\Omega_{s^*} p_0(x)/|\hat{n} \cdot \nabla s|} = \frac{p_1(x)}{p_0(x)} \qquad \forall x \in \Omega_{s^*} . \tag{2.8}$$

One can think of the ratio $p_1(s)/p_0(s)$ as a way of calibrating the the discriminative classifier and correcting for the monotonic transformation m of the desired likelihood ratio as in Eq. 1.3.

A toy example

REPRODUCIBLE NOTEBOOK WITH CODE AND PLOTS:
https://github.com/cranmer/carl-notebooks/blob/master/reweighing-high-dimensional-data.ipynb

THE DATA



Let assume 5D data **x** generated from the following process p_0:

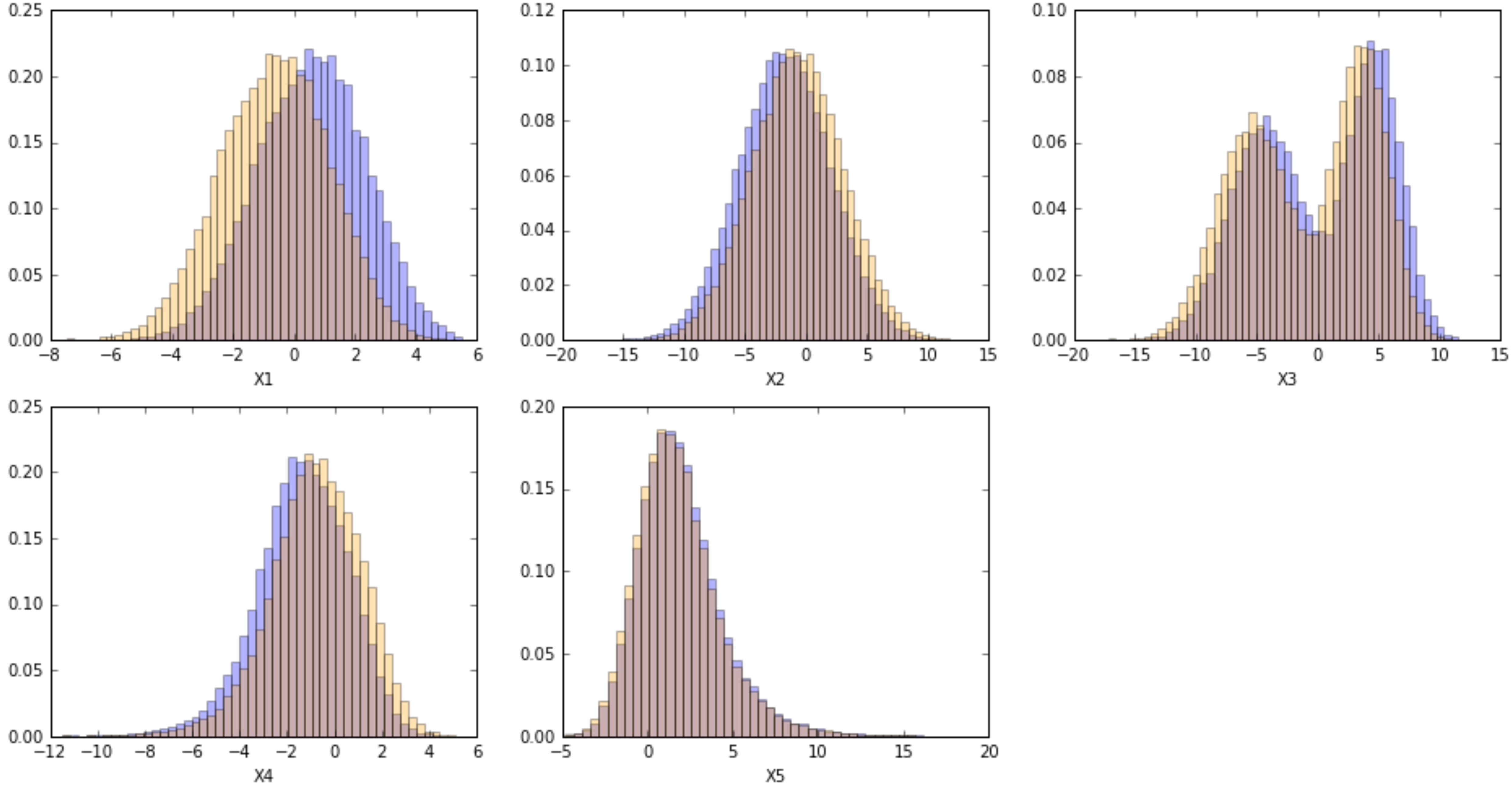
1. $\mathbf{z} := (z_0, z_1, z_2, z_3, z_4)$, such that $z_0 \sim \mathcal{N}(\mu = \alpha, \sigma = 1)$, $z_1 \sim \mathcal{N}(\mu = \beta, \sigma = 3)$, $z_2 \sim \text{Mixture}(\frac{1}{2}\mathcal{N}(\mu = -2, \sigma = 1), \frac{1}{2}\mathcal{N}(\mu = 2, \sigma = 0.5))$, $z_3 \sim \text{Exponential}(\lambda = 3)$, and $z_4 \sim \text{Exponential}(\lambda = 0.5)$;
2. $\mathbf{x} := R\mathbf{z}$, where R is a fixed semi-positive definite 5×5 matrix defining a fixed projection of $\mathbf{z}$ into the observed space.

p_0 has α=1, β=-1

p_1 has α=0, β=0

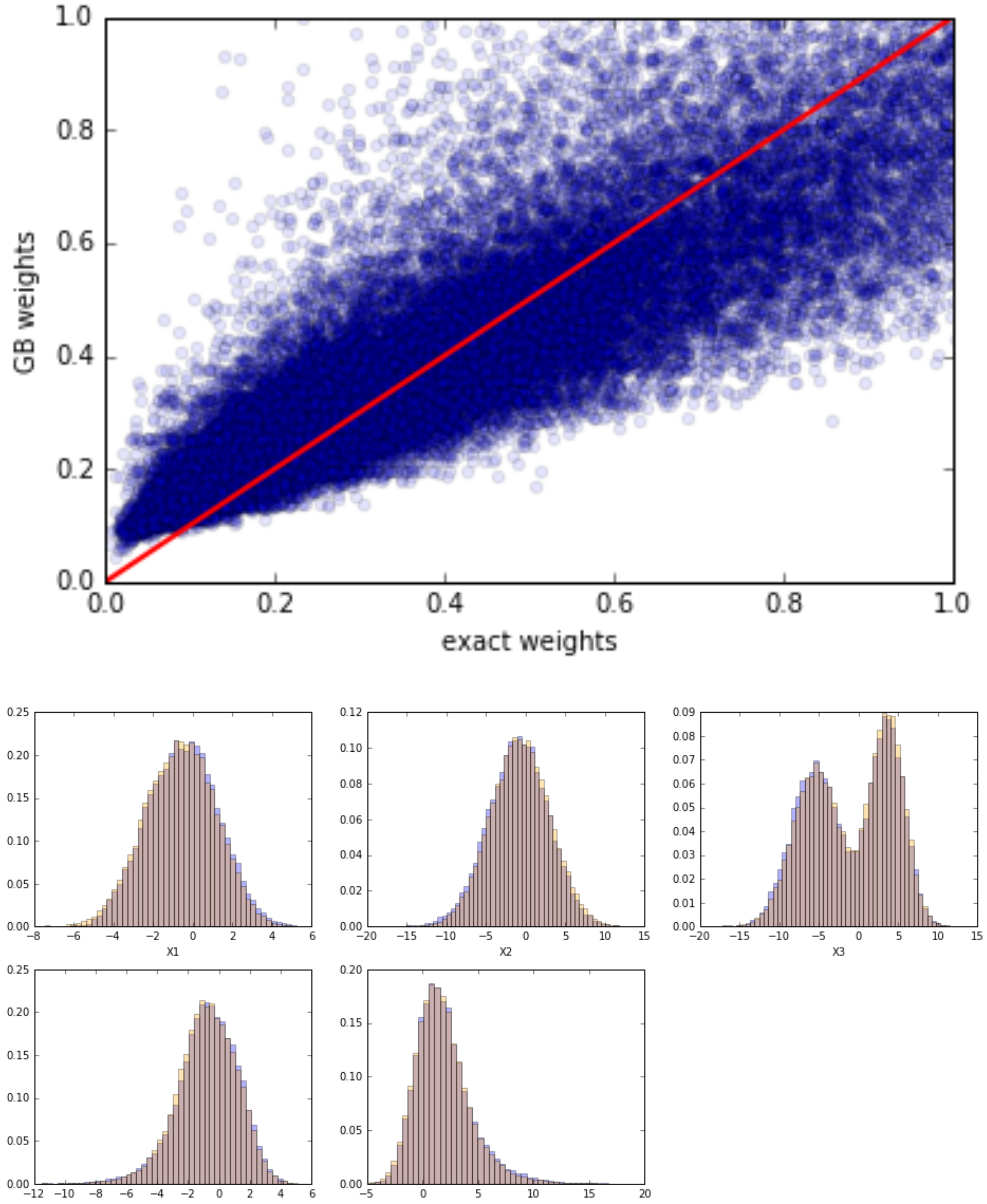
ORIGINAL VS. TARGET DISTRIBUTIONS

1-d projections of the original and target distributions

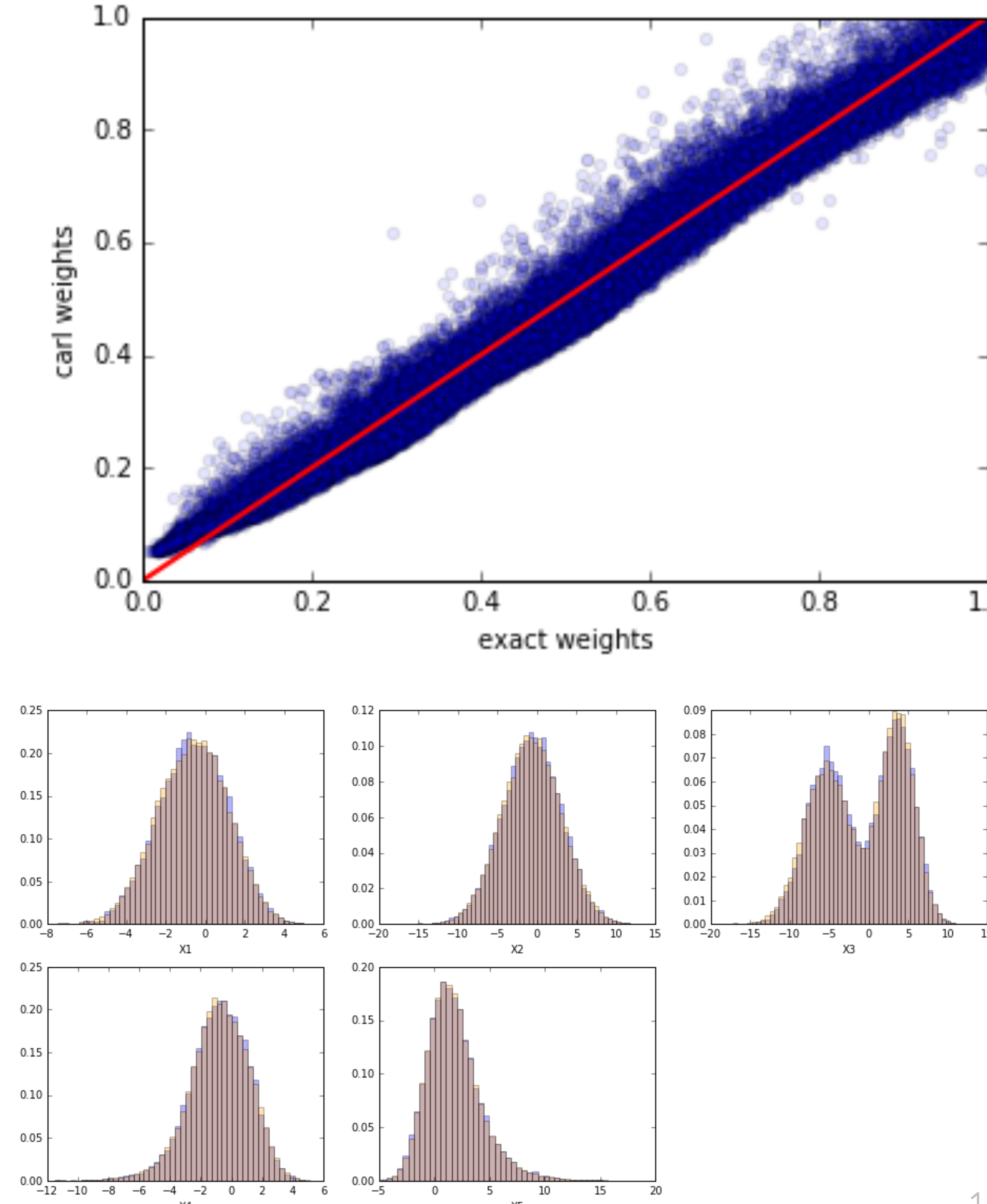


TWO REWEIGHING METHODS: 100K SAMPLES

hep_ml.GBReweigher



carl with calibrated MLP

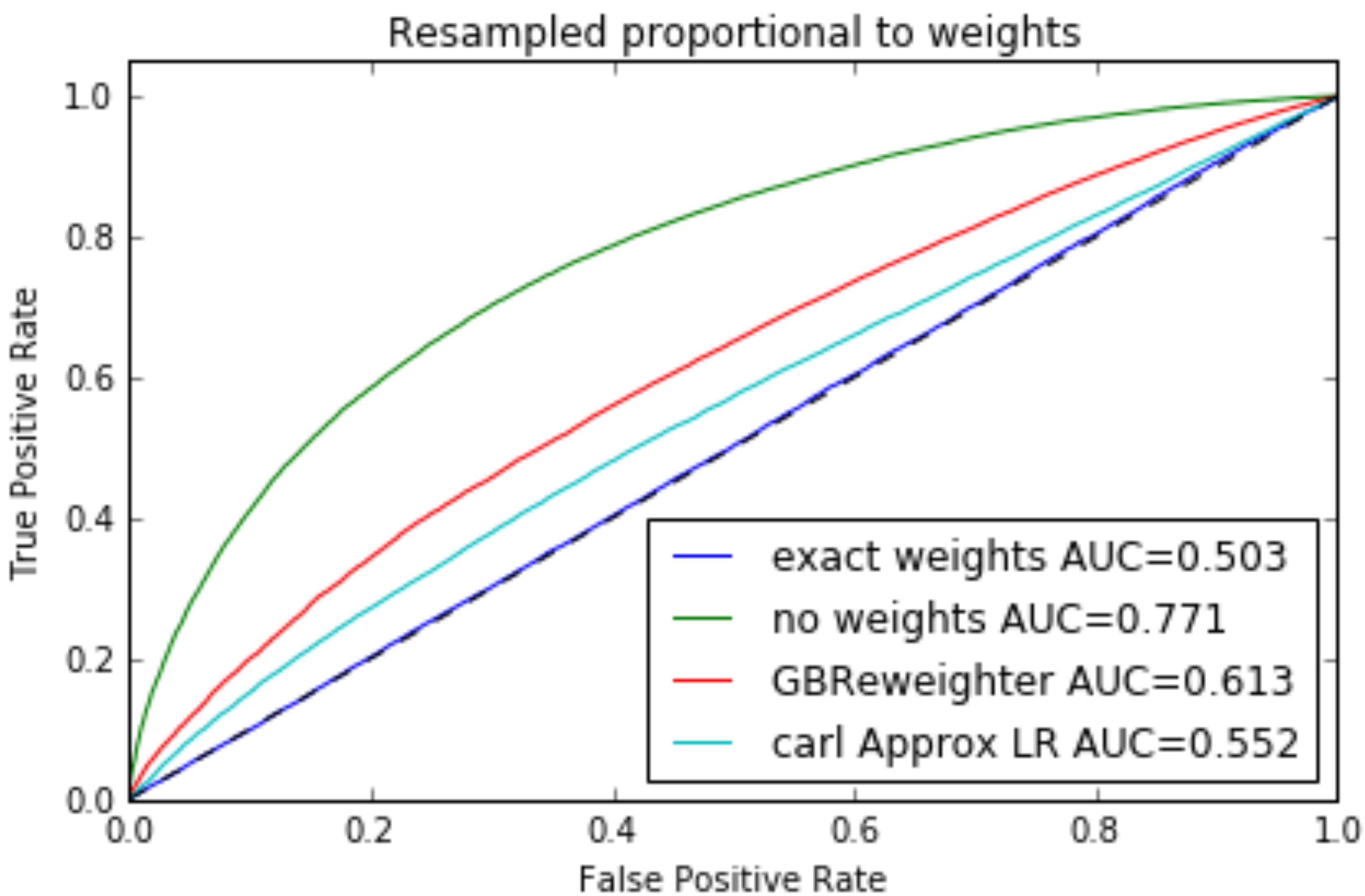


EVALUATING THE QUALITY OF THE REWEIGHTING

Train a new classifier to **discriminate** between events from target and events resampled from original distribution with probabilities given by the predicted weights

- classifier can easily distinguish unweighted distributions;
- exact weights are perfect (AUC~0.5)
- carl doing a little better than GBReweighter on this problem (no special effort to tune either)
- neither is perfect

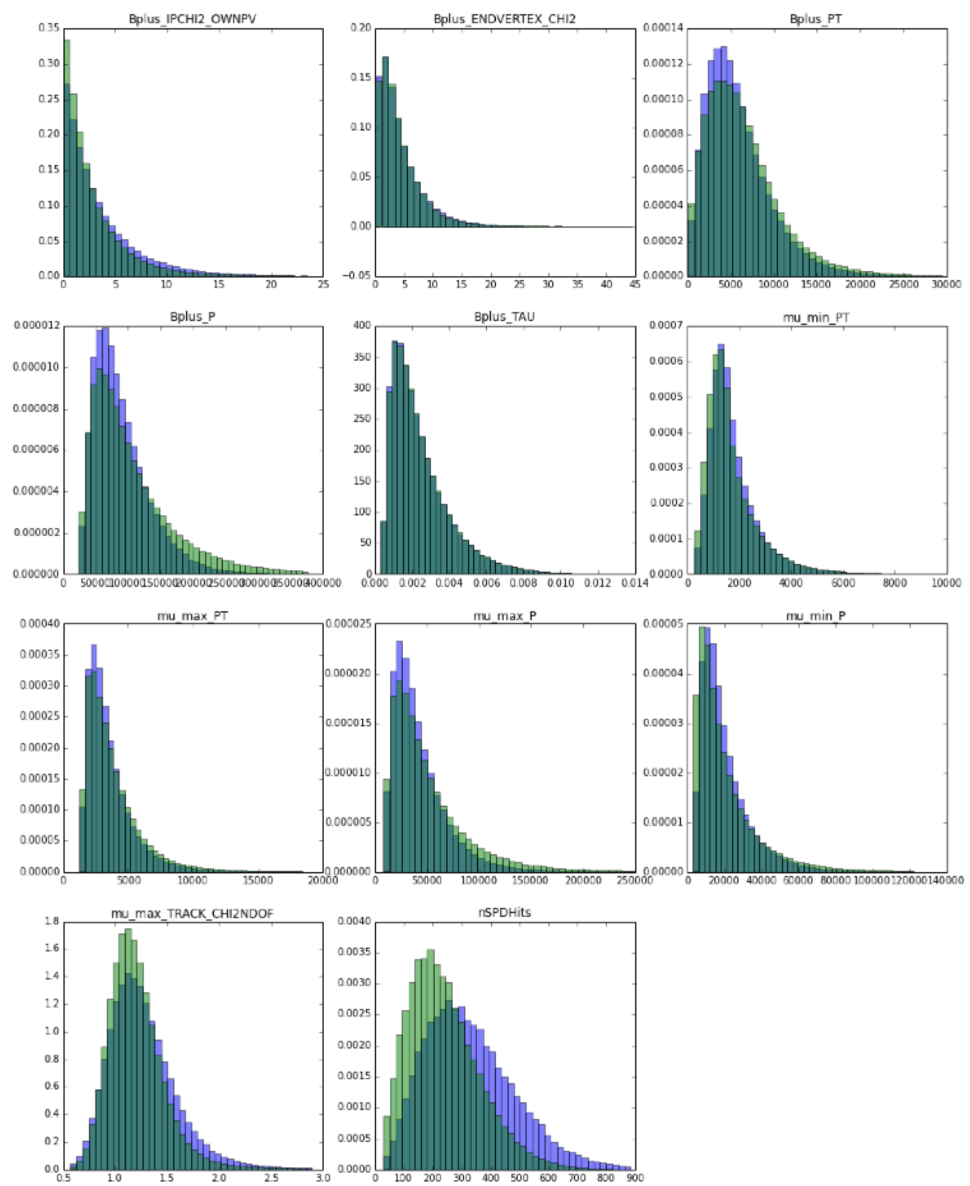
Important:
Performance evaluated on independent testing sample



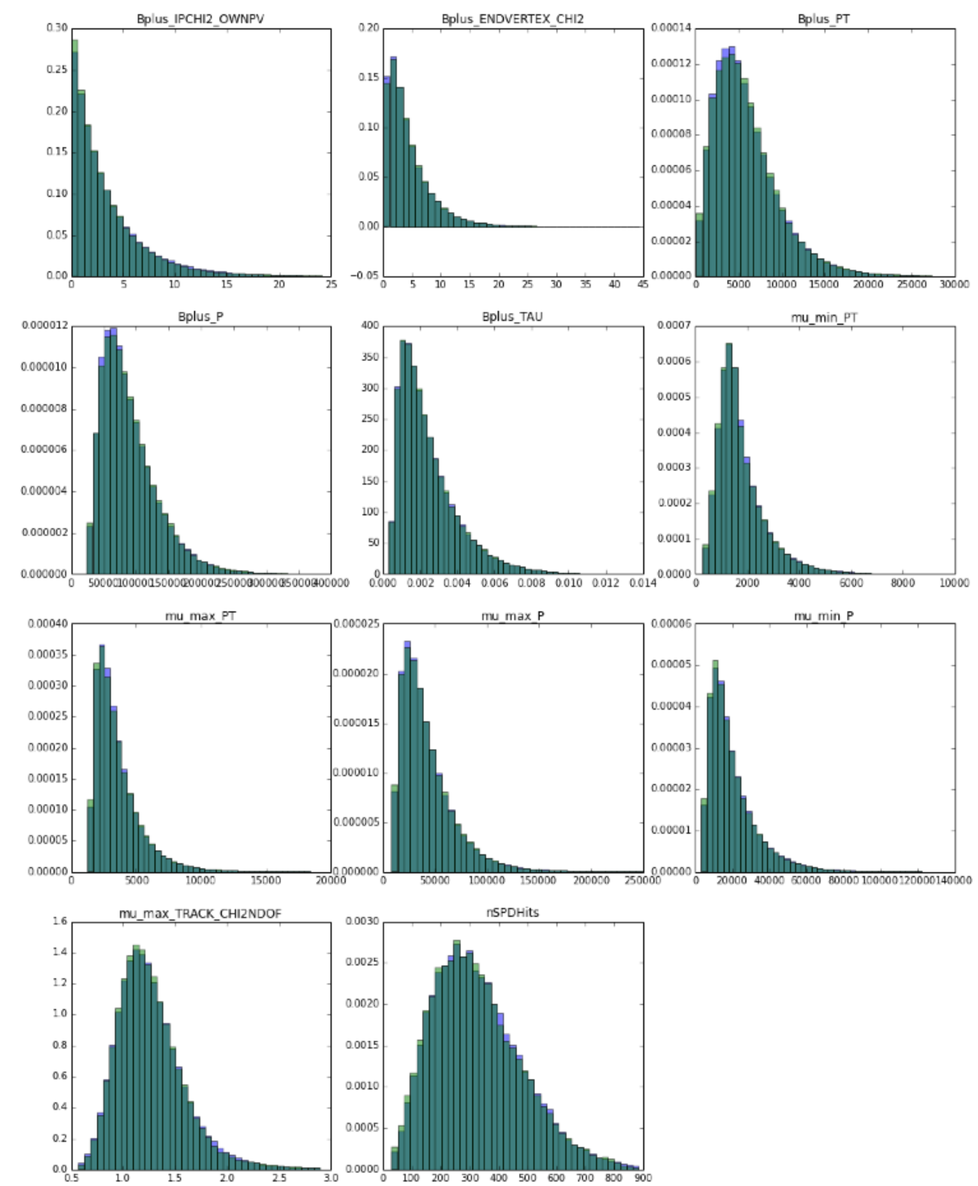
Alex's example

REPRODUCIBLE NOTEBOOK WITH CODE AND PLOTS:

https://github.com/cranmer/carl-notebooks/blob/master/reweighing-mc-data.ipynb example data: https://github.com/arogozhnikov/hep_ml/blob/data/data_to_download/



before reweighting

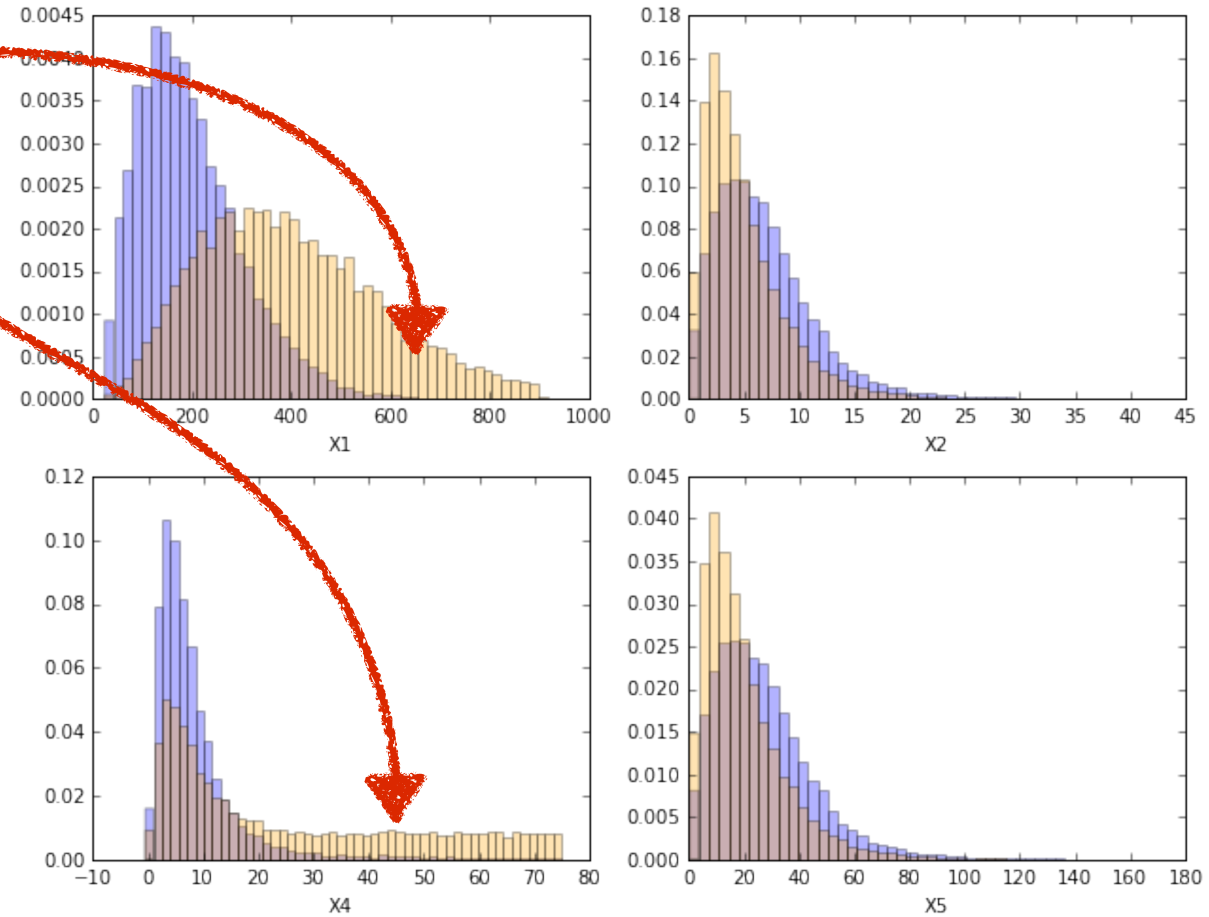


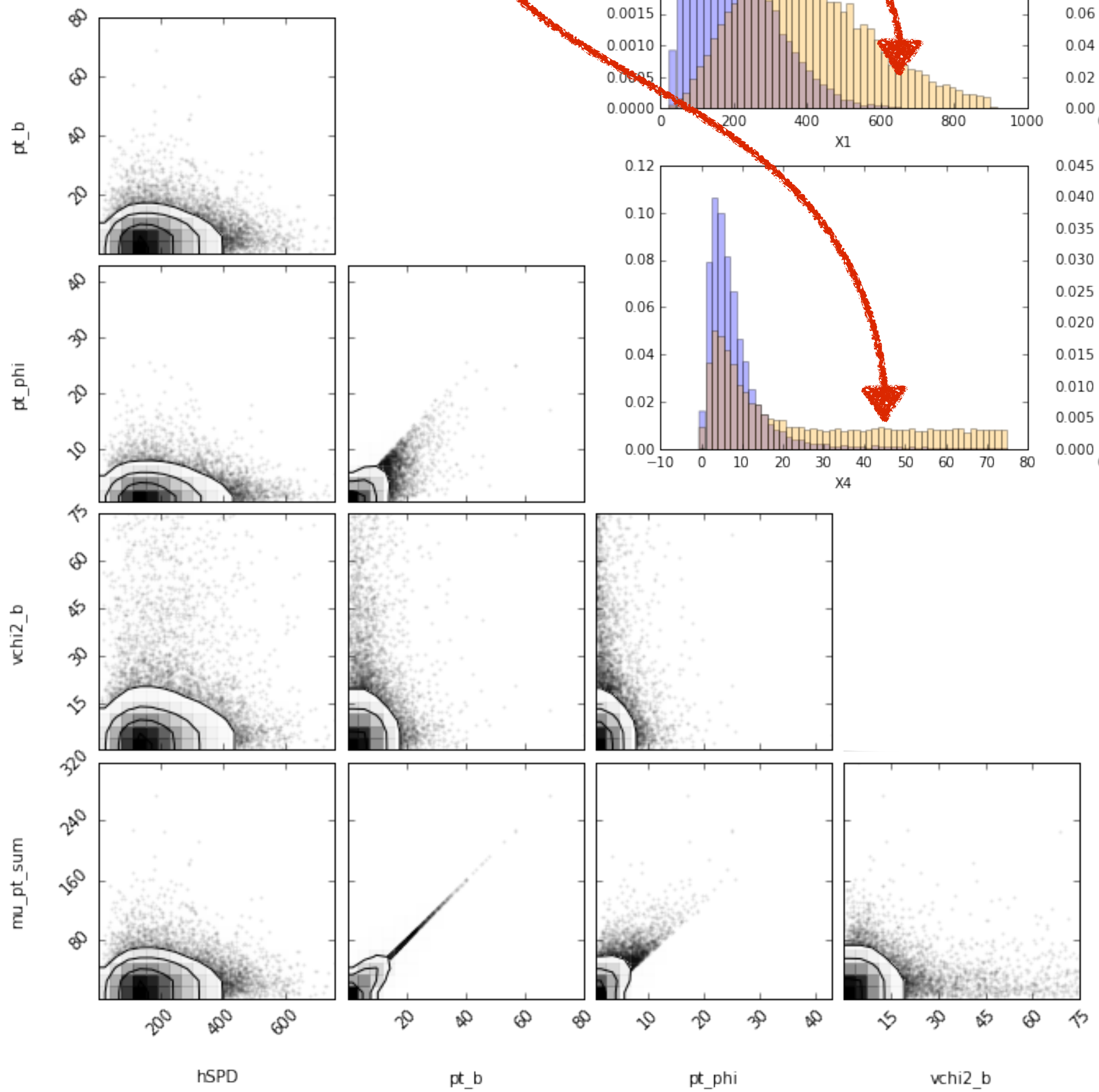
after reweighting

looks great here, but using all the same data for training and making the plots. What does the performance look like if we hold out an independent testing set?

1D AND 2D PROJECTIONS OF THE DATA

Huge differences





I SPENT SOME TIME ON THIS BUT SEEMS LIKE A BAD EXAMPLE, HUGE WEIGHTS!

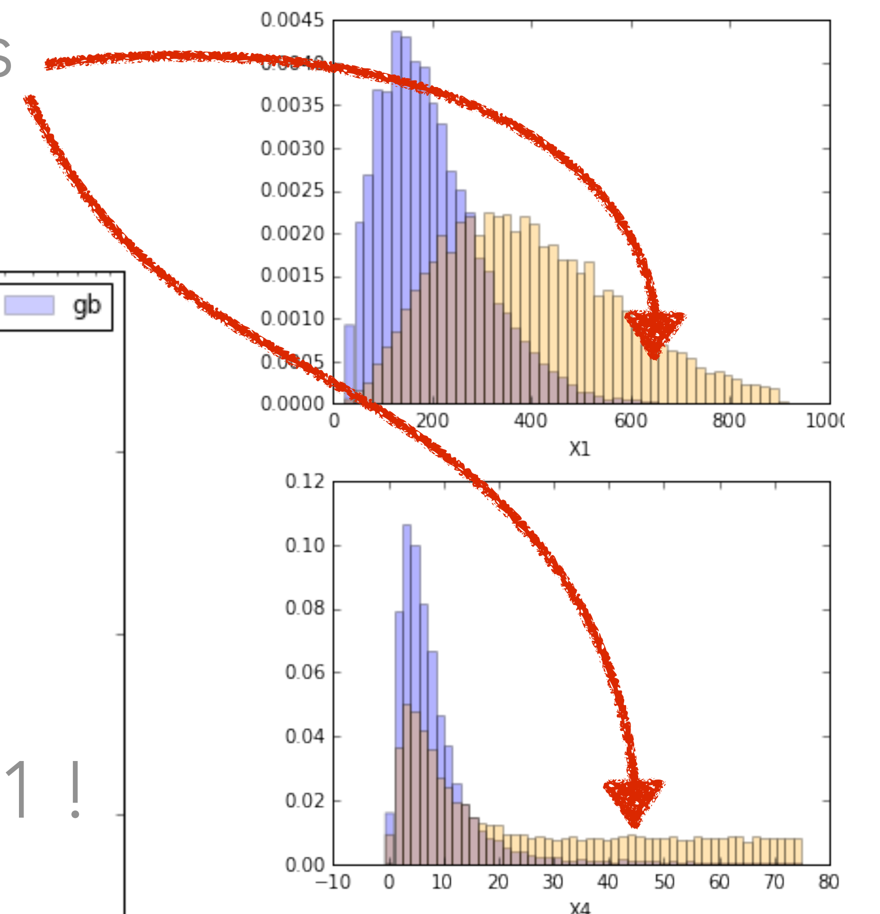
HUGE WEIGHTS AND "COMMON SUPPORT"

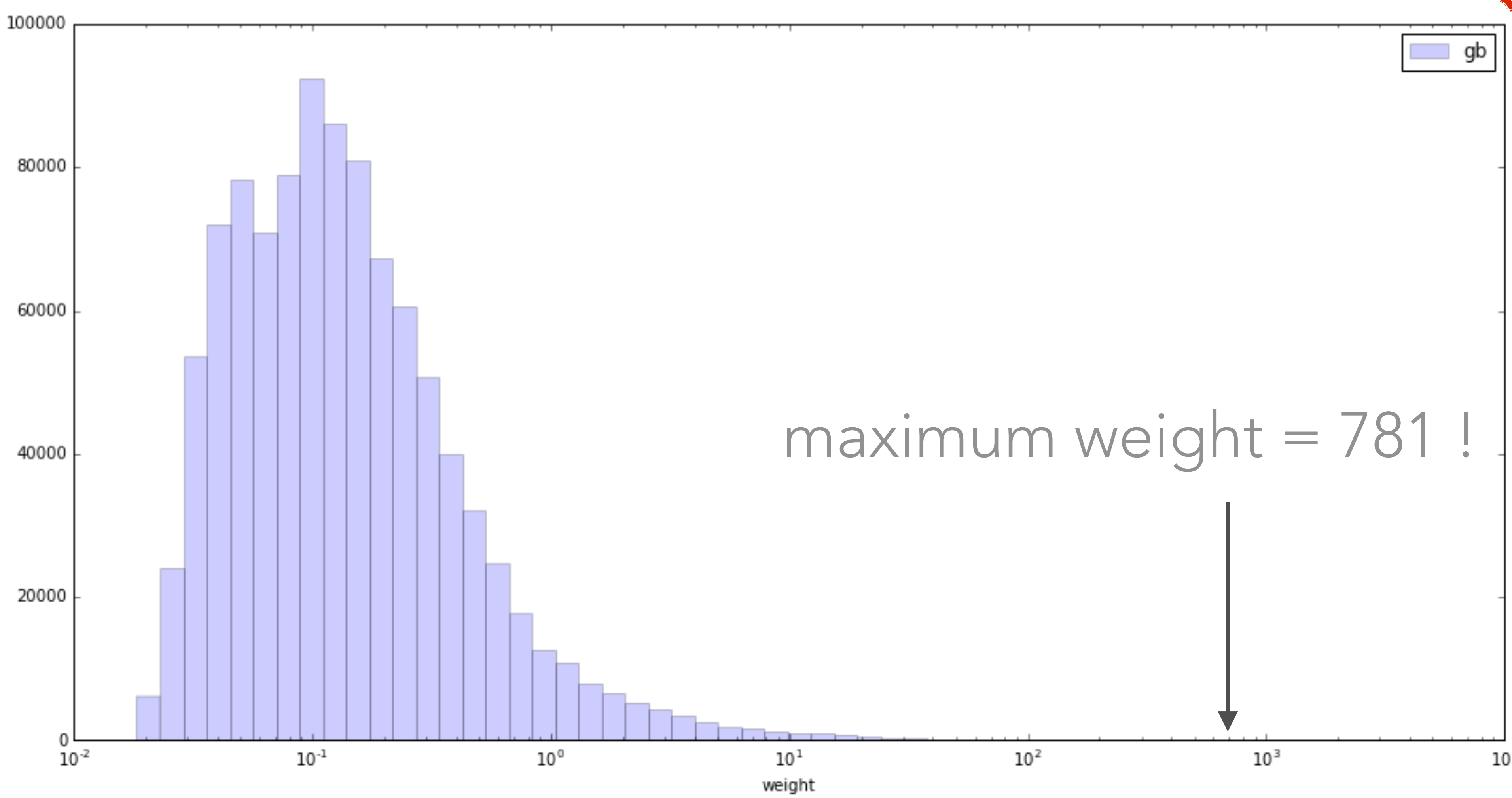
Since the distributions are so different, you expect to see huge weights, and you do.

For reweighting to work, $p_0(x)$ and $p_1(x)$ need a common support. To check this, I recommend to make a histogram of the weights.

This causes all sorts of problems downstream. It's like that one QCD event that passes your cuts and has a huge weight.

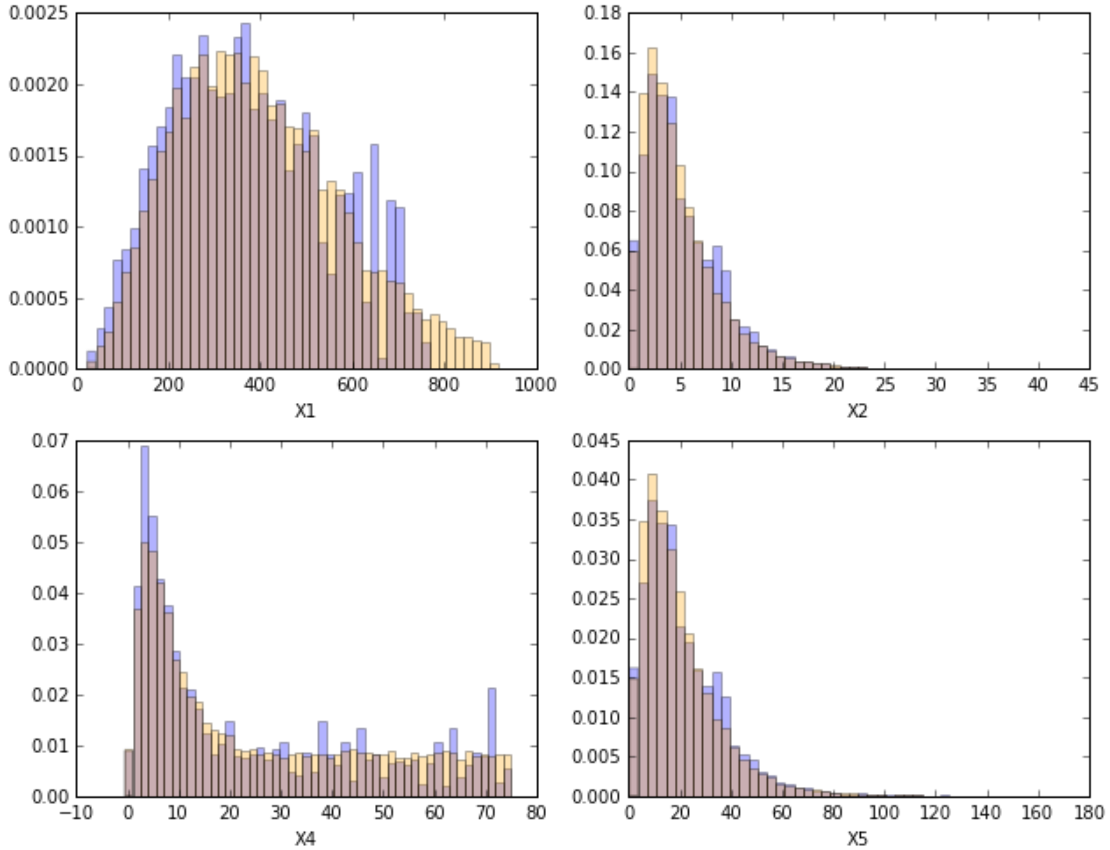
Huge differences



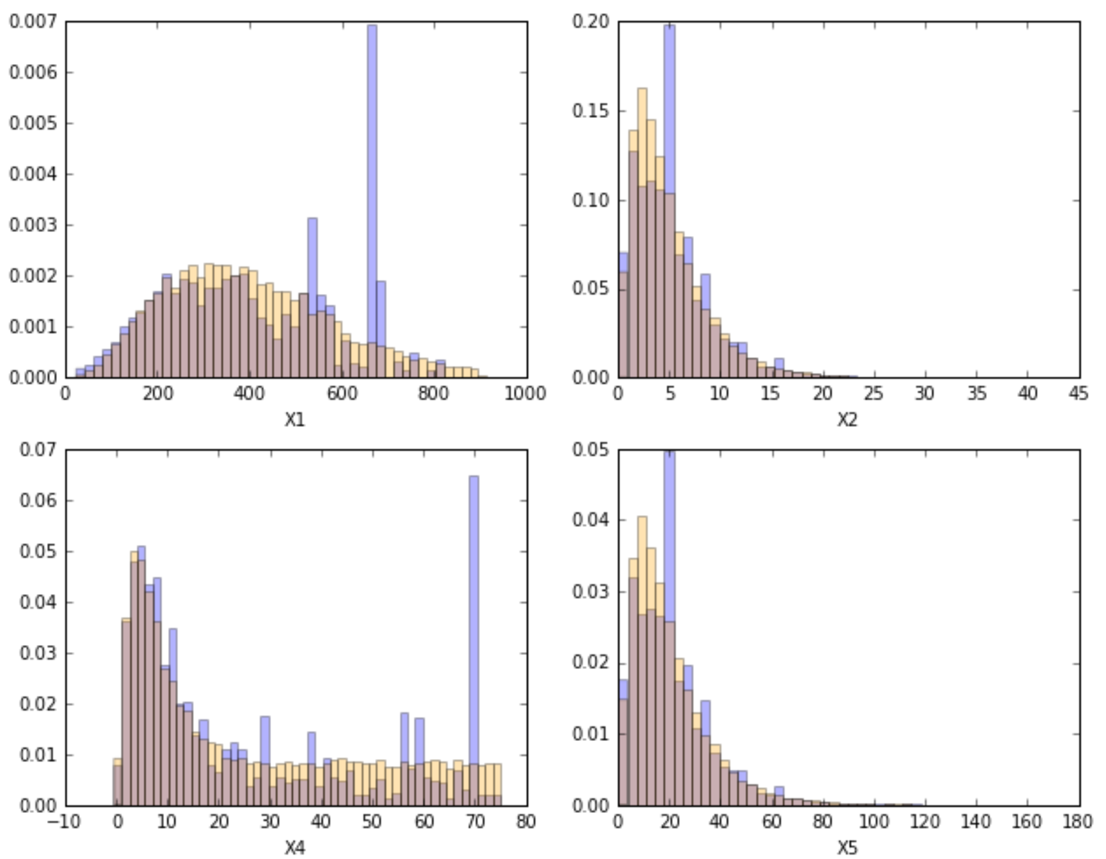


OVERFITTING

NB: original example had many more events from target distribution.
Here I'm using balanced data



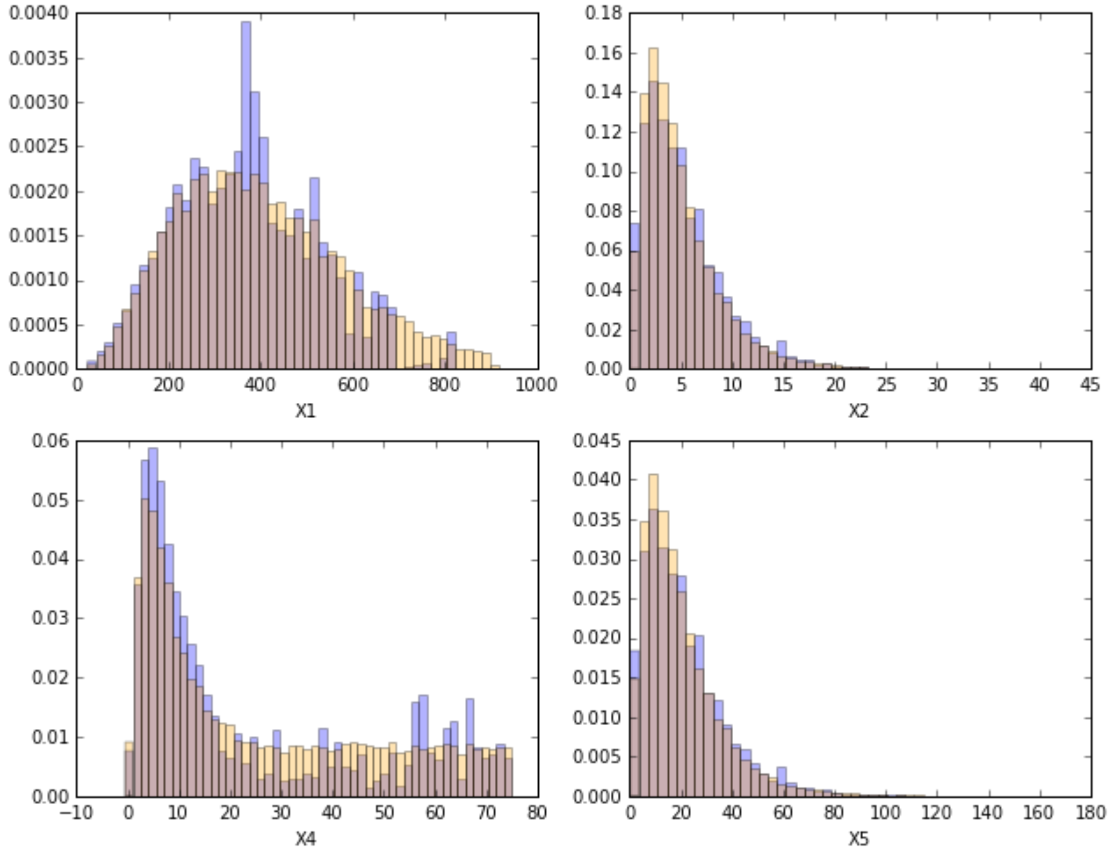
GBRweighter:
apply reweighing to training data



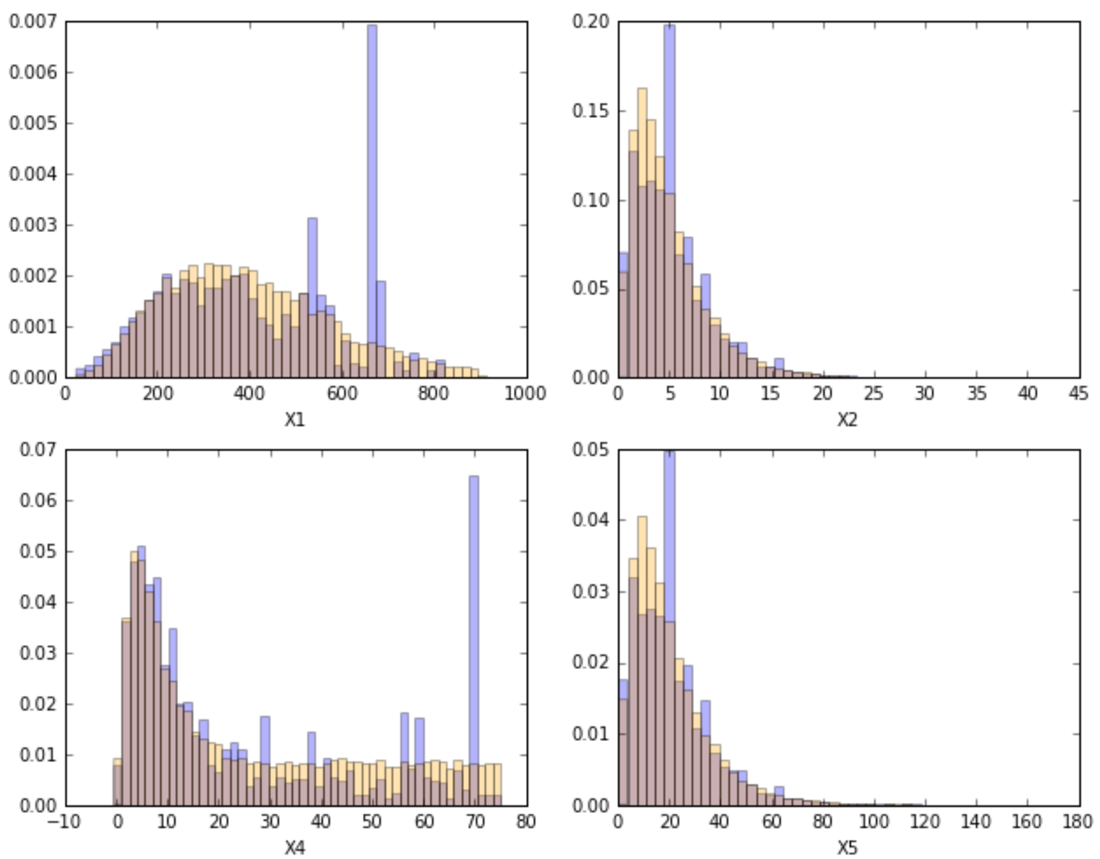
GBRweighter:
apply reweighing to testing data

carl vs. GBRewighter

NB: original example had many more events from target distribution.
Here I'm using balanced data



carl:
apply reweighing to testing data



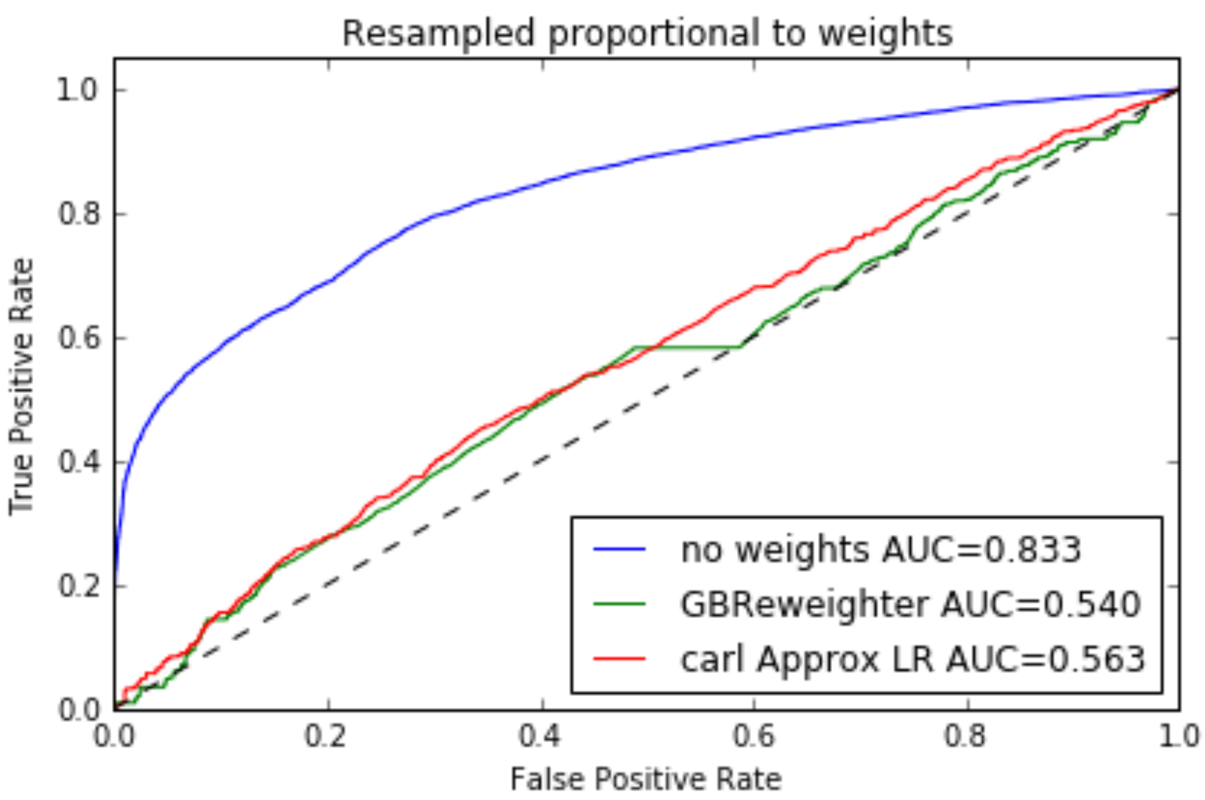
GBRweighter:
apply reweighing to testing data

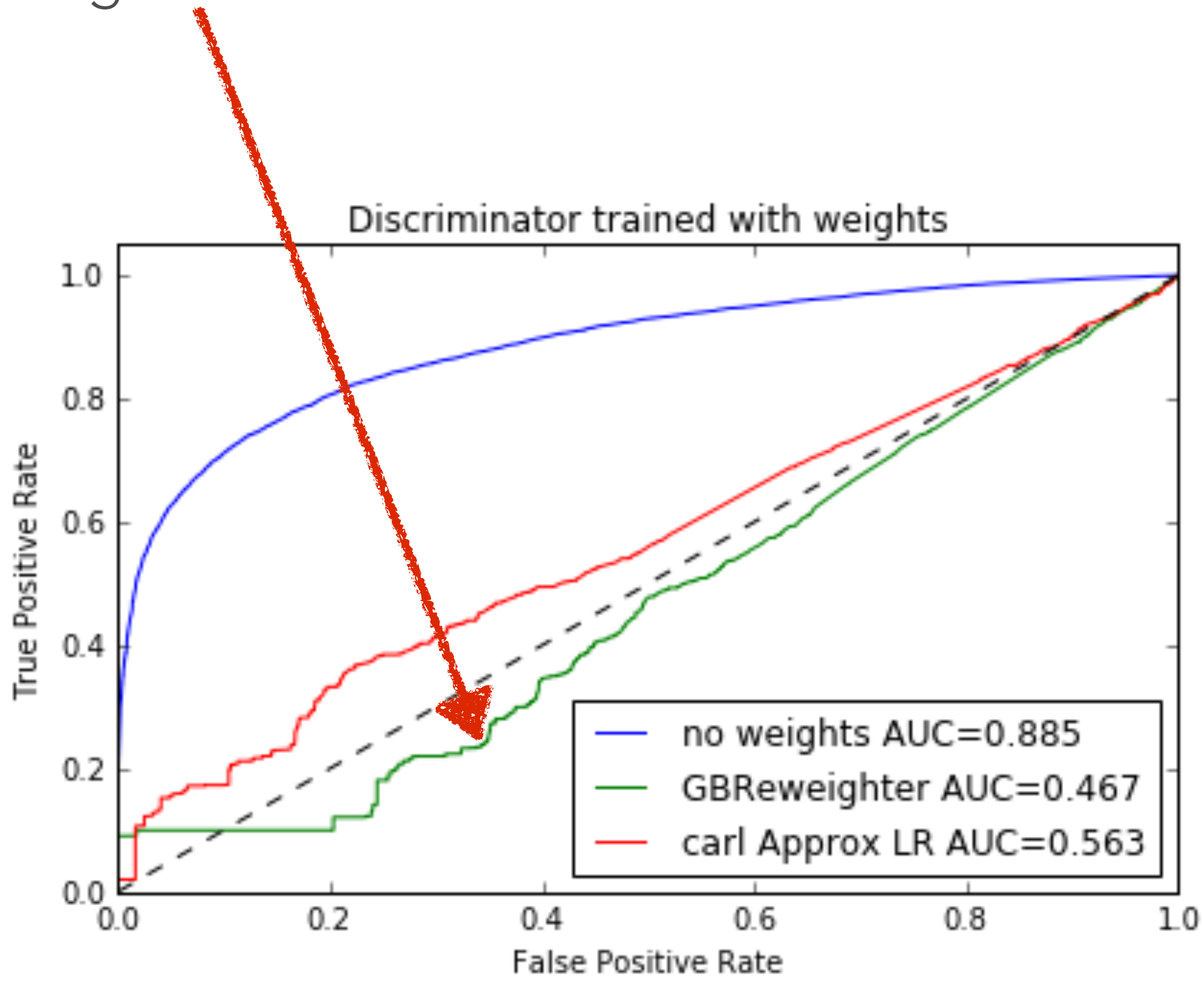
NEITHER LOOK GOOD, HUGE WEIGHTS A PROBLEM

DIFFERENT APPROACHES TO DISCRIMINATOR

A **discriminator** is a good tool to quantify the performance of the reweighting. Two approaches:

- Resample the original distribution with probabilities proportional to the weights. Train classifier with the resulting unweighted events.
  - large weights lead to large fluctuations in the resampling
- Use a discriminator trained with weighted events.
  - large weights can lead to problems in training and evaluation of ROC curve





SUMMARY

Reweighting in high dimensions is hard when you don't have can't evaluate $p_0(x)$ and $p_1(x)$

- histograms and density estimation won't work well
- As Gilles discussed yesterday, classifiers can be used to approximate likelihood/density ratios (implemented in **carl**), which can be used for reweighting
- the GBReweighter is another strategy, and there are other direct density ratio techniques as well

Instead of relying on goodness of fit variables for 1-d projections, it is better to use a discriminator to look for differences between target and reweighted distribution in the high dimensional space

Use cross-validation (independent testing data) to evaluate the performance, or you can fool yourself

Large weights will cause problems downstream, so check that explicitly.

Likelihood Free

THE PLAYERS

forward modeling
generation
simulation

PREDICTION

θ
parameters of interest

p(x, z | θ, ν)

z
latent variables
Monte Carlo truth

x
observed data
simulated data

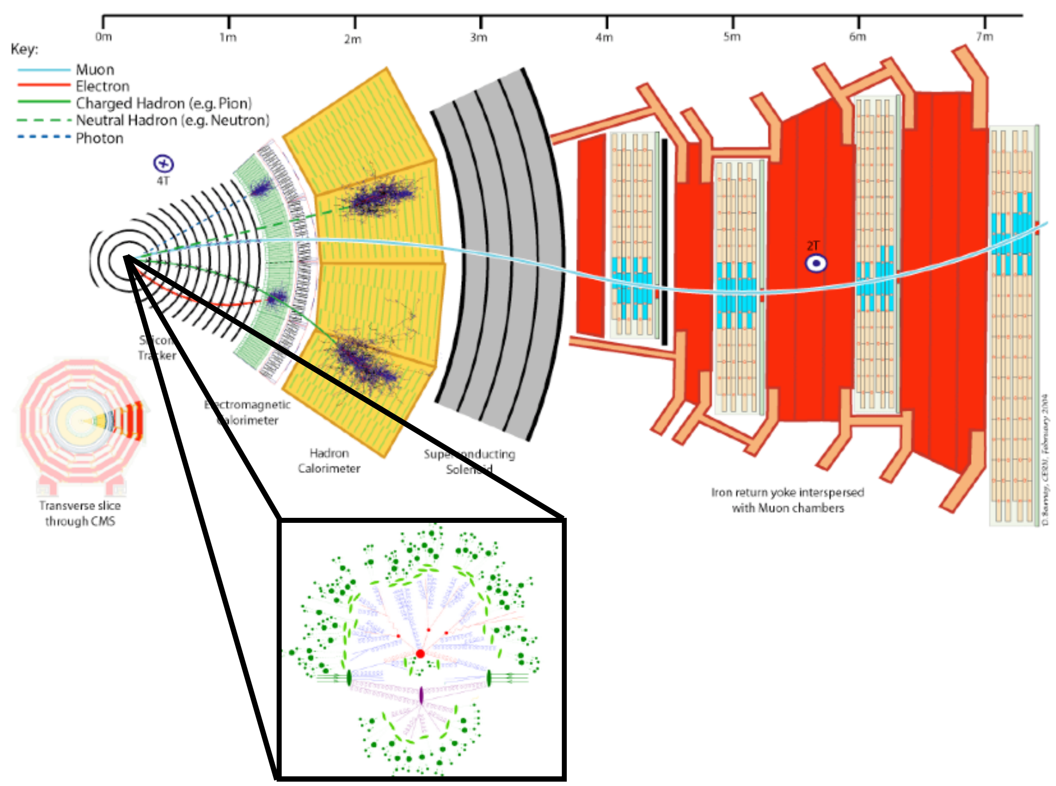
ν
nuisance parameters

INFERENCE

inverse problem
measurement
parameter estimation

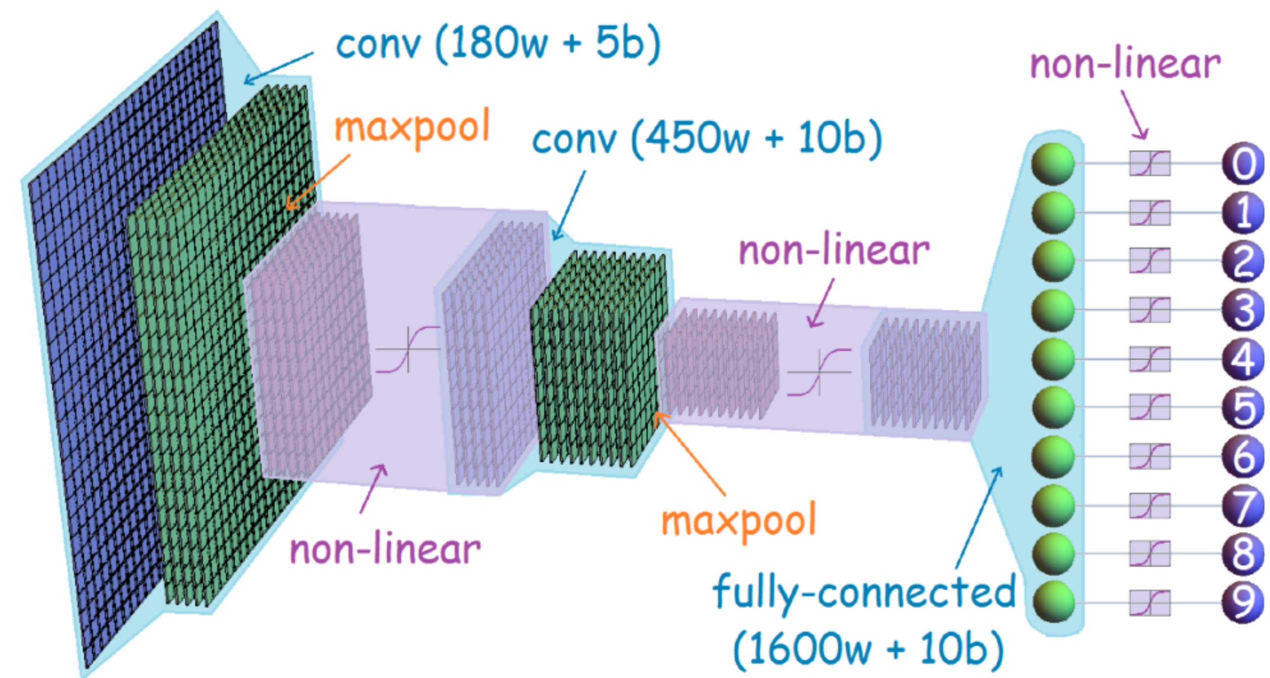
TWO APPROACHES

Use simulator
(much more efficiently)



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

Learn simulator
(with deep learning)

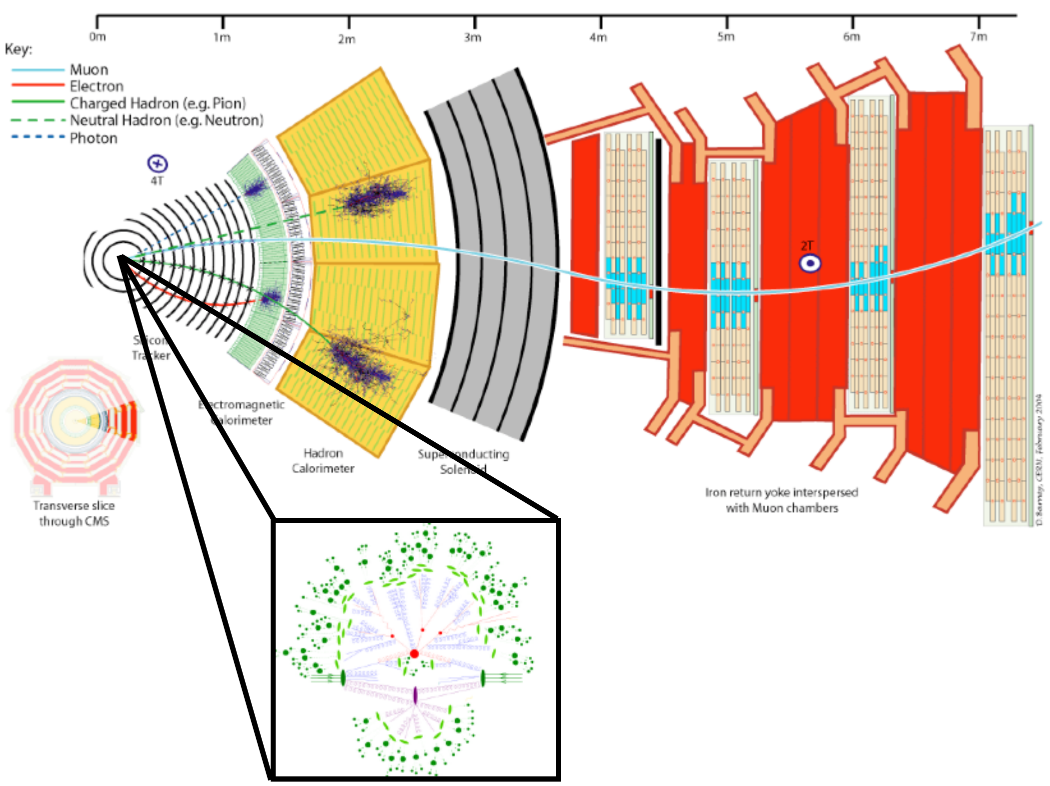


- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

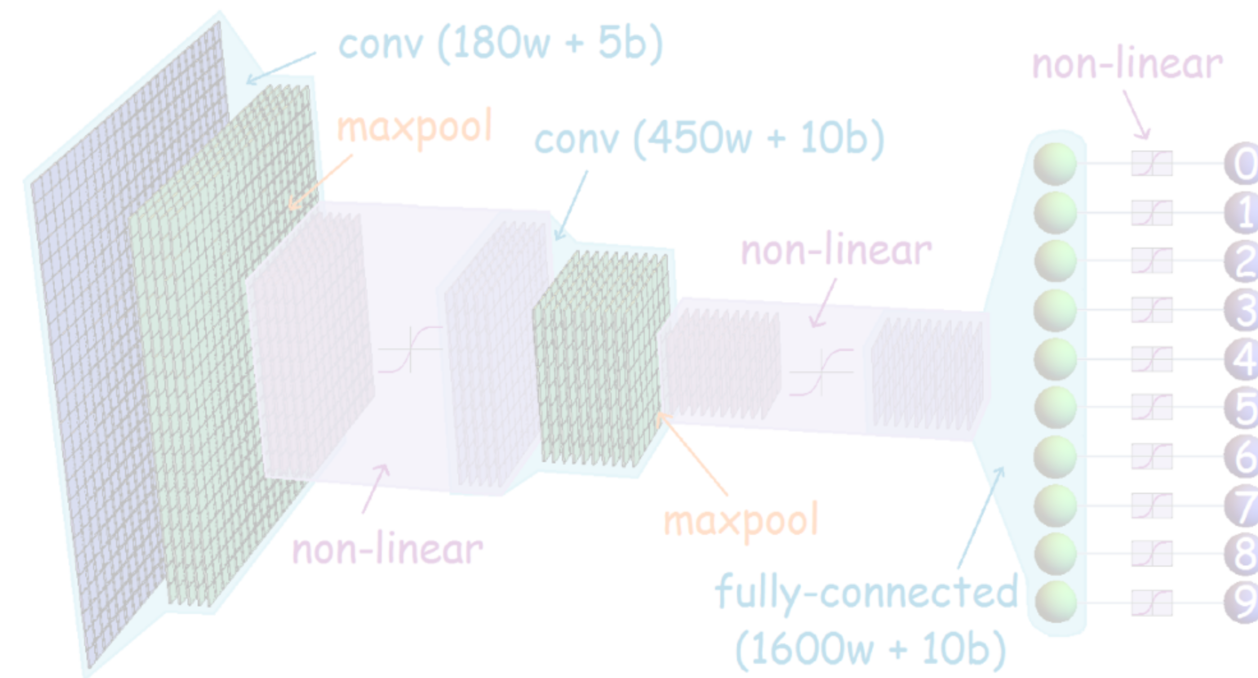
TWO APPROACHES

Use simulator
(much more efficiently)



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

Learn simulator
(with deep learning)

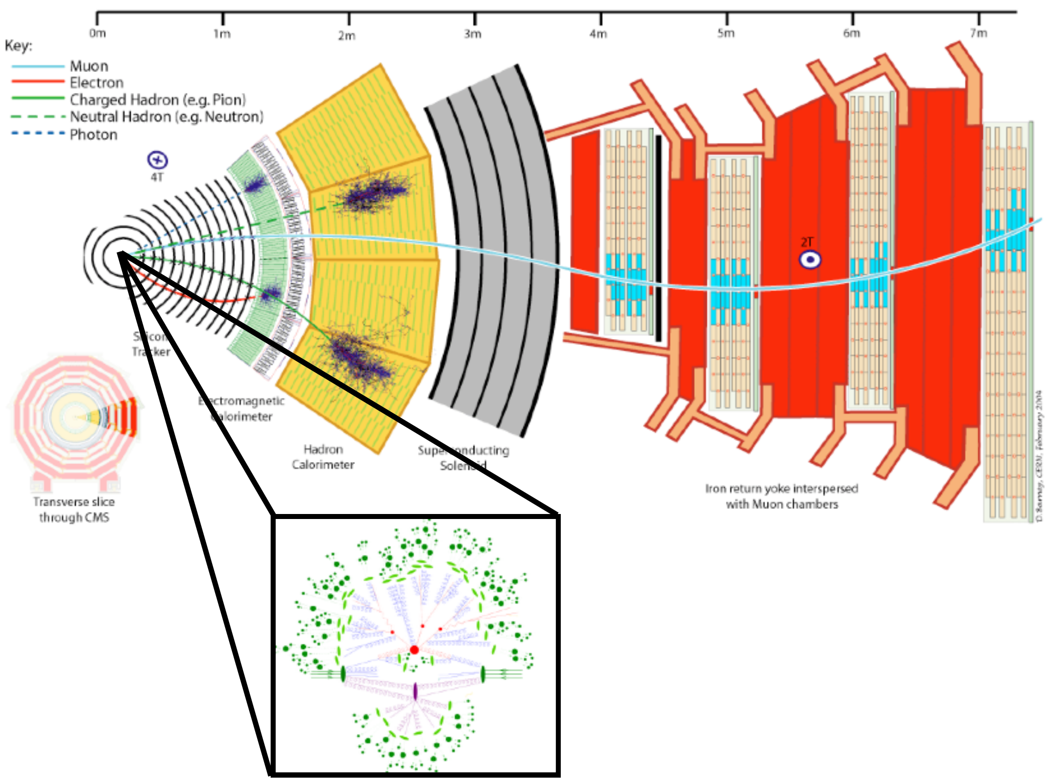


- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

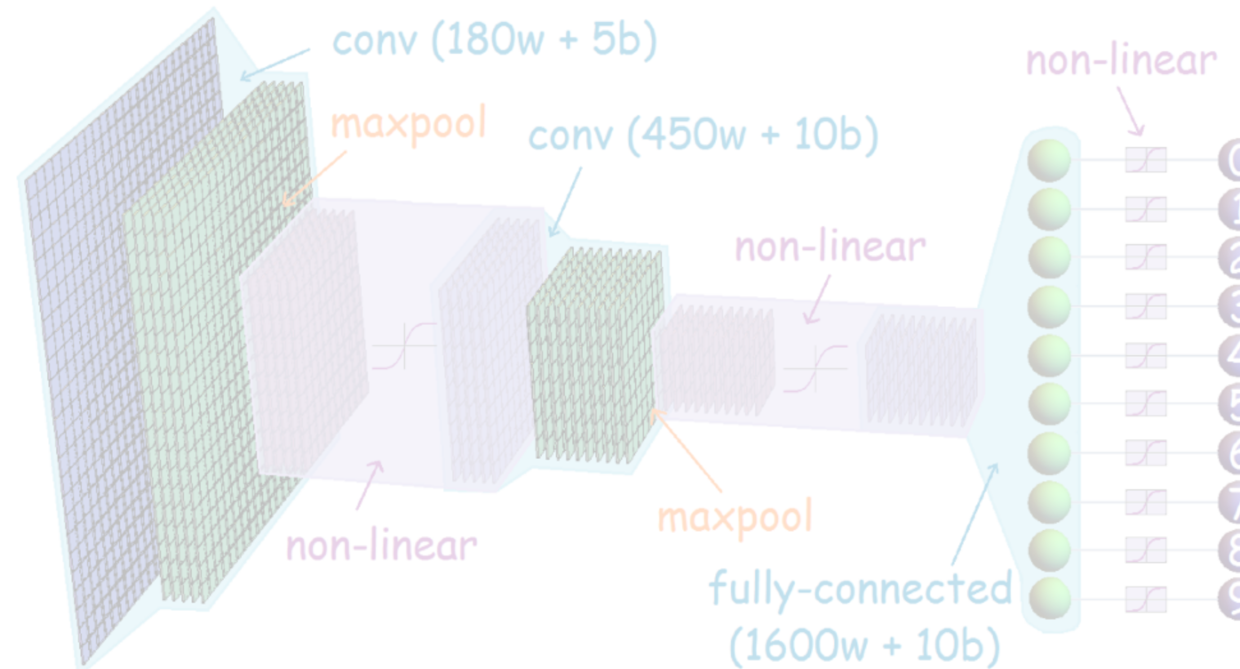
TWO APPROACHES

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]

‘Likelihood-Free’ Inference

$\leftarrow$ exact Bayesian Computation

Rejection Algorithm

- Draw θ from prior $\pi(\cdot)$
- Accept θ with probability $\pi(D \mid \theta)$

Accepted θ are independent draws from the posterior distribution, $\pi(\theta \mid D)$.
If the likelihood, $\pi(D|\theta)$, is unknown:

‘Mechanical’ Rejection Algorithm

- Draw θ from $\pi(\cdot)$
- Simulate $X \sim f(\theta)$ from the computer model
- Accept θ if $D = X$, i.e., if computer output equals observation

The acceptance rate is $\int \mathbb{P}(D|\theta)\pi(\theta)\mathrm{d}\theta = \mathbb{P}(D)$.

*From Richard Wilkinson’s talk at Data science @LHC

Rejection ABC

If $\mathbb{P}(D)$ is small (or D continuous), we will rarely accept any θ. Instead, there is an approximate version:

Uniform Rejection Algorithm

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(D, X) \leq \epsilon$

ϵ reflects the tension between computability and accuracy.

- As $\epsilon \to \infty$, we get observations from the prior, $\pi(\theta)$.
- If $\epsilon = 0$, we generate observations from $\pi(\theta \mid D)$.

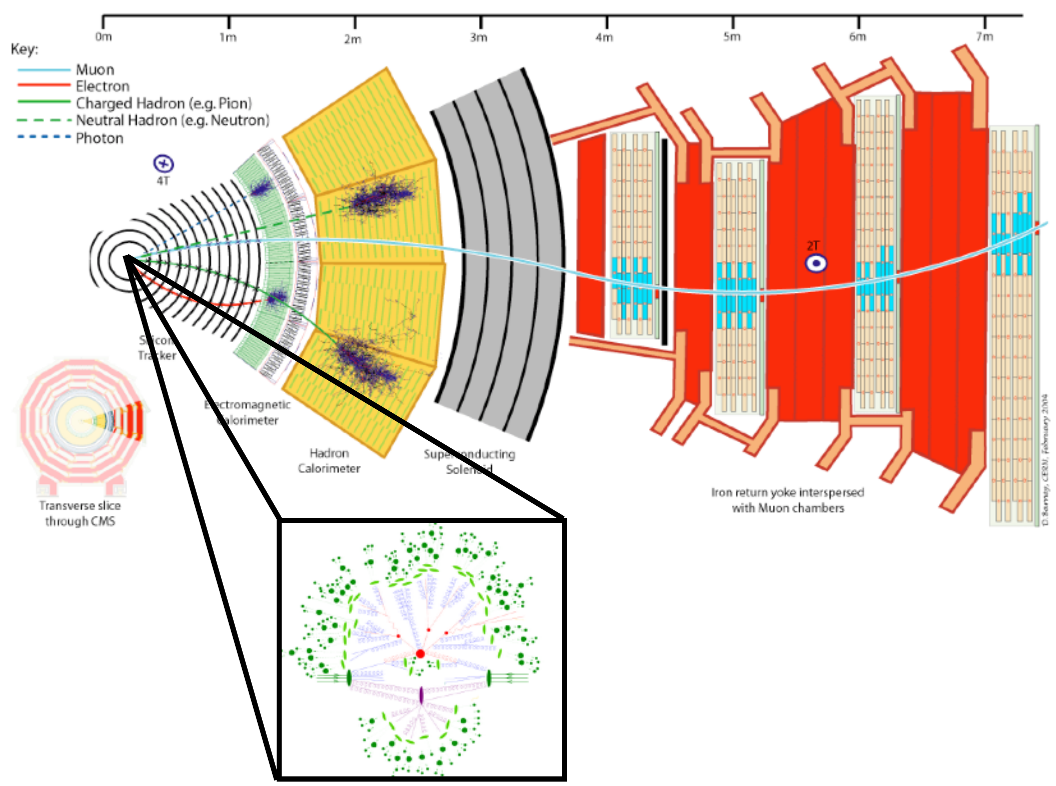
For reasons that will become clear later, we call this *uniform-ABC*.

*From Richard Wilkinson's talk at Data science @LHC

TWO APPROACHES

Use simulator

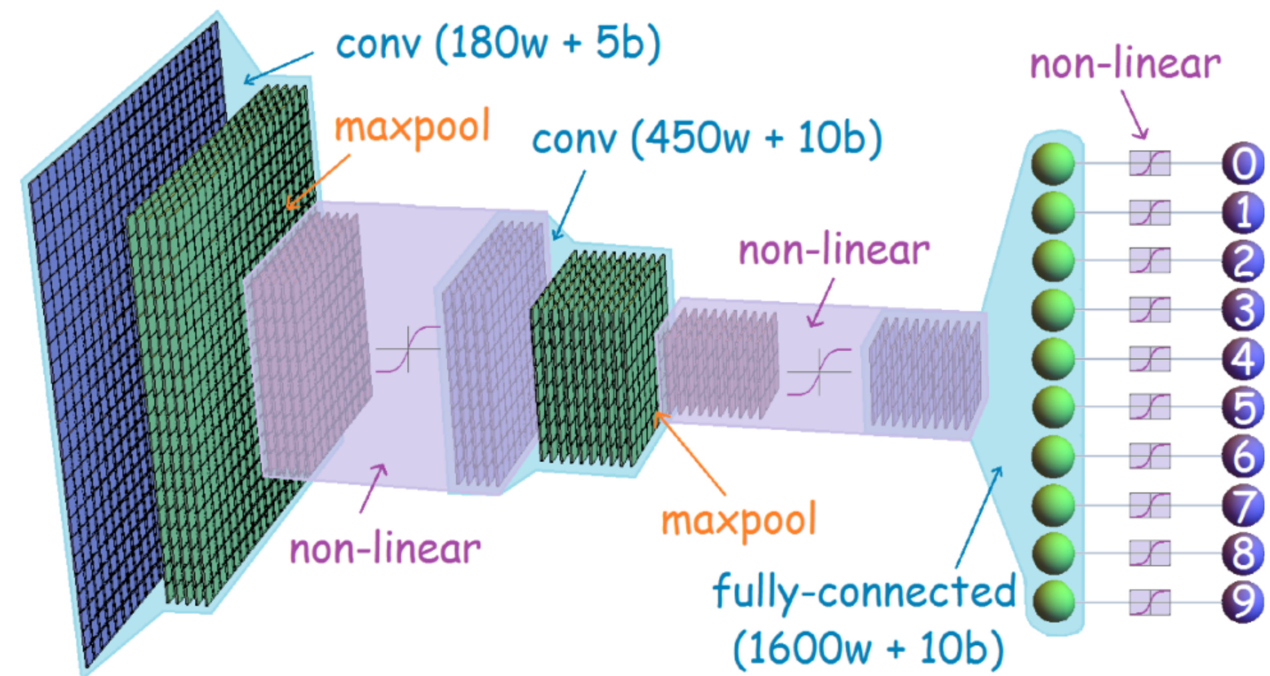
(much more efficiently)



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

Learn simulator

(with deep learning)



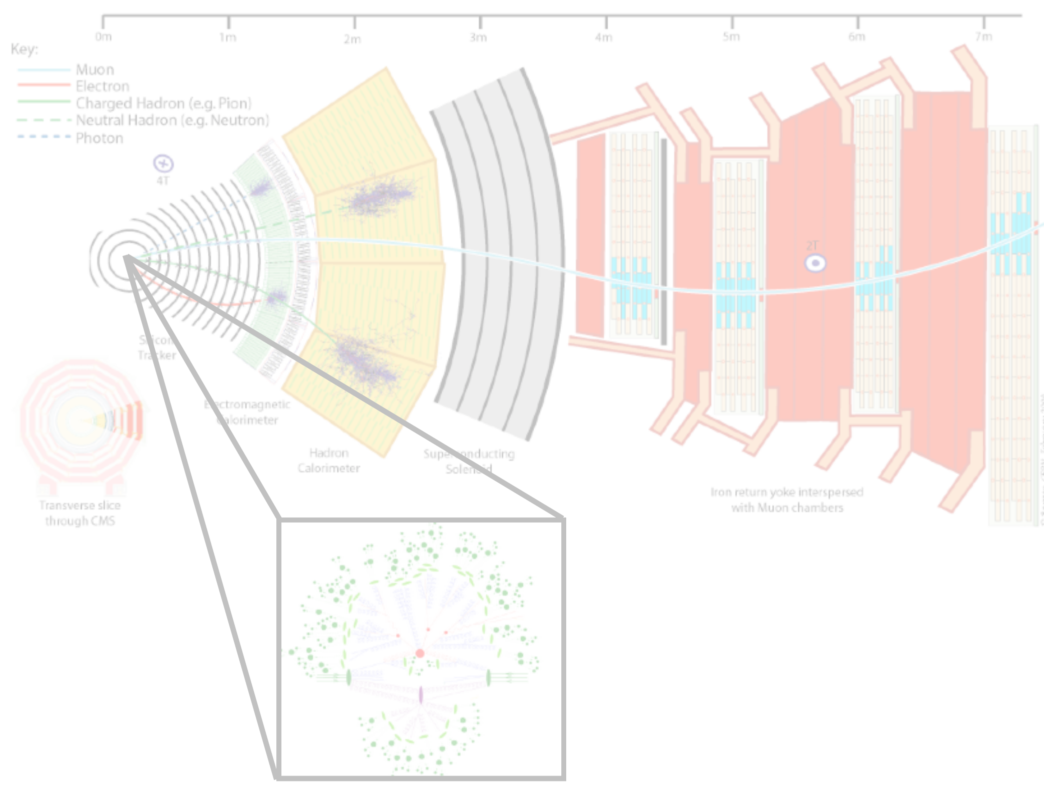
- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

TWO APPROACHES

Use simulator

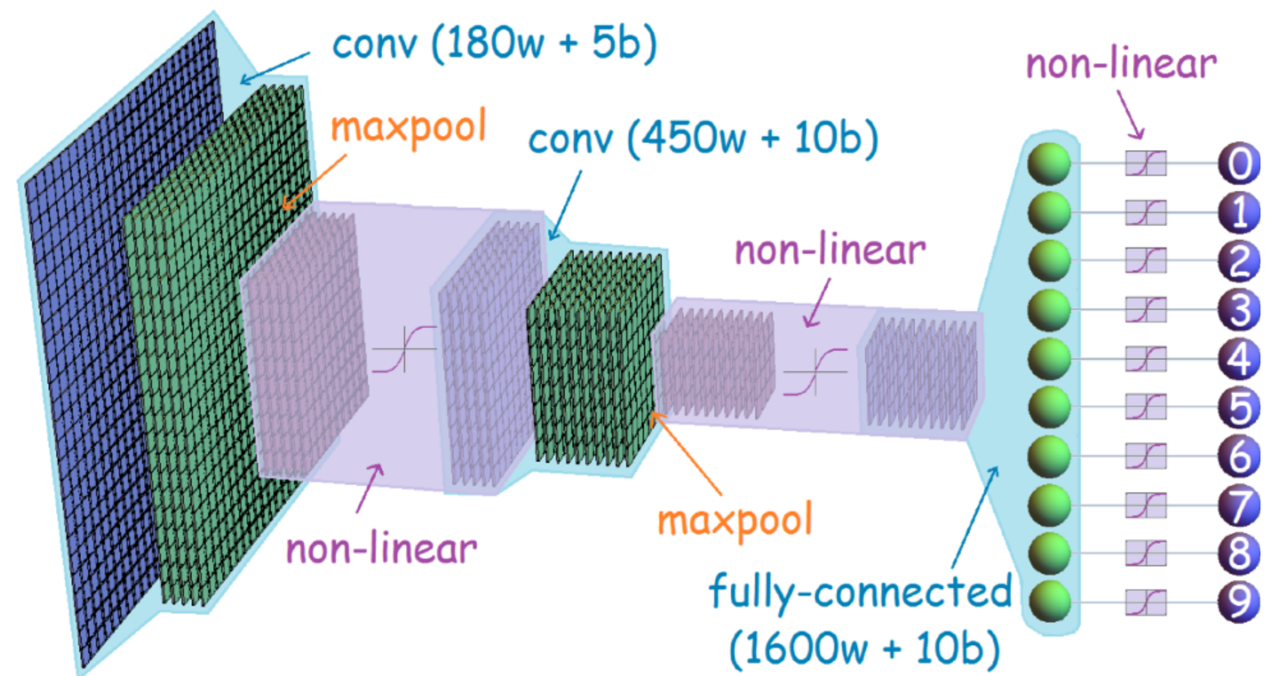
(much more efficiently)



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

Learn simulator

(with deep learning)

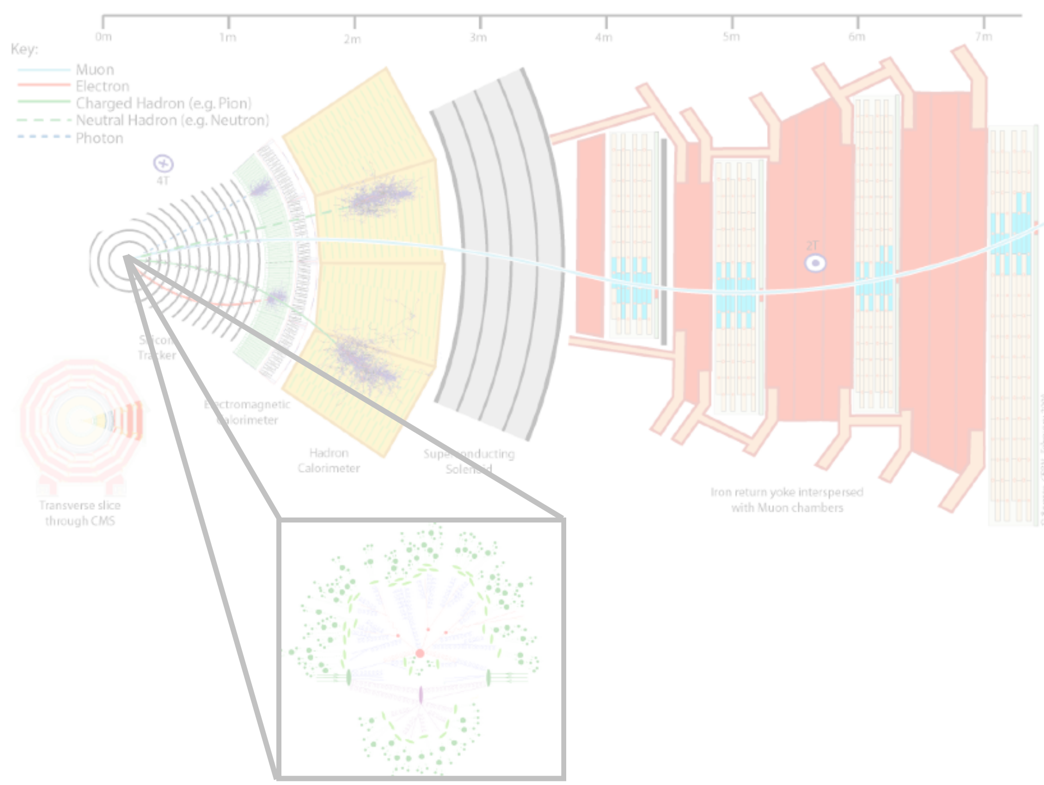


- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

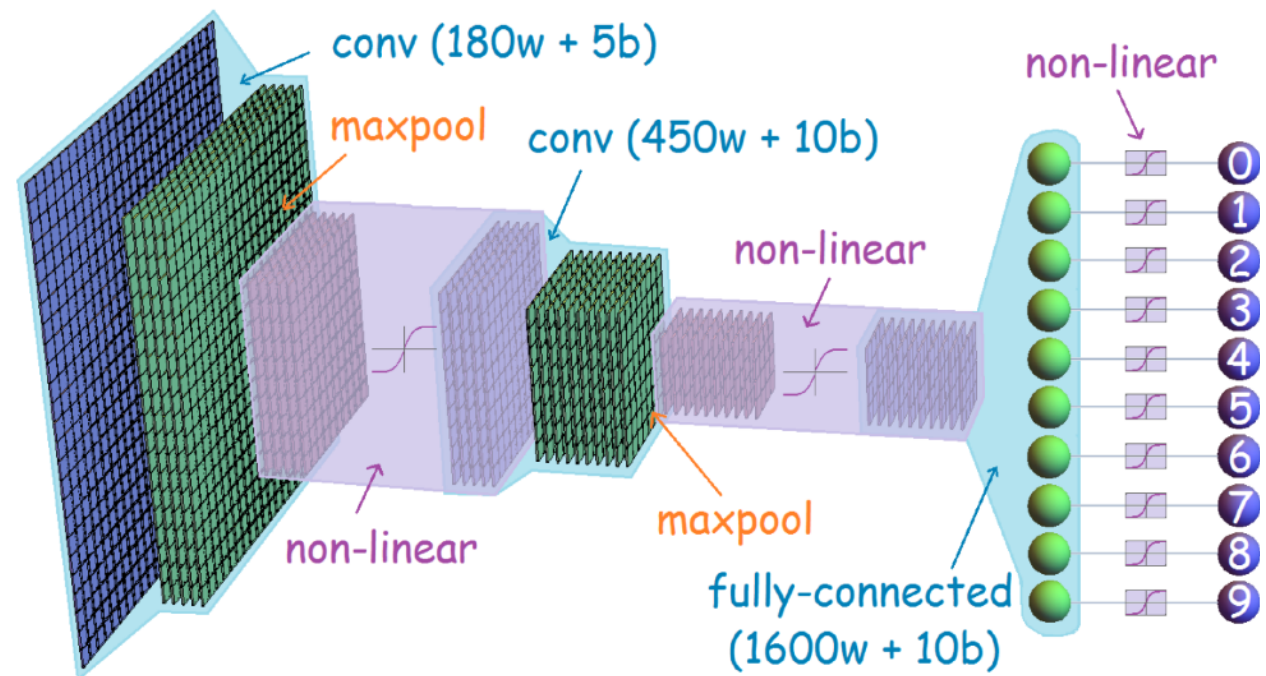
TWO APPROACHES

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]

THE PLAYERS

forward modeling
generation
simulation

PREDICTION

θ
parameters of interest

$p(x, z \mid \theta, \nu)$

z
latent variables
Monte Carlo truth

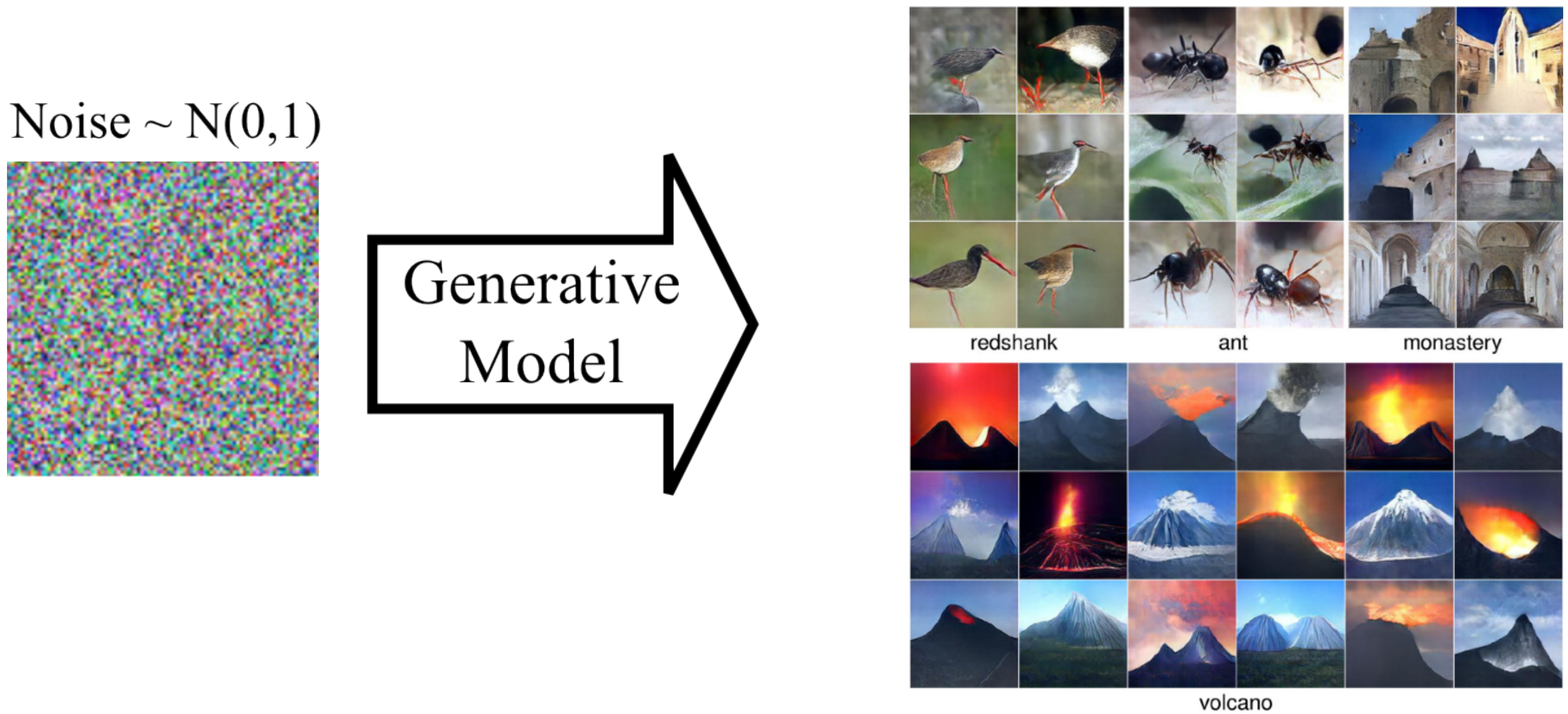
x
observed data
simulated data

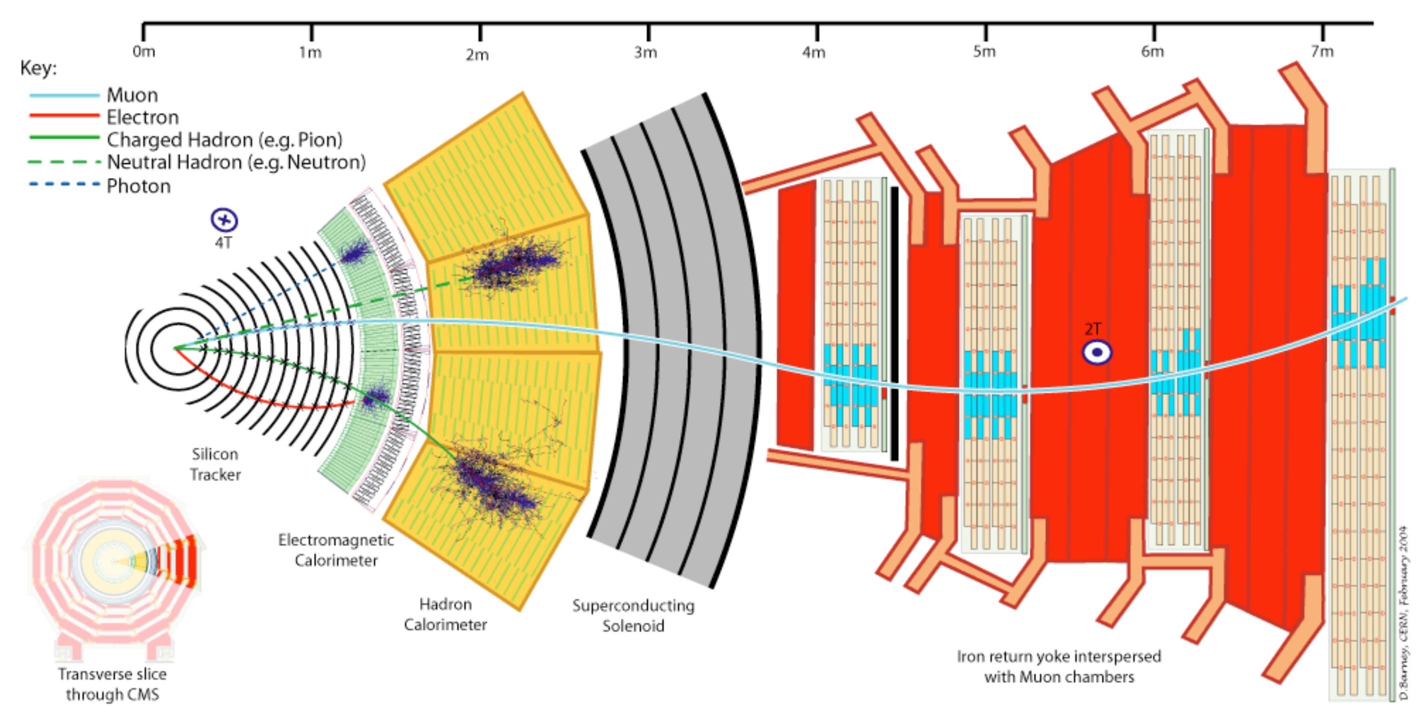
ν
nuisance parameters

INFERENCE

inverse problem
measurement
parameter estimation

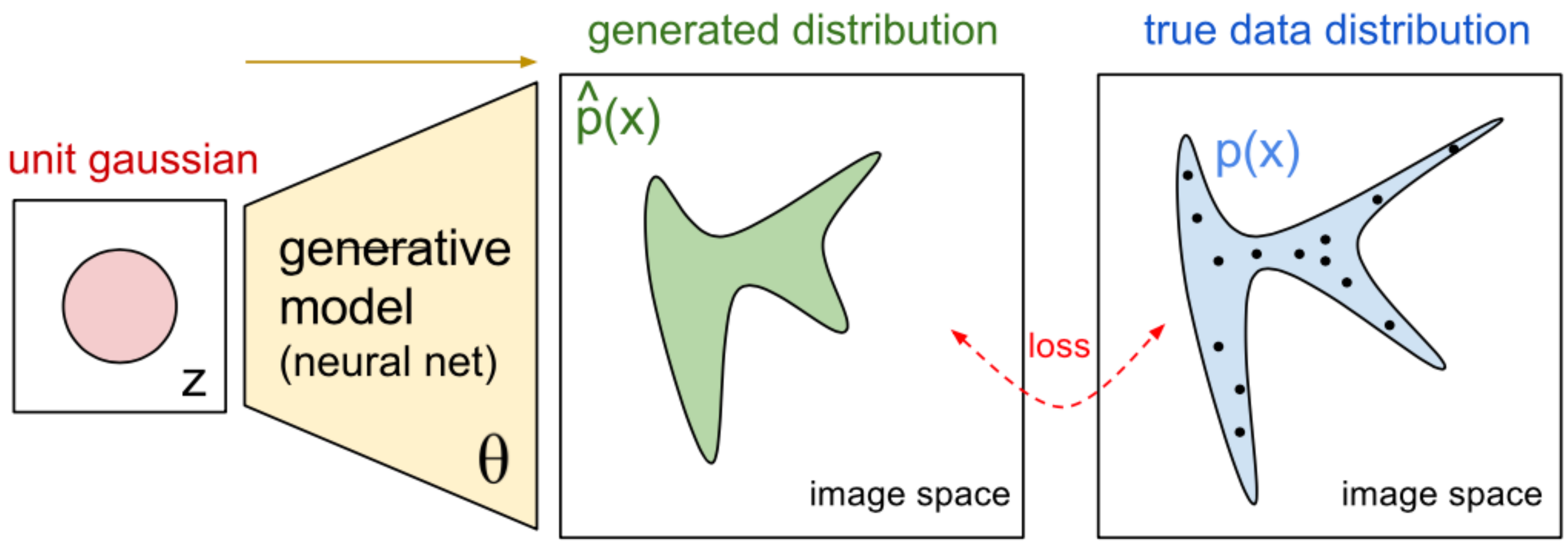
LEARNING THE GENERATIVE MODEL





GENERATIVE ADVERSARIAL NETWORKS

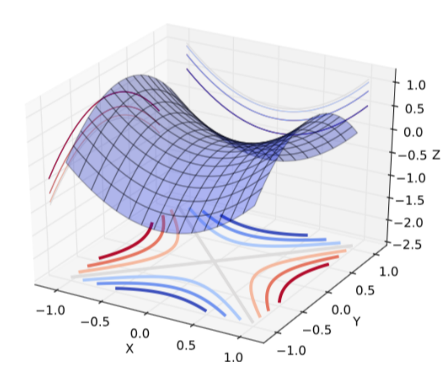
Goodfellow, et al arXiv:1406.2661
slide Gilles Louppe



Leo is G Tom is D

- Two-player game:
  - a discriminator D,
  - a generator G;
- D is a classifier $\mathcal{X} \mapsto \{0, 1\}$ that tries to distinguish between
  - a sample from the data distribution ($D(\mathbf{x}) = 1$, for $\mathbf{x} \sim p_{\text{data}}$),
  - and a sample from the model distribution ($D(G(\mathbf{z})) = 0$, for $\mathbf{z} \sim p_{\text{noise}}$);
- G is a generator $\mathcal{Z} \mapsto \mathcal{X}$ trained to produce samples $G(\mathbf{z})$ (for $\mathbf{z} \sim p_{\text{noise}}$) that are difficult for D to distinguish from data.

$$(D^*, G^*) = \max_D \min_G V(D, G).$$

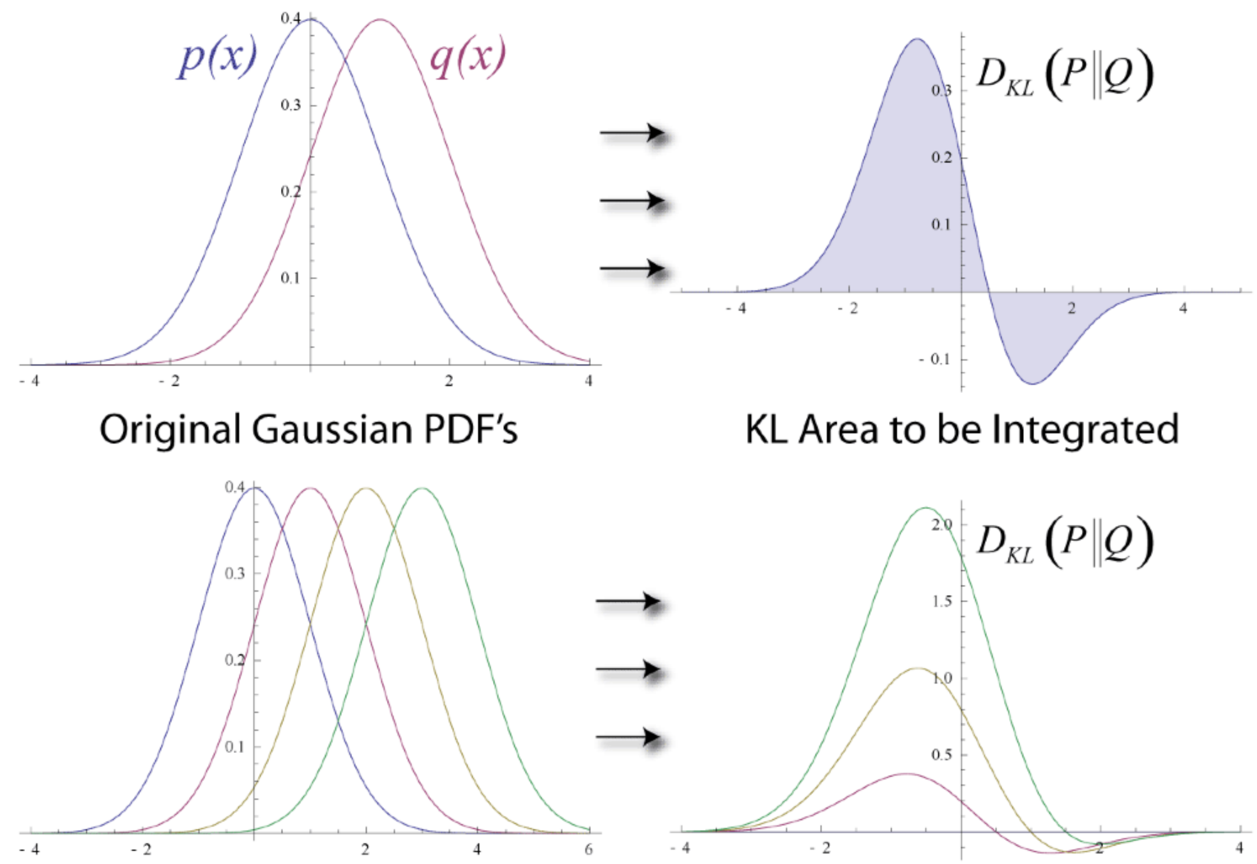


KULLBACK–LEIBLER DIVERGENCE

$$D_{\mathrm{KL}}(P\|Q) = \int_{x_a}^{x_b} P(x)\log\left(\frac{P(x)}{Q(x)}\right)dx = \int_{y_a}^{y_b} P(y)\log\left(\frac{P(y)dy/dx}{Q(y)dy/dx}\right)dy = \int_{y_a}^{y_b} P(y)\log\left(\frac{P(y)}{Q(y)}\right)dy$$

$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= -\textstyle\sum_x p(x)\log q(x) \quad + \quad \sum_x p(x)\log p(x) \\ &= H(P,Q) \quad - \quad H(P) \end{aligned}$$

where $H(P,Q)$ is the cross entropy of P and Q, and $H(P)$ is the entropy of P.



- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x) \ ,$$

where both $\mathbb{P}_r$ and $\mathbb{P}_g$ are assumed to be absolutely continuous, and therefore admit densities, with respect to a same measure μ defined on $\mathcal{X}$.[2] The KL divergence is famously assymetric and possibly infinite when there are points such that $P_g(x) = 0$ and $P_r(x) > 0$.

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ ,$$

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$.

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \ , \qquad (1)$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

Dual Description

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \le 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks

Michela Paganini[a,b], **Luke de Oliveira**[a], **and Benjamin Nachman**[a]

[a] *Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA, 94720, USA*
[b] *Department of Physics, Yale University, New Haven, CT 06520, USA*

E-mail: michela.paganini@yale.edu, lukedeoliveira@lbl.gov, bnachman@cern.ch

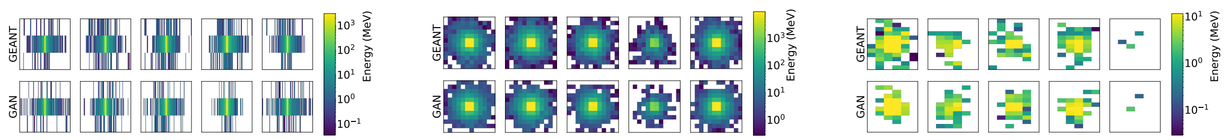


Figure 9: Five randomly selected e^+ showers per calorimeter layer from the training set (top) and the five nearest neighbors (by euclidean distance) from a set of CaloGAN candidates.

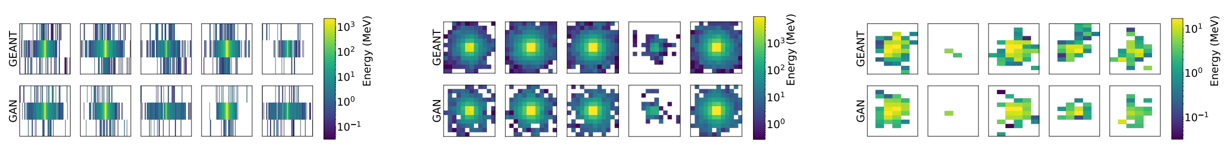


Figure 10: Five randomly selected γ showers per calorimeter layer from the training set (top) and the five nearest neighbors (by euclidean distance) from a set of CaloGAN candidates.

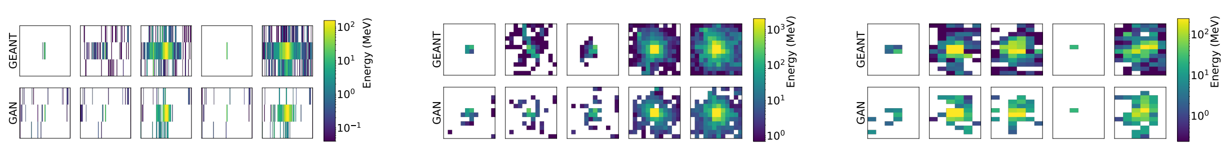
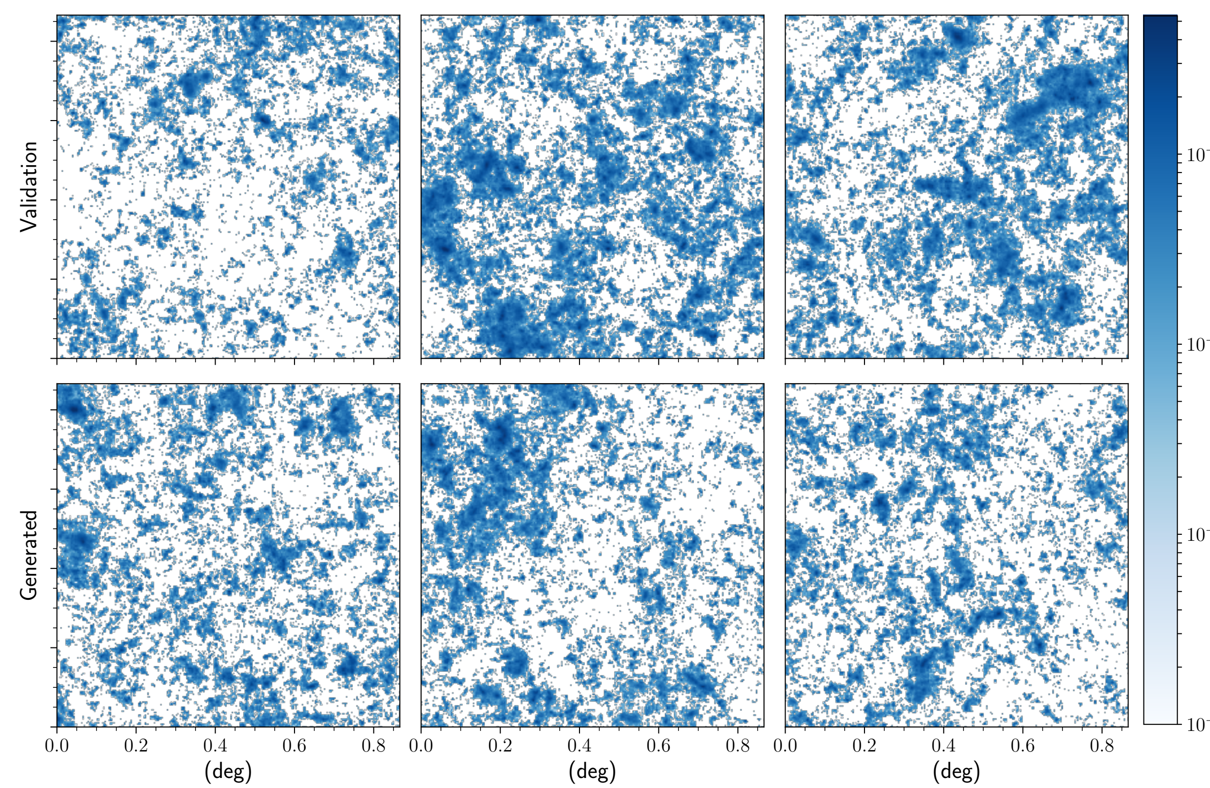


Figure 11: Five randomly selected π^+ showers per calorimeter layer from the training set (top) and the five nearest neighbors (by euclidean distance) from a set of CaloGAN candidates.

Creating Virtual Universes Using Generative Adversarial Networks

Mustafa Mustafa*[1], Deborah Bard[1], Wahid Bhimji[1], Rami Al-Rfou[2], and Zarija Lukić[1]

[1] Lawrence Berkeley National Laboratory, Berkeley, CA 94720
[2] Google Research, Mountain View, CA 94043



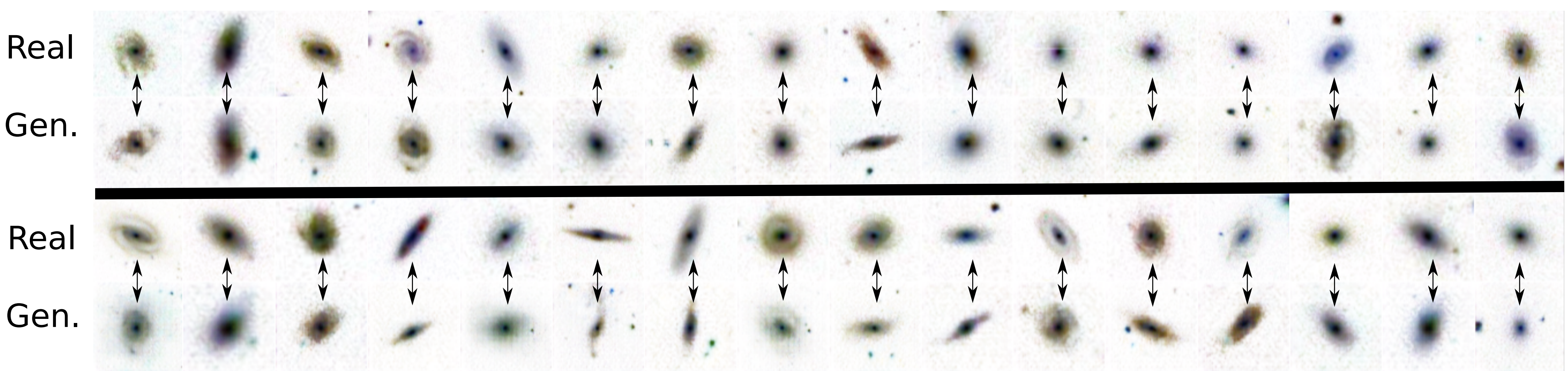
Use of generative models of galaxy images to help calibrate down-stream analysis in next-generation surveys.

Enabling Dark Energy Science with Deep Generative Models of Galaxy Images

Siamak Ravanbakhsh[1], François Lanusse[2], Rachel Mandelbaum[2], Jeff Schneider[1], and Barnabás Póczos[1]

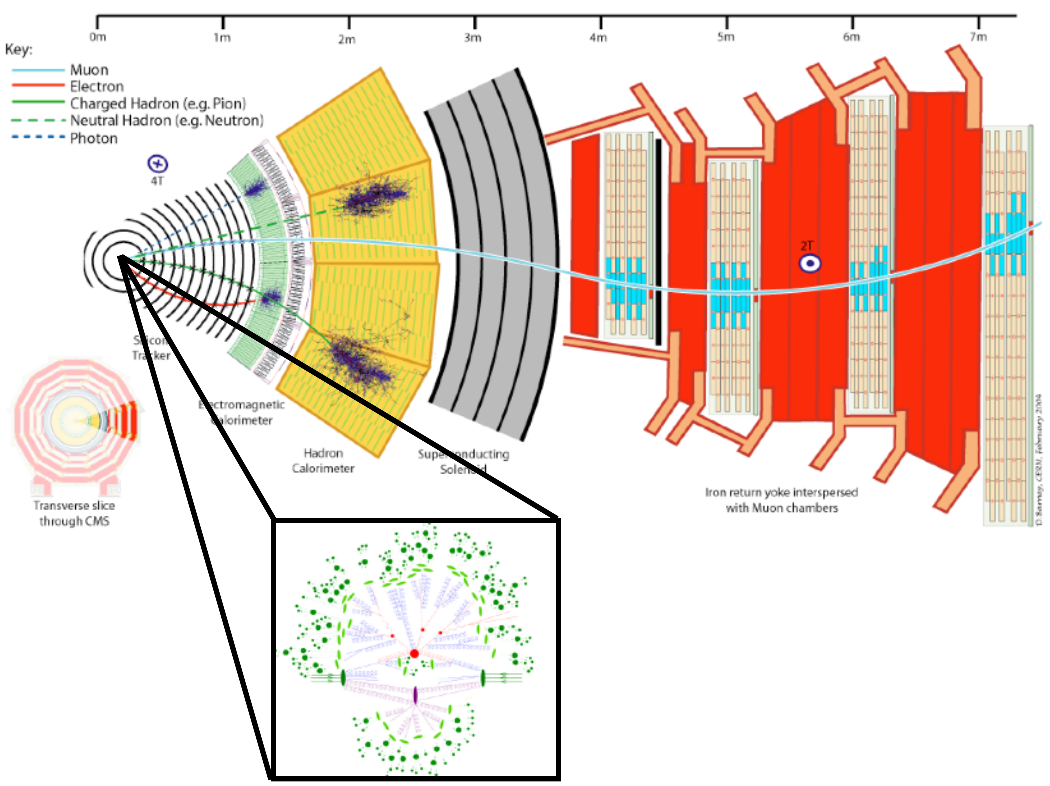
[1]School of Computer Science, Carnegie Mellon University
[2]McWilliams Center for Cosmology, Carnegie Mellon University

***Abstract*—Understanding the nature of dark energy, the mysterious force driving the accelerated expansion of the Universe, is a major challenge of modern cosmology. The next generation of cosmological surveys, specifically designed to address this issue, rely on accurate measurements of the apparent shapes of distant galaxies. However, shape measurement methods suffer from various unavoidable biases and therefore will rely on a precise calibration to meet the accuracy requirements of the science analysis. This calibration process remains an open challenge as it requires large sets of high quality galaxy images. To this end, we study the application of deep conditional generative models in generating realistic galaxy images. In particular we consider variations on conditional variational autoencoder and introduce a new adversarial objective for training of conditional generative networks. Our results suggest a reliable alternative to the acquisition of expensive high quality observations for generating the calibration data needed by the next generation of cosmological surveys.**

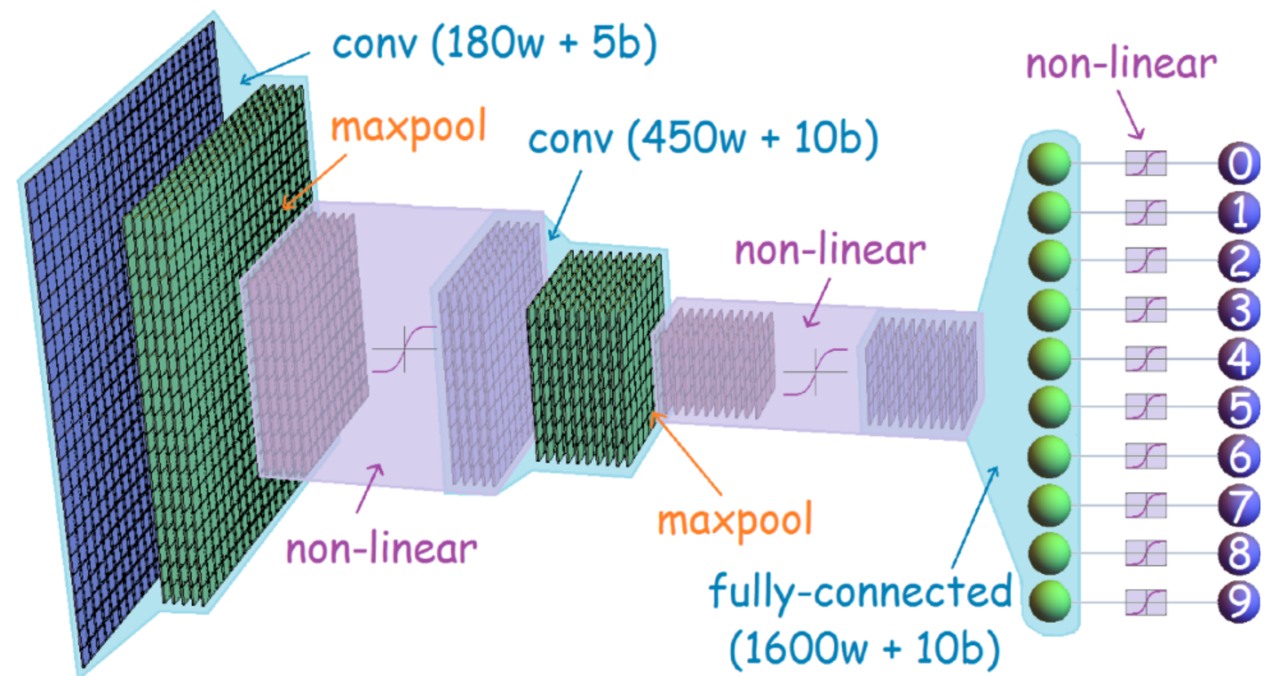


TWO APPROACHES

Use simulator
(much more efficiently)



Learn simulator
(with deep learning)



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

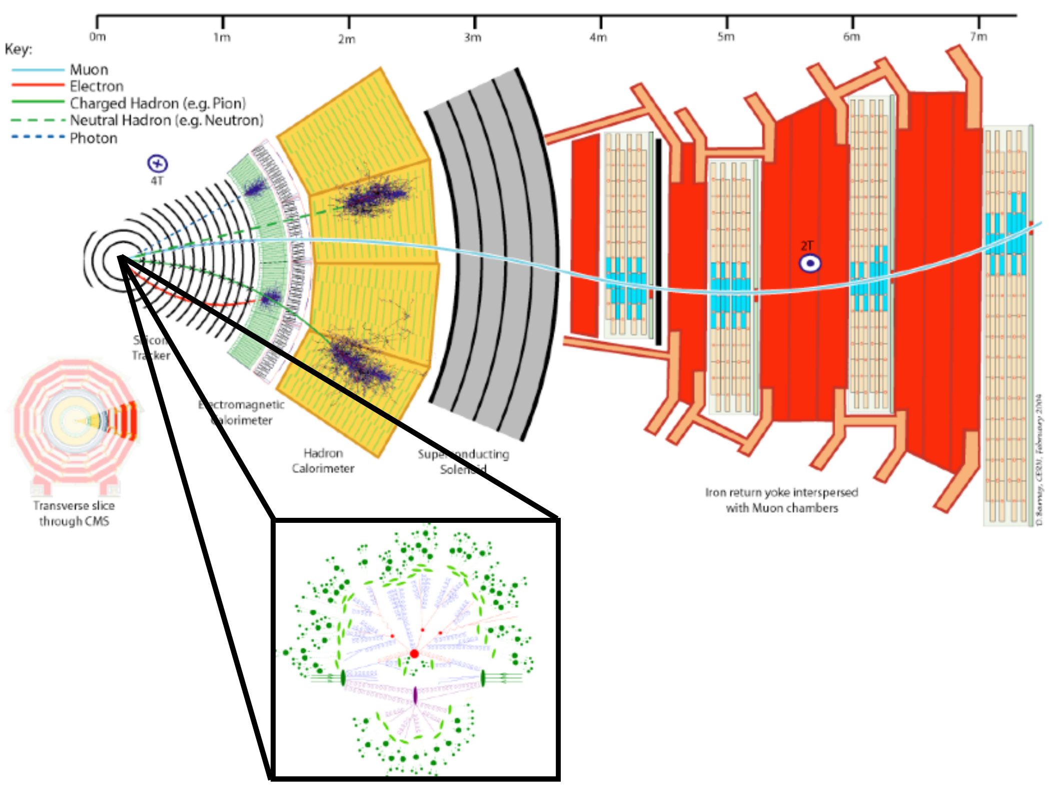
- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

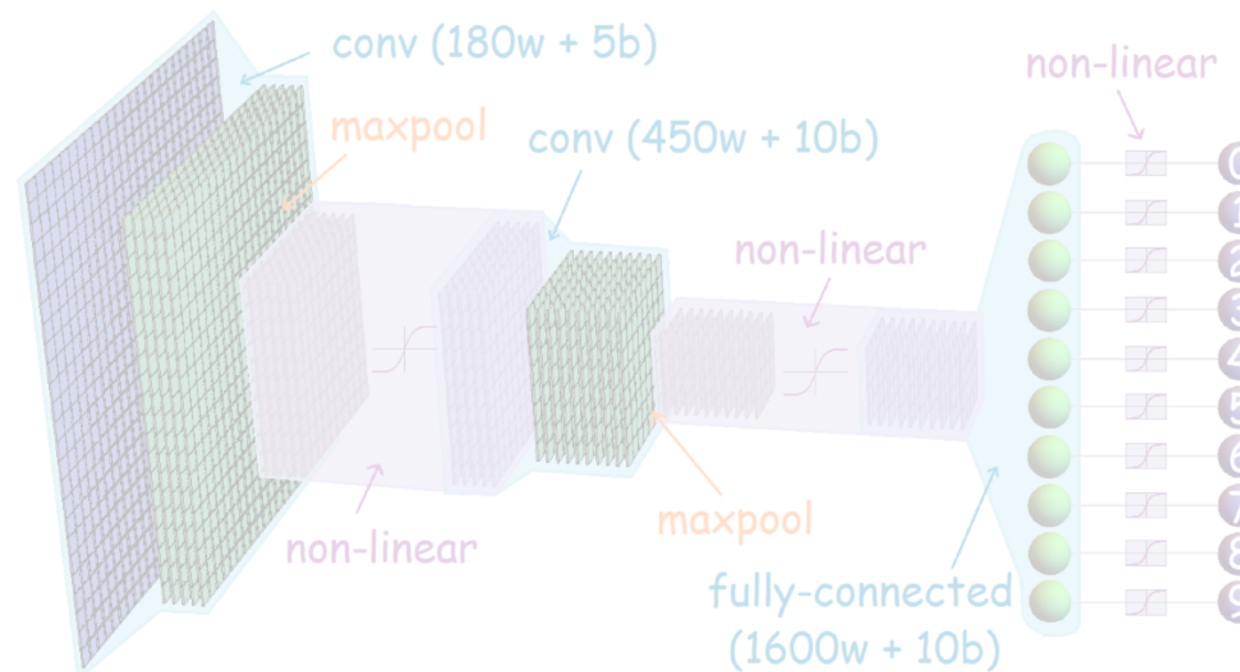
[image credit: A.P. Goucher]

TWO APPROACHES

Use simulator
(much more efficiently)

Learn simulator
(with deep learning)





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

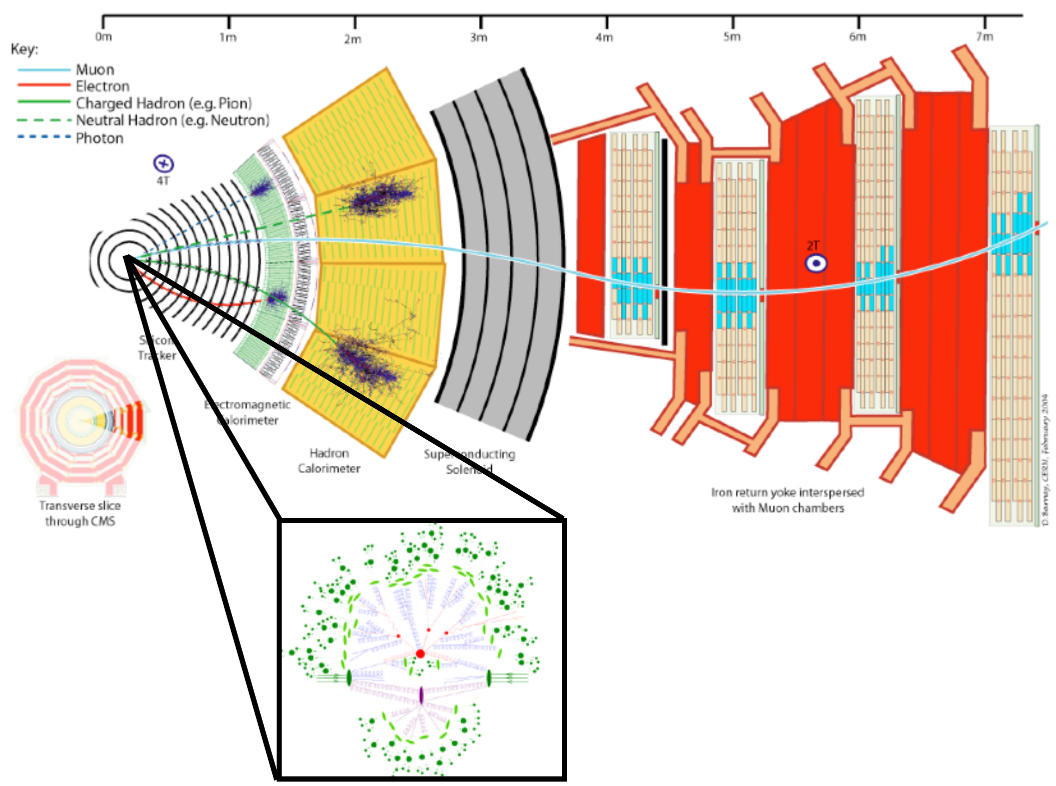
- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

TWO APPROACHES

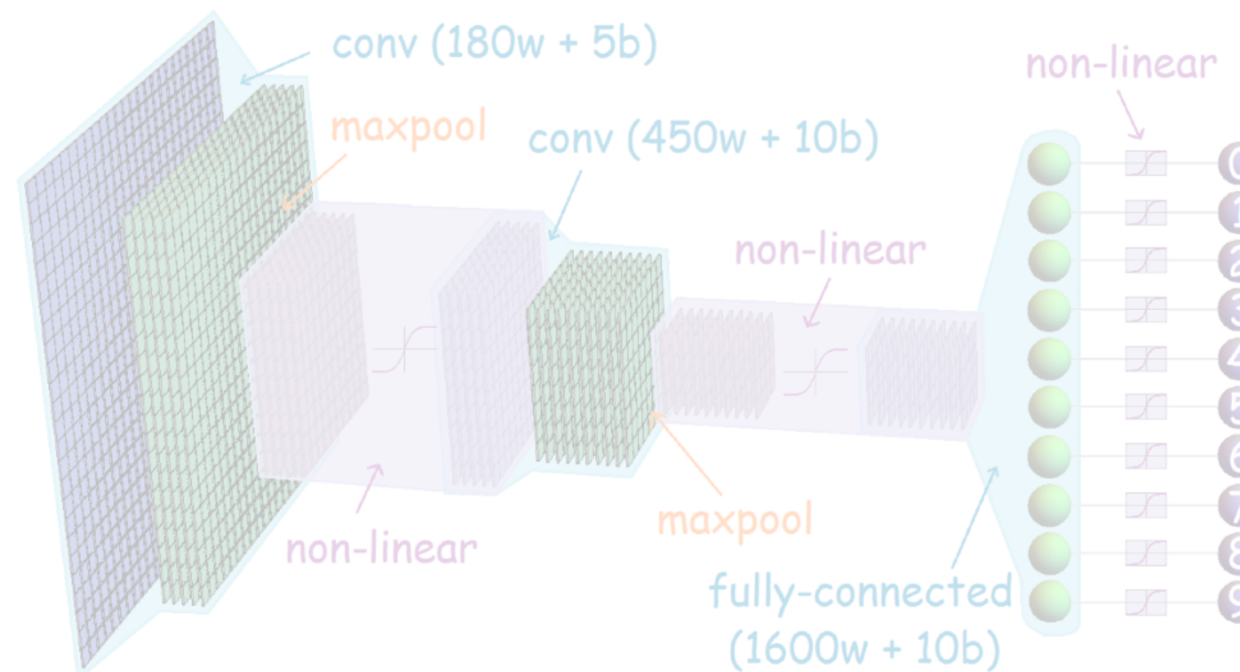
Use simulator

(much more efficiently)



Learn simulator

(with deep learning)



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

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

[image credit: A.P. Goucher]

NEW! AVO

Adversarial Variational Optimization of Non-Differentiable Simulators

Gilles Louppe[1] and **Kyle Cranmer**[1]
[1]*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

Continue to use a neural network discriminator / critic.

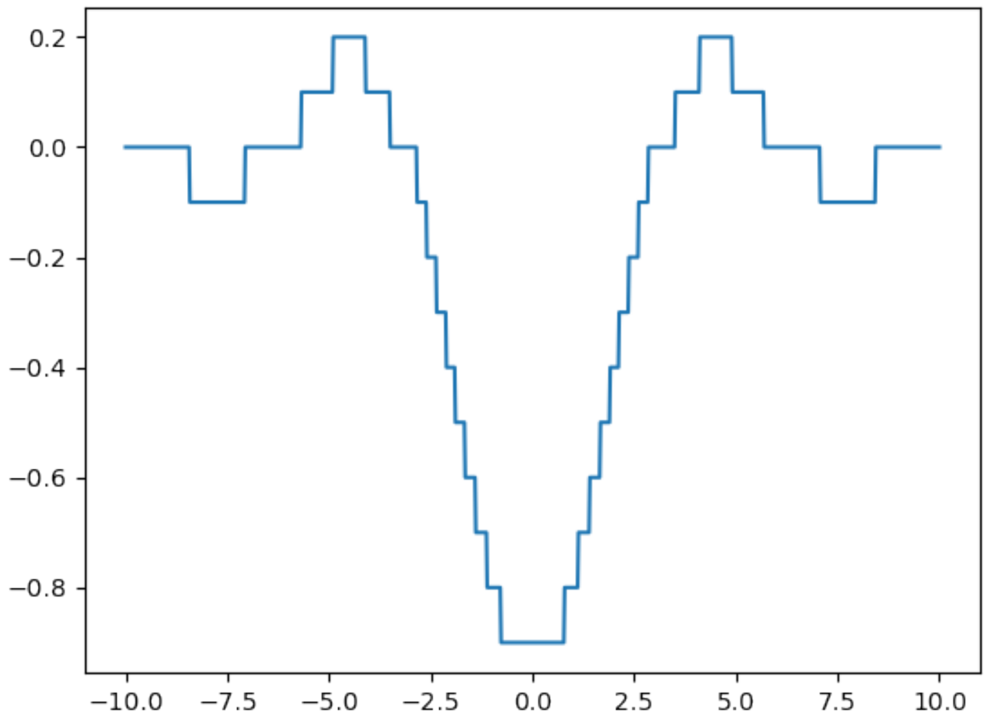
Difficulty: the simulator isn't differentiable, but there's a **trick**!

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

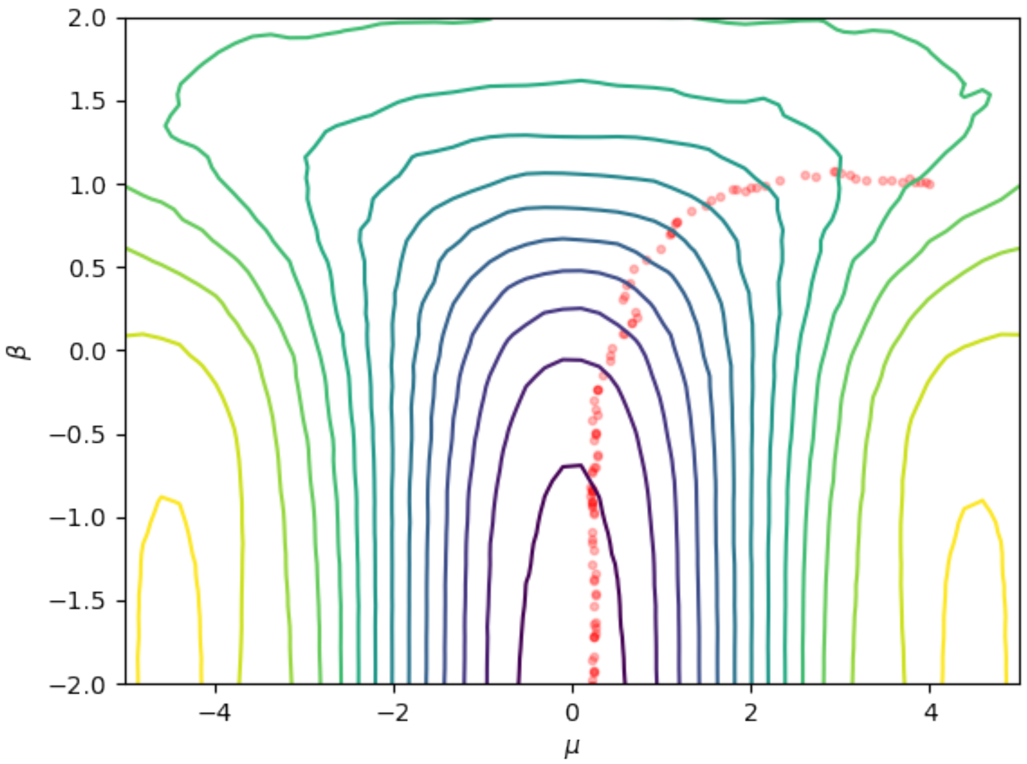
VARIATIONAL OPTIMIZATION

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

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



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



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

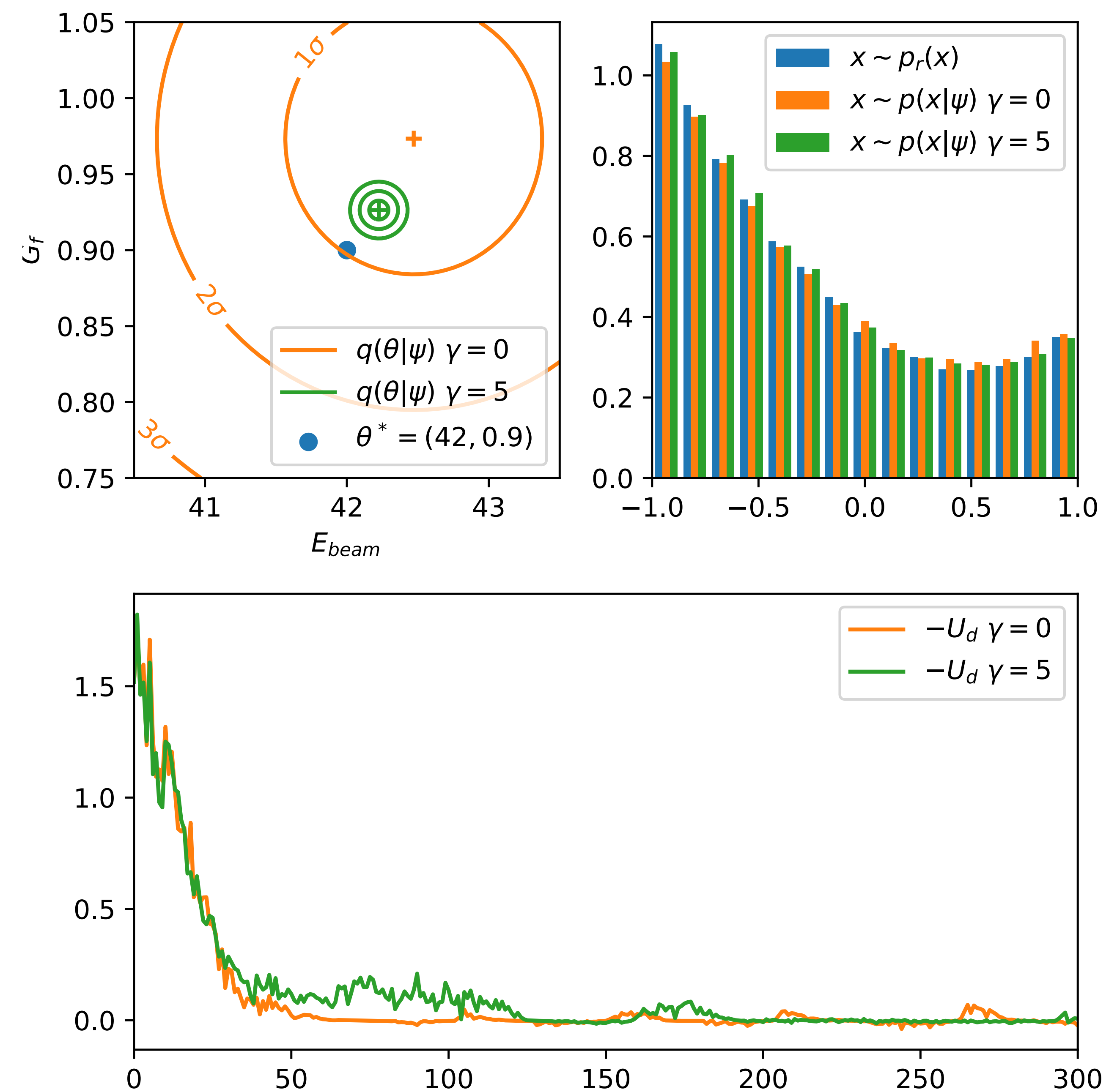
Like a GAN, but generative model is non-differentiable and the parameters of simulator have meaning

- Replace the generative network with a non-differentiable forward simulator $g(\mathbf{z};\boldsymbol{\theta})$.
- With VO, optimize upper bounds of the adversarial objectives:

$$U_d = \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\boldsymbol{\psi})}[\mathcal{L}_d] \quad (1)$$

$$U_g = \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\boldsymbol{\psi})}[\mathcal{L}_g] \quad (2)$$

respectively over $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$.



Effectively sampling from marginal model

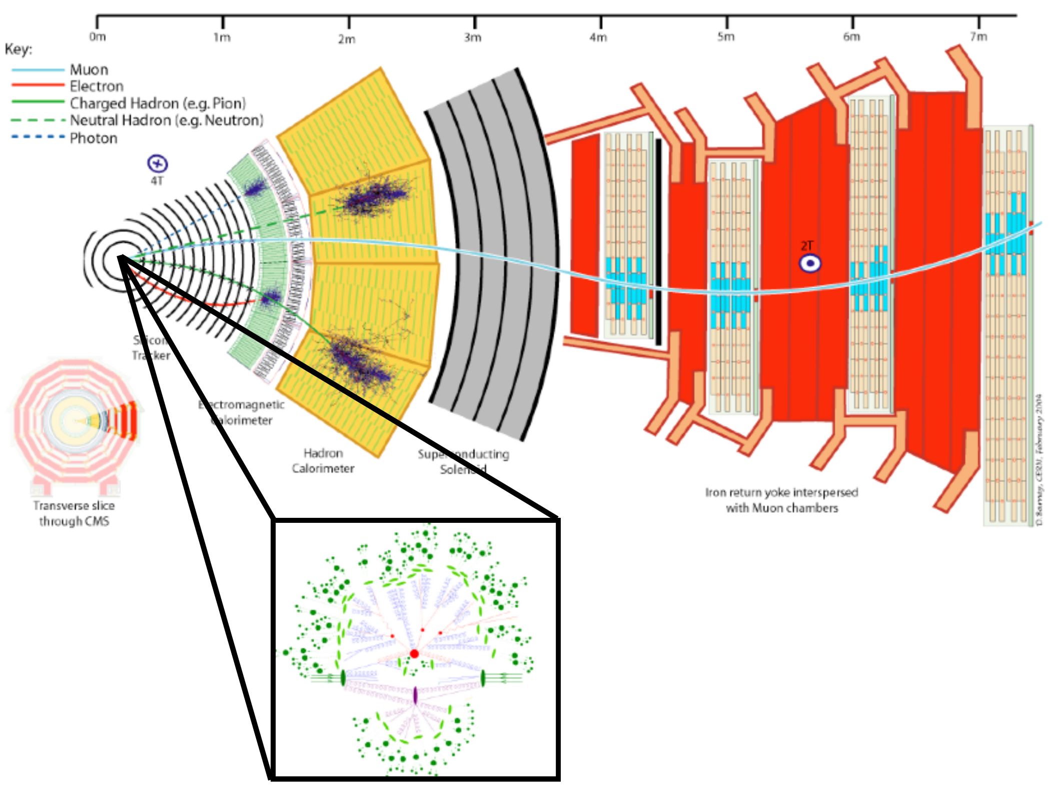
$$\mathbf{x} \sim q(\mathbf{x}|\boldsymbol{\psi}) \equiv \boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\boldsymbol{\psi}), \mathbf{z} \sim p(\mathbf{z}|\boldsymbol{\theta}), \mathbf{x} = g(\mathbf{z};\boldsymbol{\theta})$$

We use Wasserstein distance, as in WGAN

G. Louppe & K.C. arXiv:1707.07113

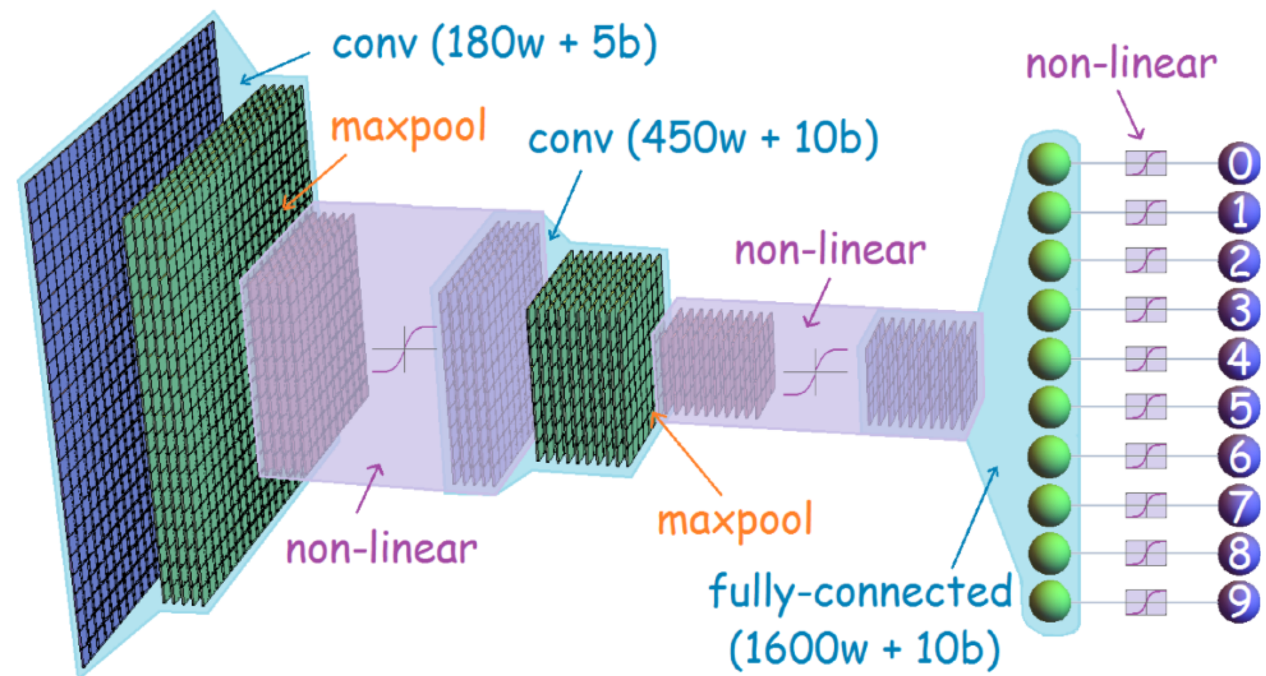
TWO APPROACHES

Use simulator
(much more efficiently)



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

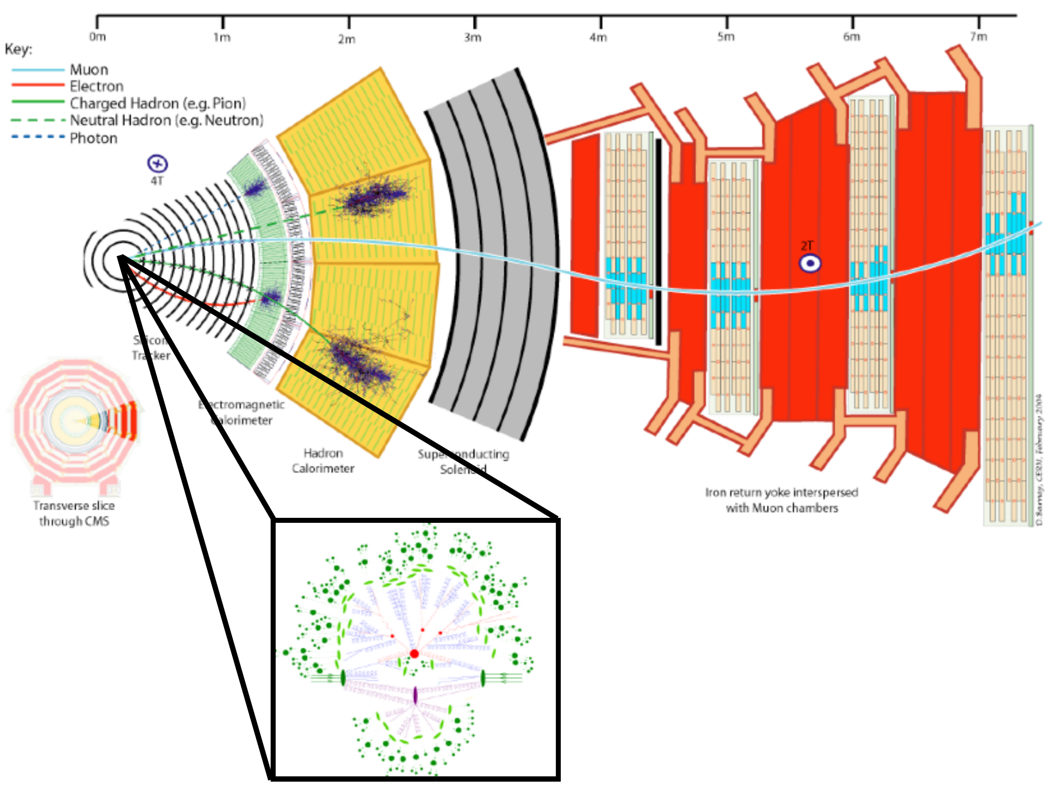
Learn simulator
(with deep learning)



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

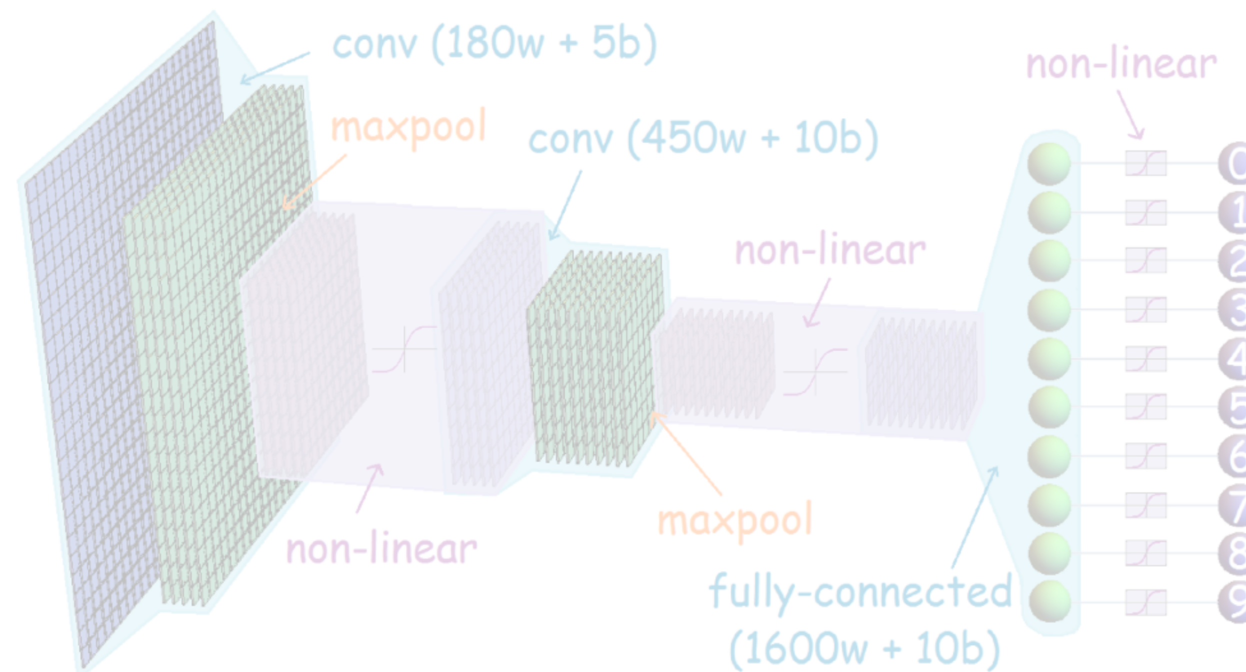
TWO APPROACHES

Use simulator
(much more efficiently)



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

Learn simulator
(with deep learning)



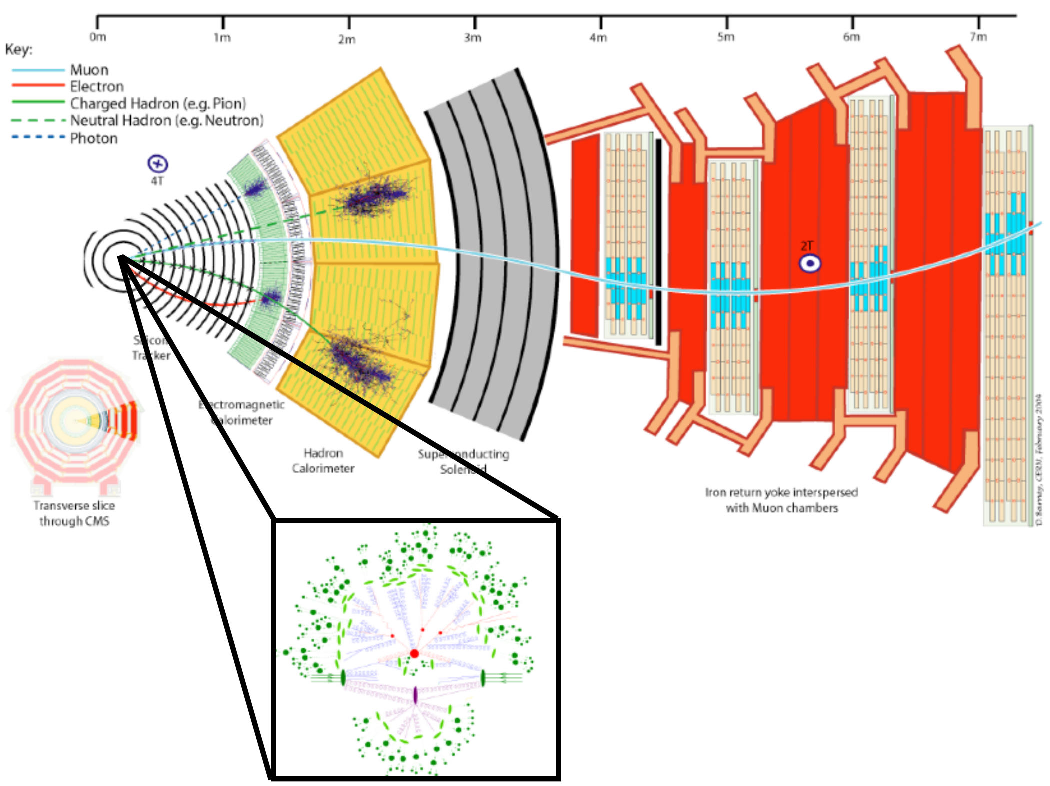
- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

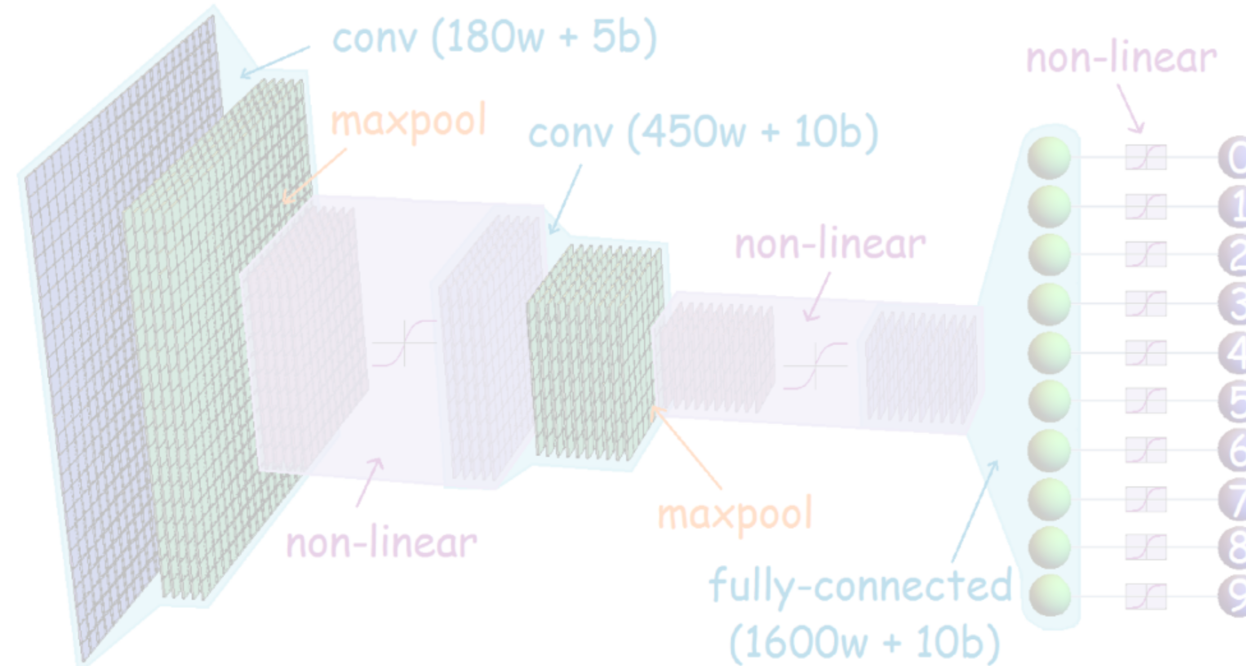
[image credit: A.P. Goucher]

TWO APPROACHES

Use simulator
(much more efficiently)

Learn simulator
(with deep learning)



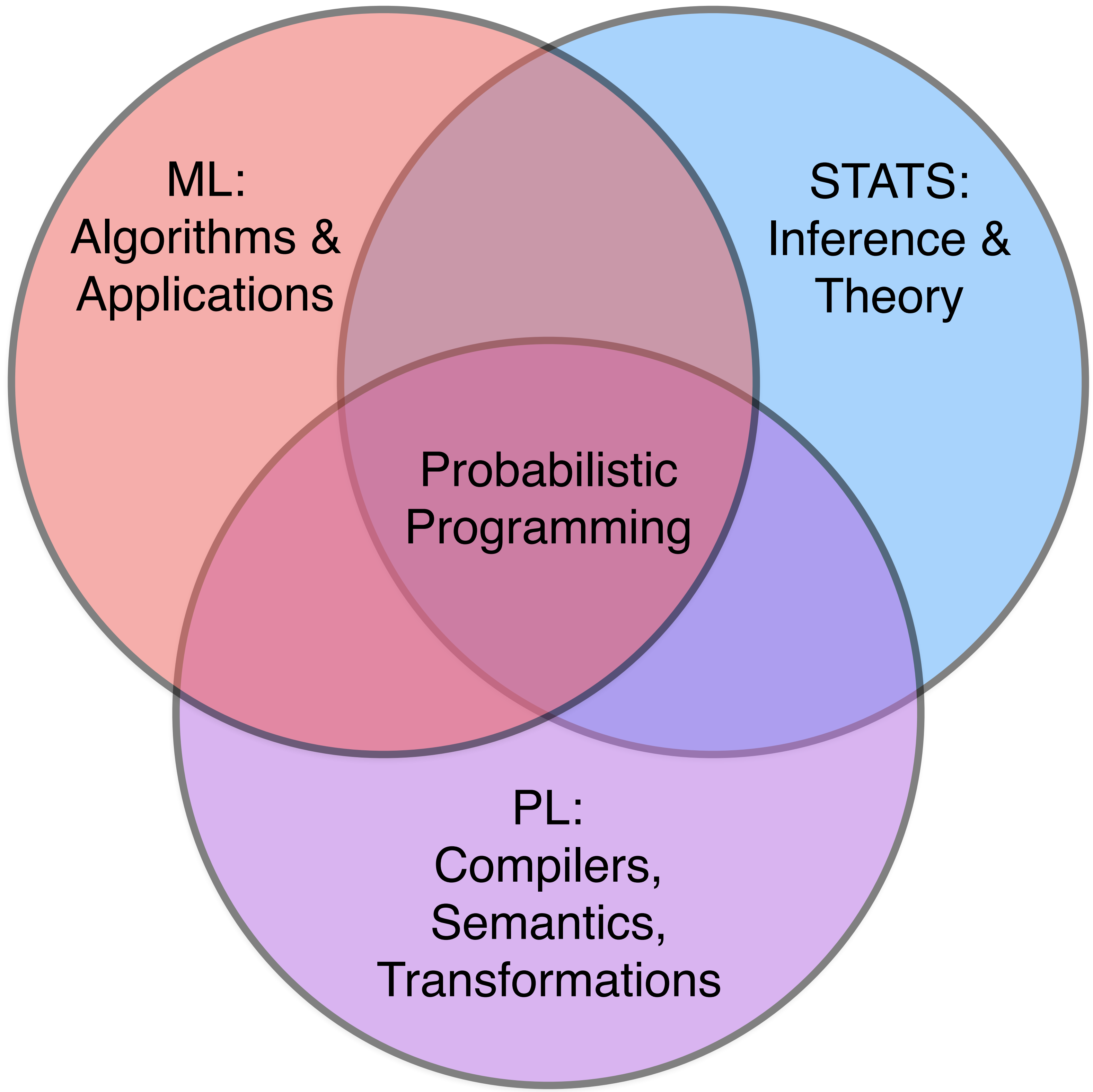


- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

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

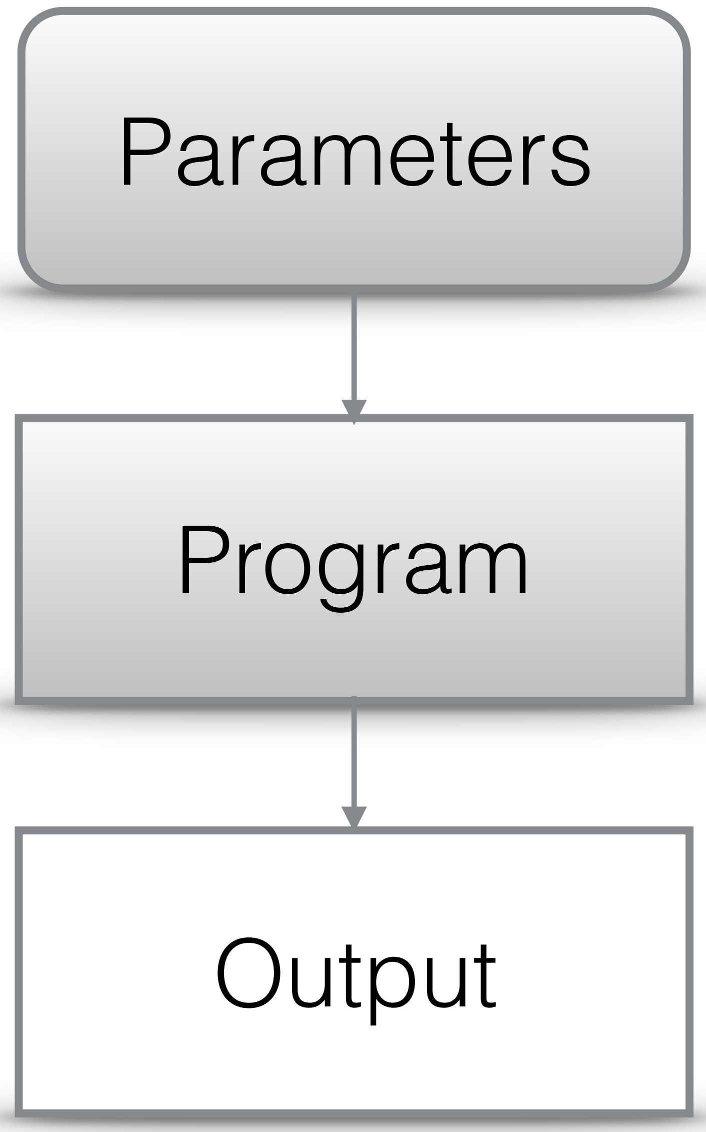
[image credit: A.P. Goucher]

Probabilistic Programming



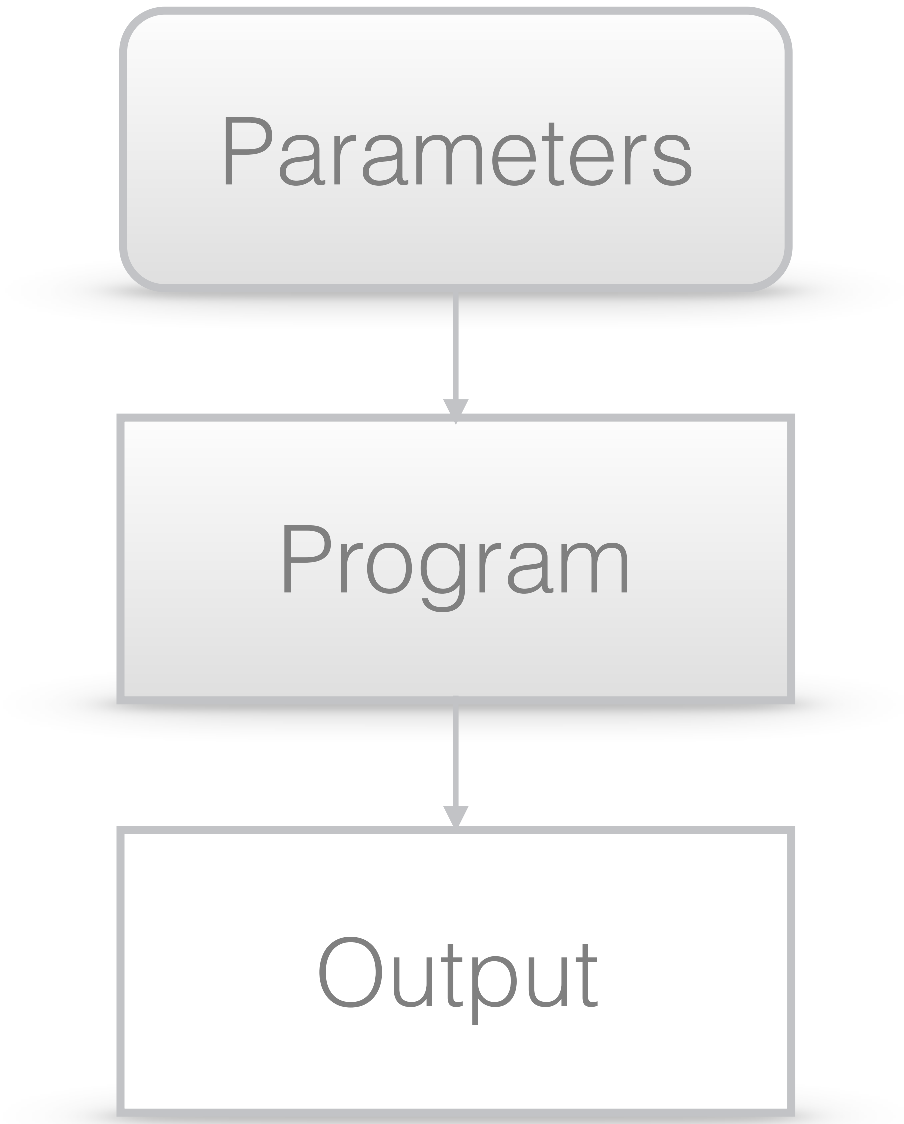
Intuition

Intuition

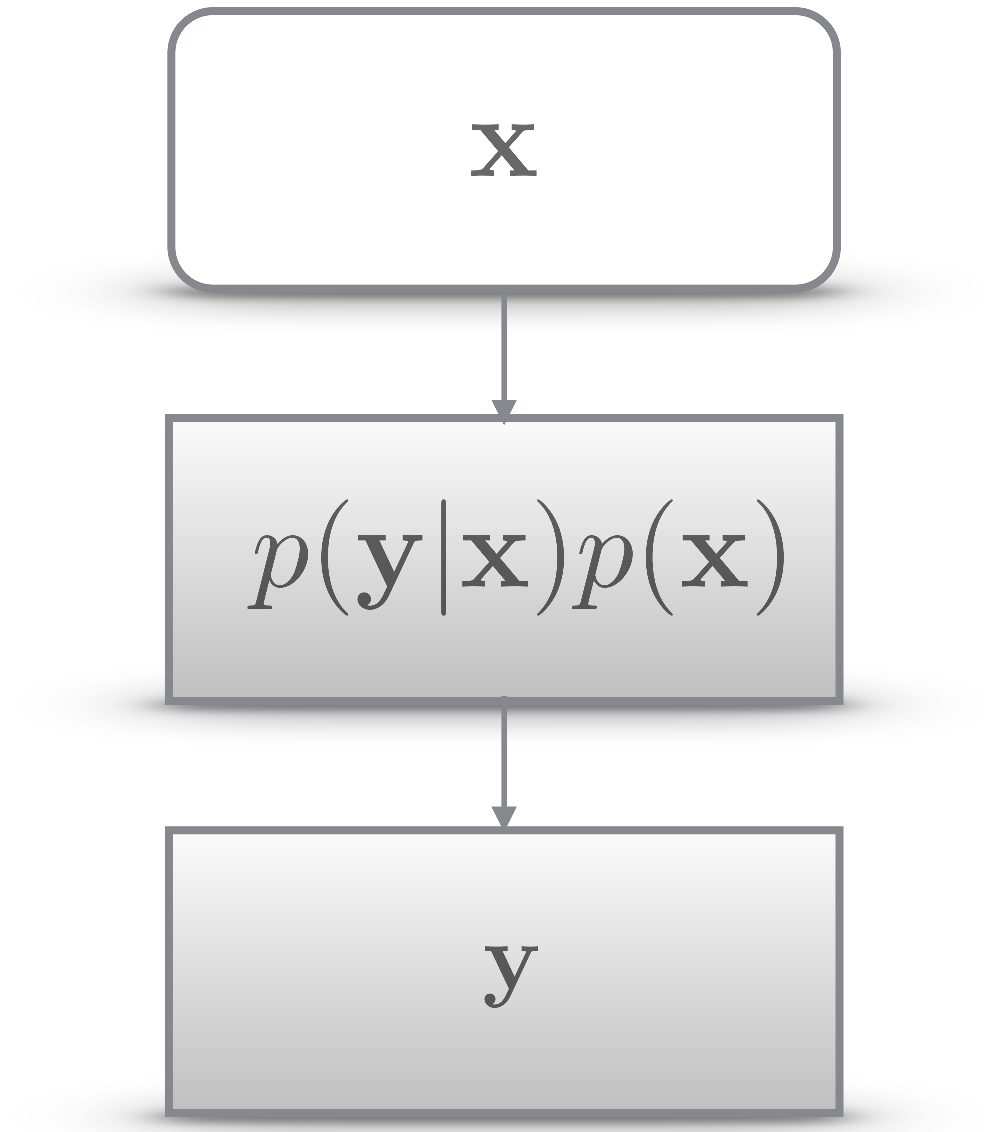


CS

Intuition

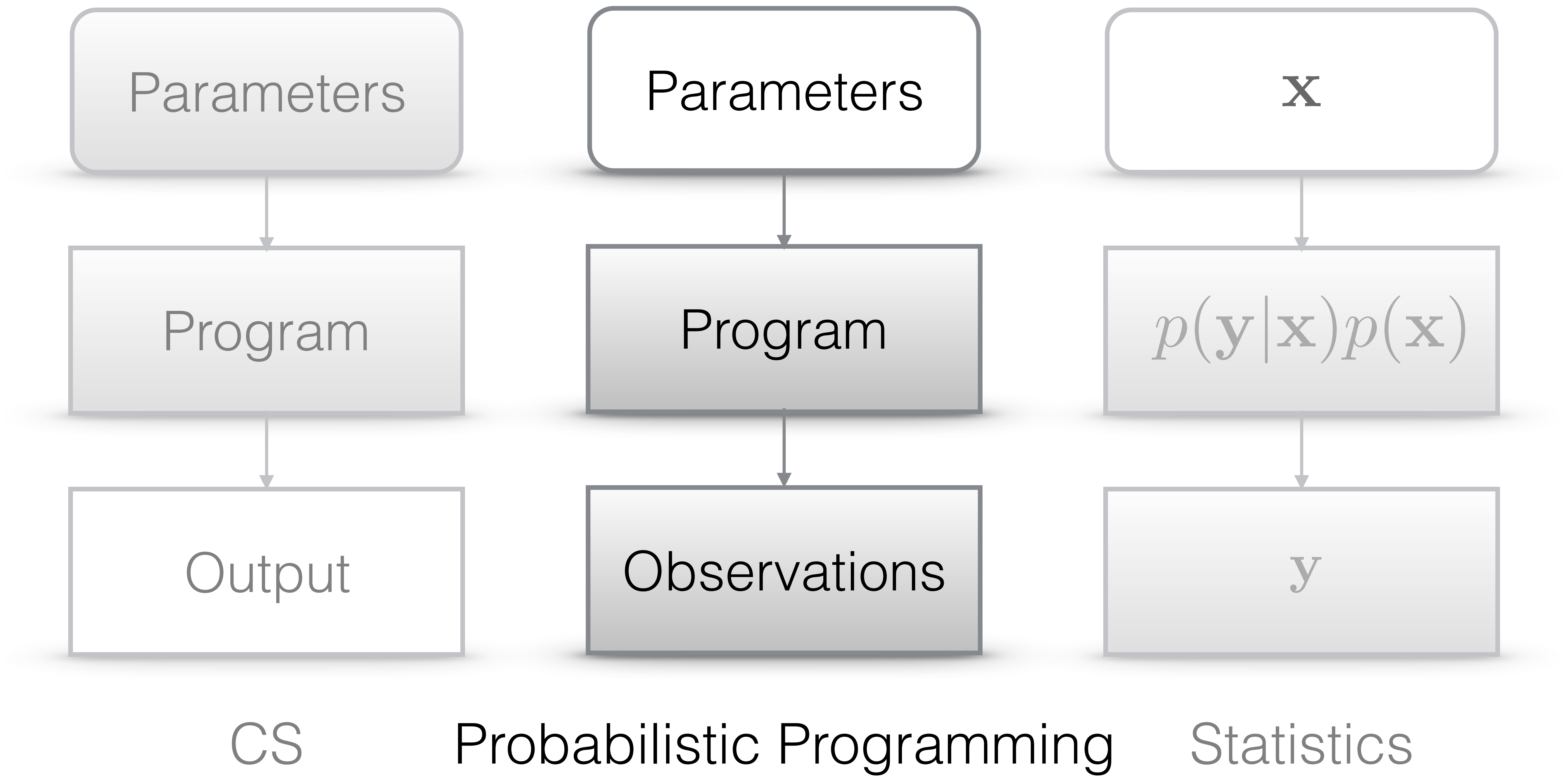


CS

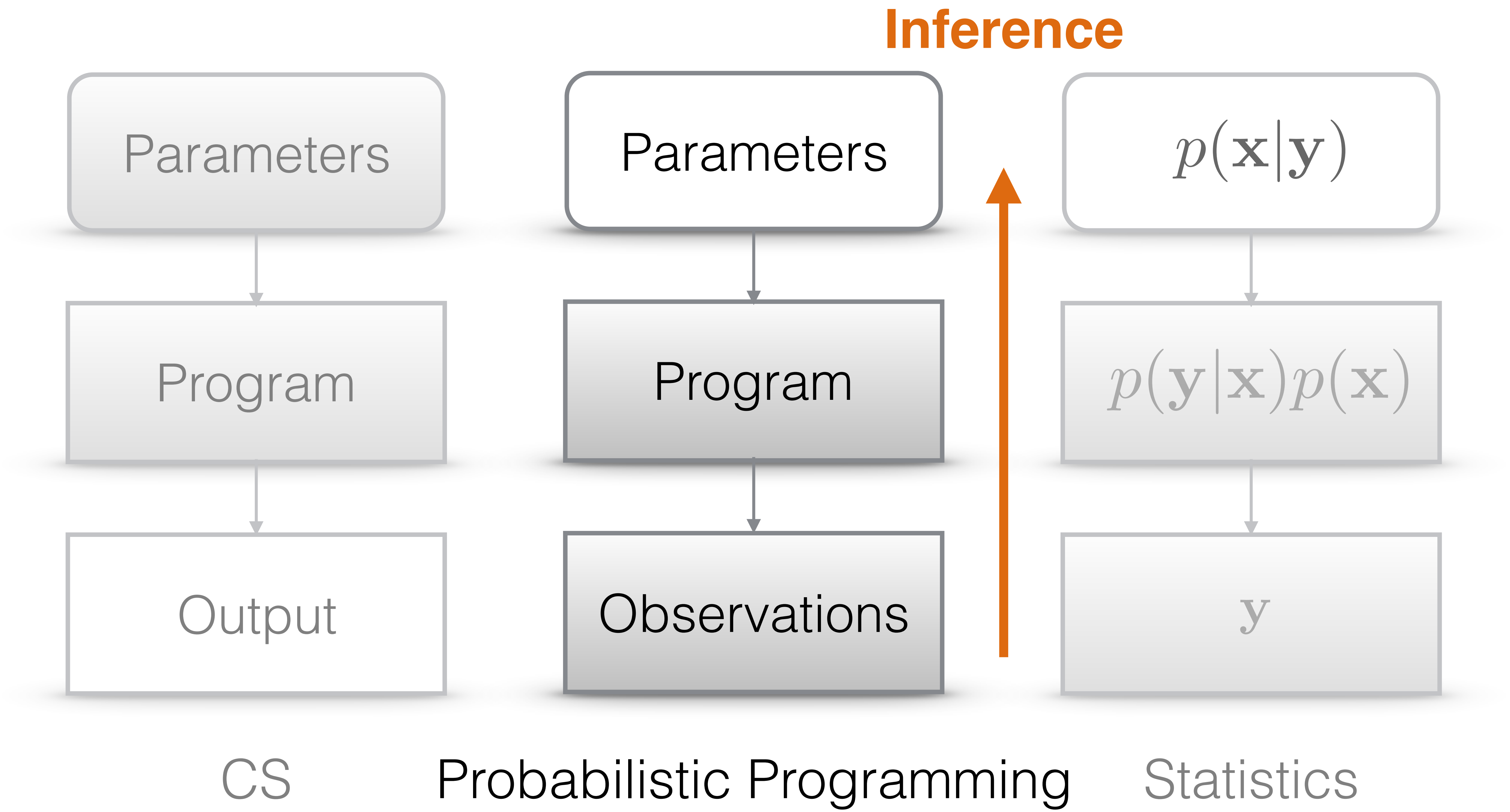


Statistics

Intuition


Parameters
Program
Output
Parameters
Program
Observations
$\mathbf{x}$
$p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$
$\mathbf{y}$
CS
Probabilistic Programming
Statistics

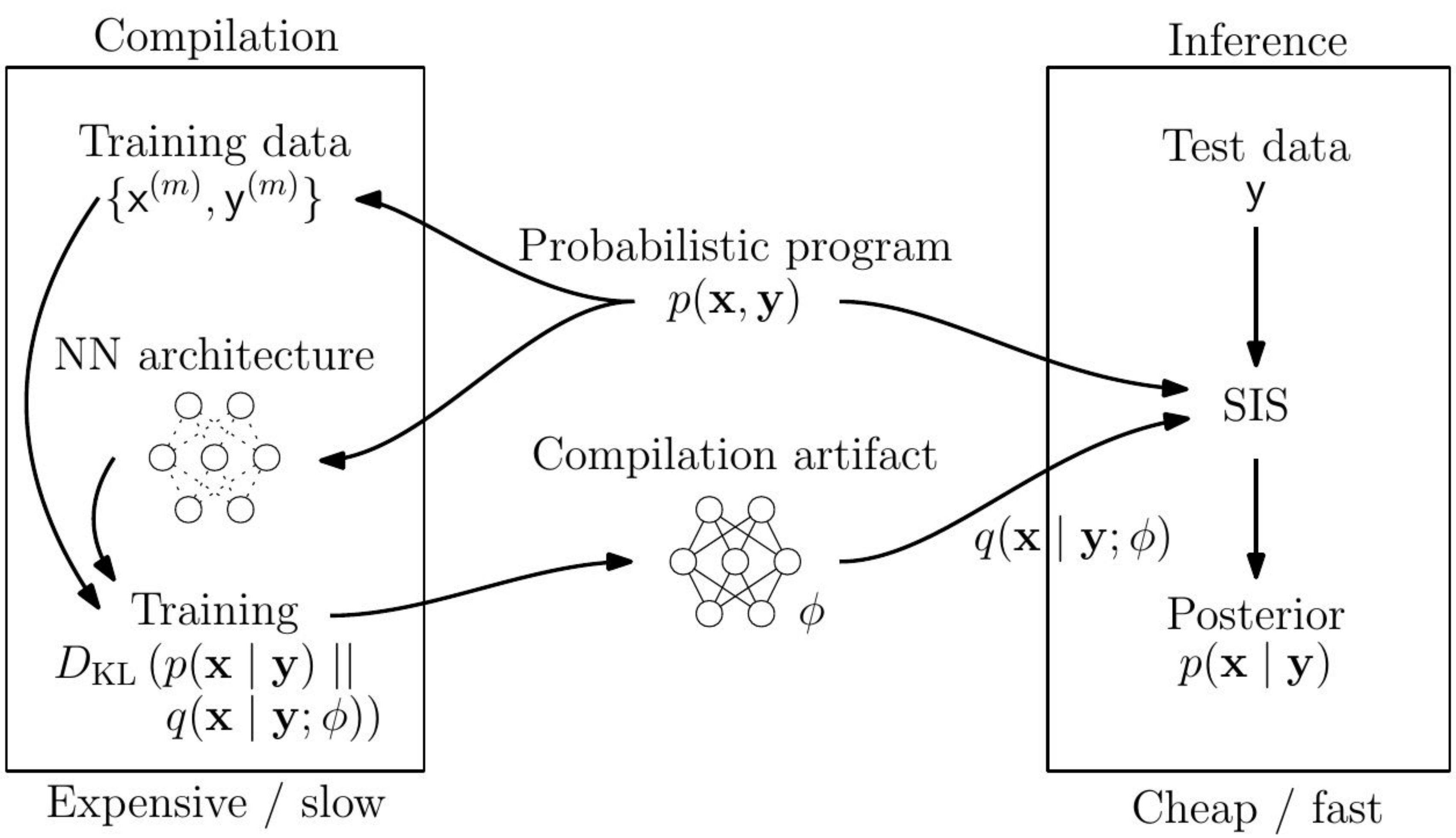
Intuition


Inference
Parameters
Program
Output
Parameters
Program
Observations
$p(\mathbf{x}|\mathbf{y})$
$p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$
$\mathbf{y}$
CS
Probabilistic Programming
Statistics

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

CAPTCHA breaking

Observation

Posterior Samples

Generative Model

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

$\mathbf{x}$	$\mathbf{y}$
text	image

Mansinghka,, Kulkarni, Perov, and Tenenbaum
"Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

CAPTCHA breaking

Observation

Posterior Samples

Generative Model

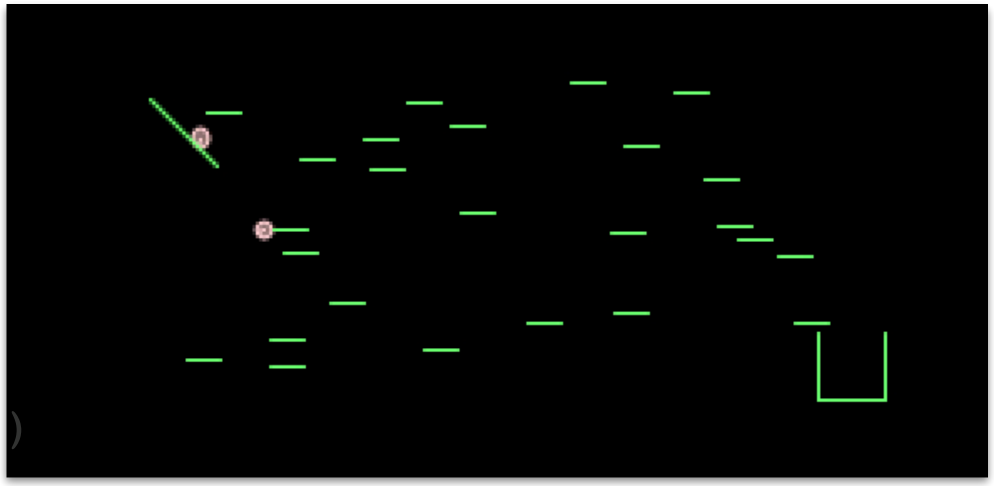
```
(defquery captcha
 [image num-chars tol]
 (let [[w h] (size image)
        ;; sample random characters
        num-chars (sample
                    (poisson num-chars))
        chars (repeatedly
                num-chars sample-char)]
  ;; compare rendering to true image
  (map (fn [y z]
      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

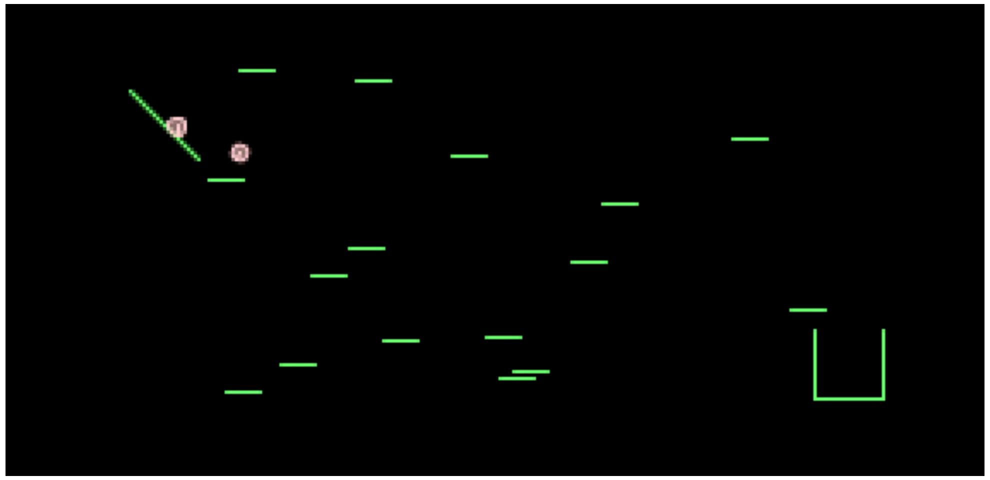
$\mathbf{x}$	$\mathbf{y}$
text	image

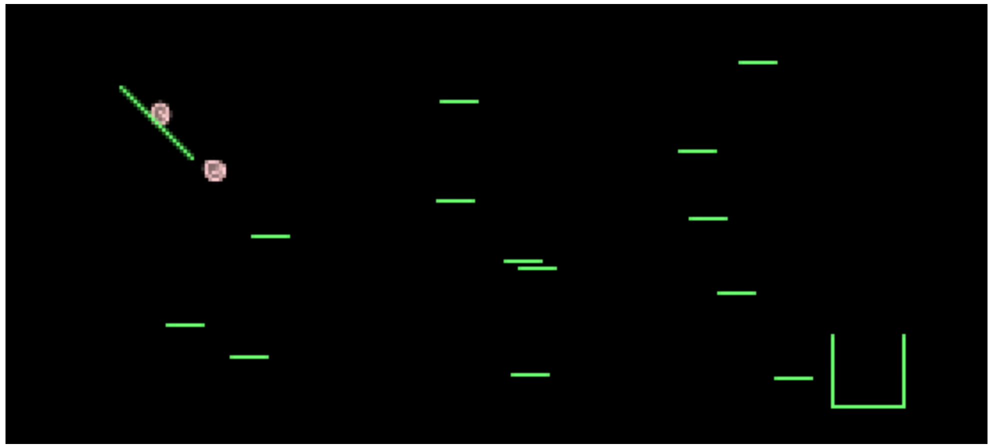
Mansinghka,, Kulkarni, Perov, and Tenenbaum
"Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

ANALOGY: RANDOM BUMPERS ~ RANDOM CALORIMETER SHOWER

```
(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}))
```



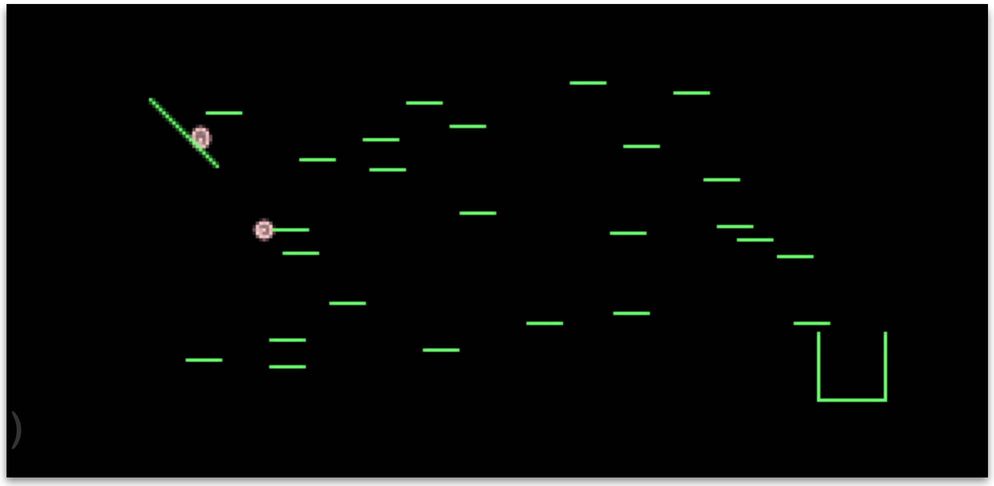


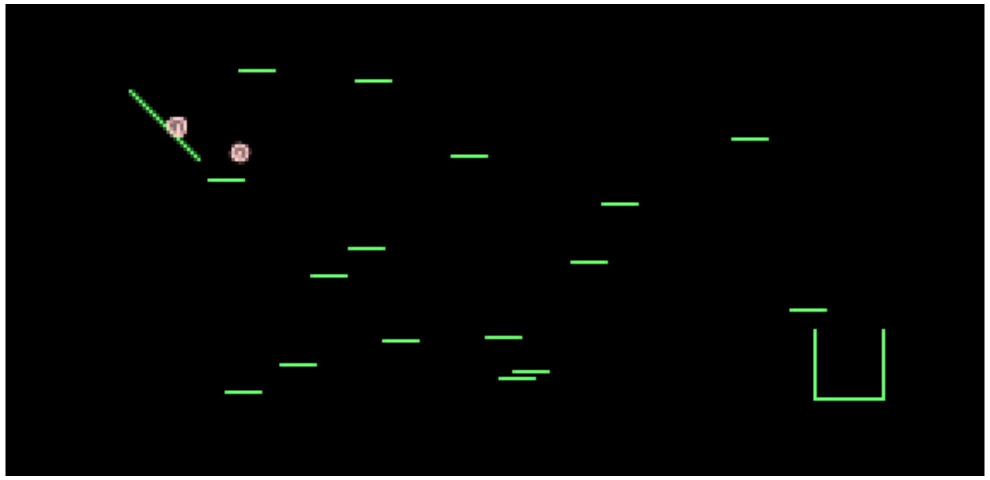


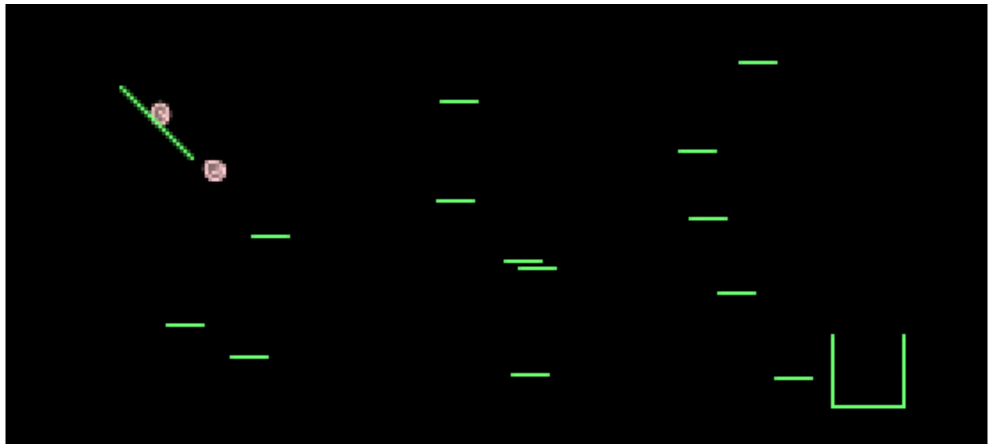
3 examples generated from simulator

[slides, Frank Wood]

```
(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}))
```





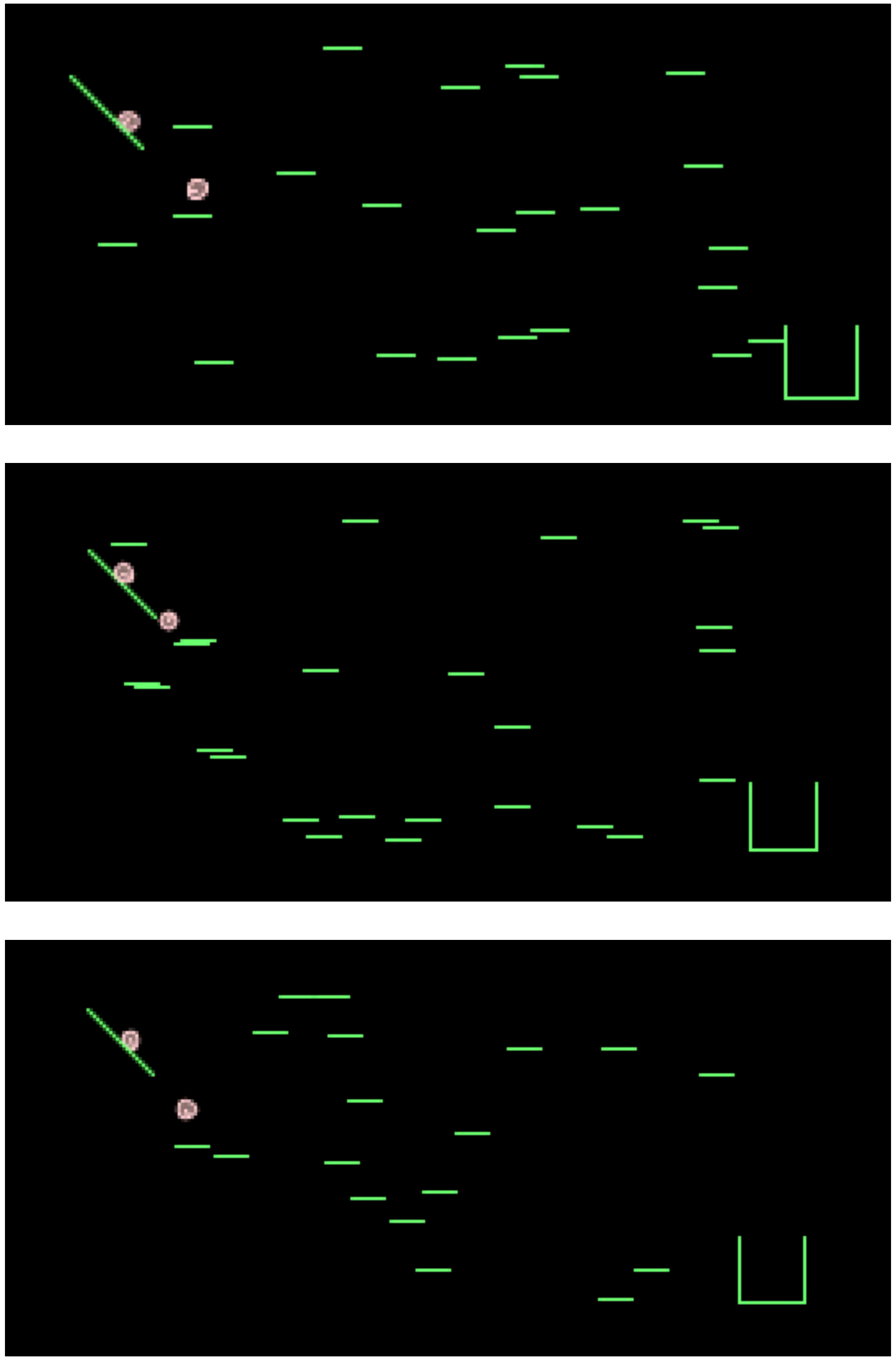


3 examples generated from simulator

[slides, Frank Wood]

UNDERSTANDING THE TAILS OF DISTRIBUTIONS

```
(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)
```

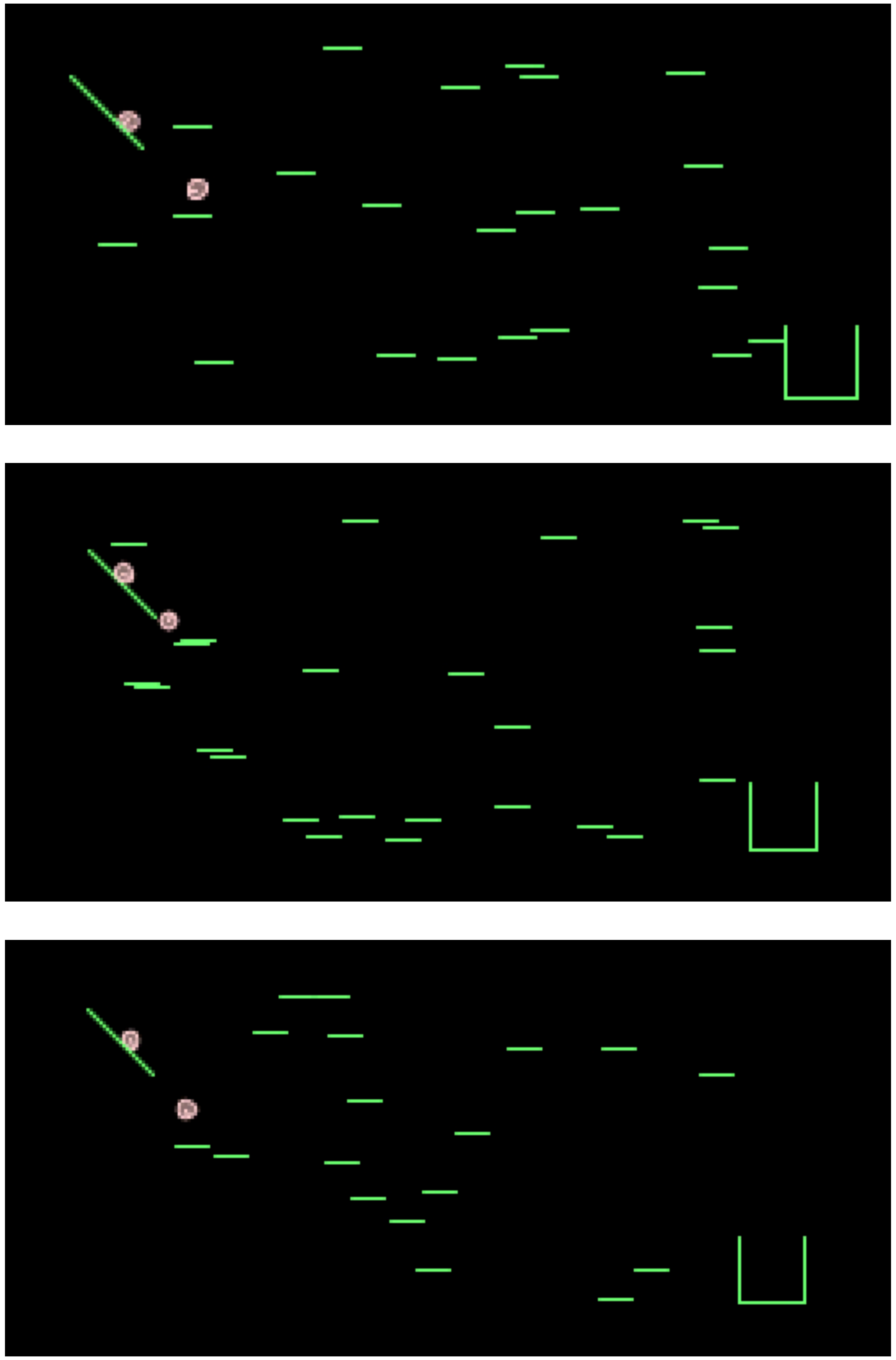


3 examples generated from simulator **conditioned** on ~20% of balls land in box (~ given observed energy deposits)

[slides, Frank Wood]

UNDERSTANDING THE TAILS OF DISTRIBUTIONS

```
(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)
```

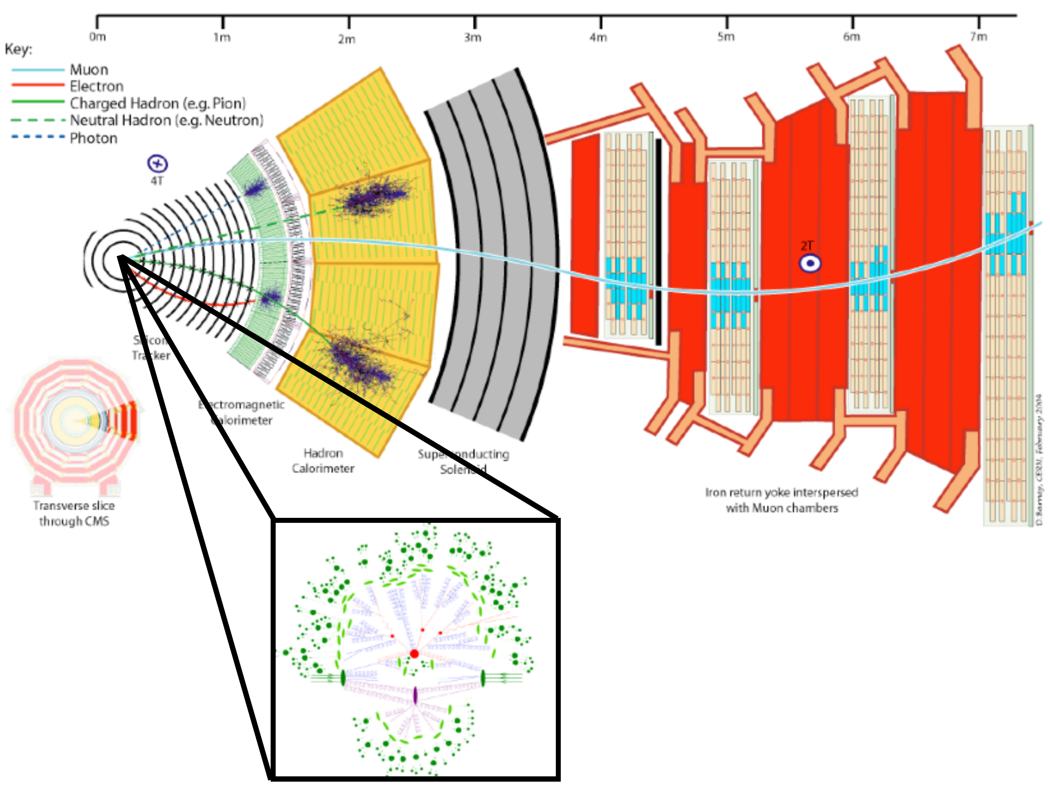


3 examples generated from simulator **conditioned** on ~20% of balls land in box (~ given observed energy deposits)

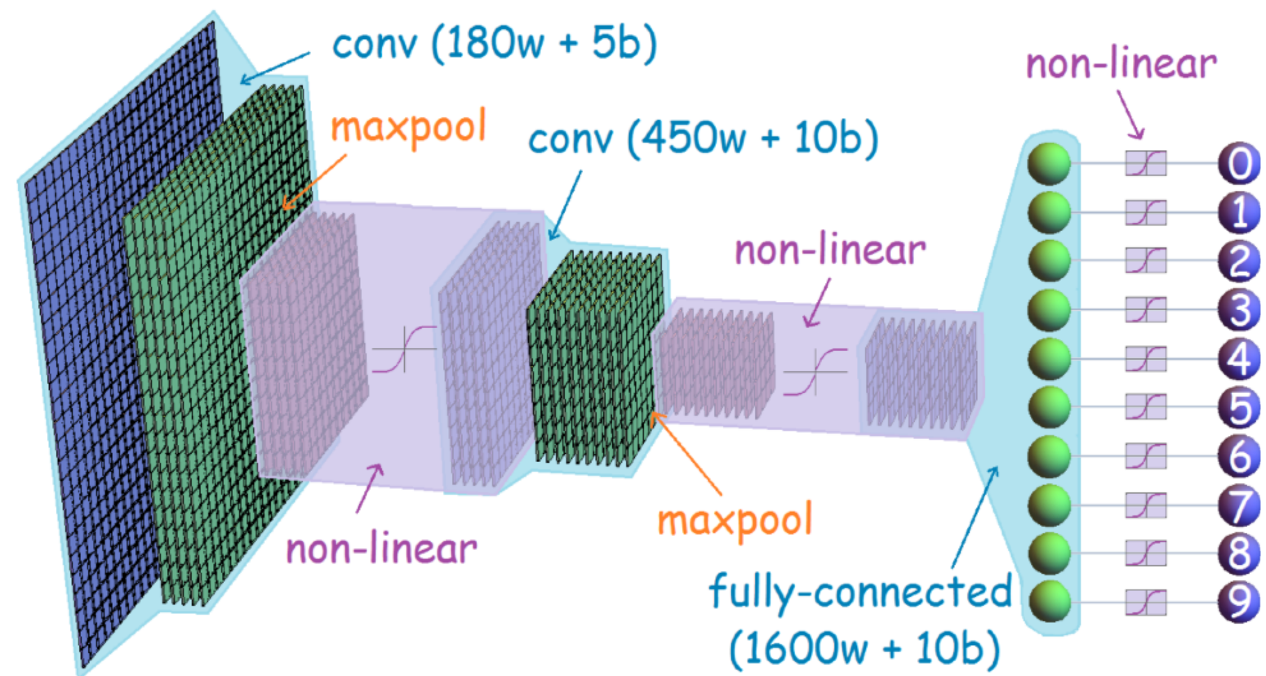
[slides, Frank Wood]

TWO APPROACHES

Use simulator
(much more efficiently)



Learn simulator
(with deep learning)



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

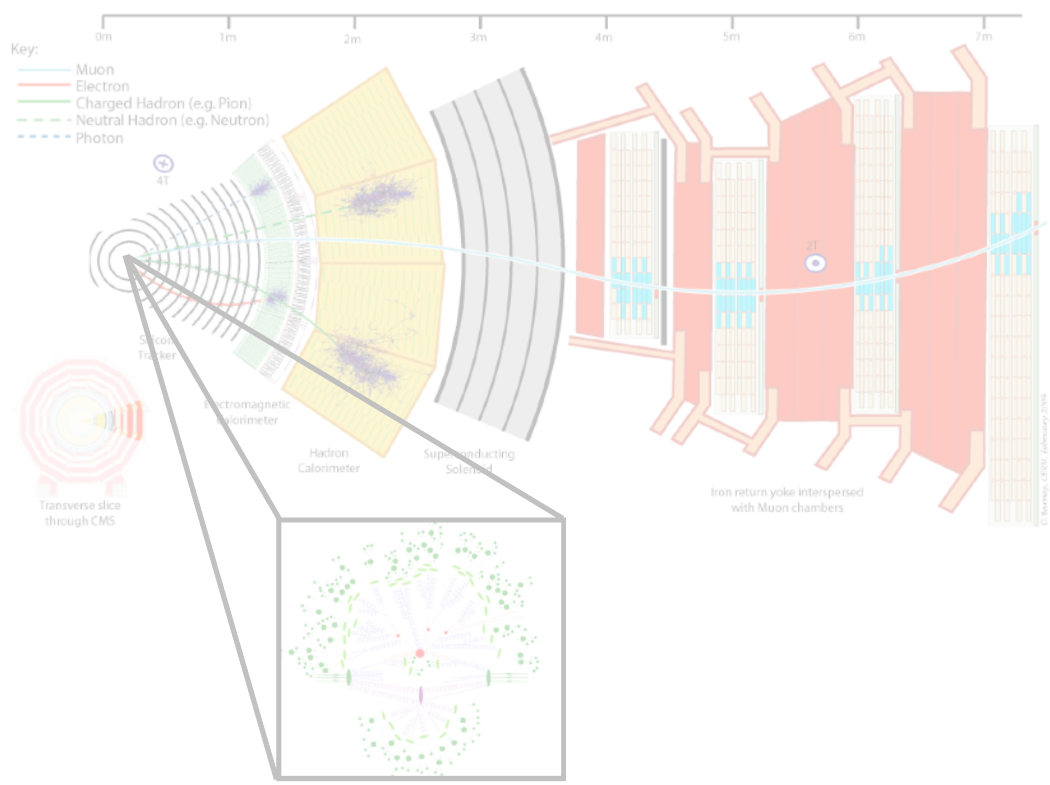
- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

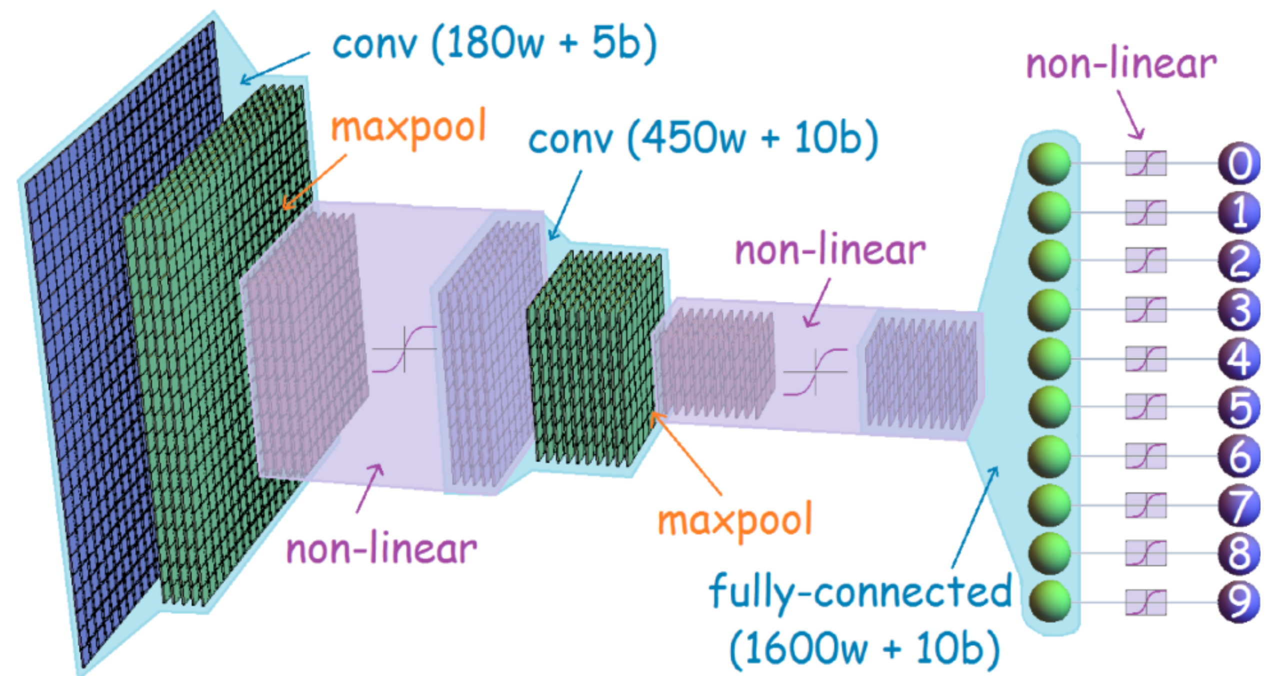
[image credit: A.P. Goucher]

TWO APPROACHES

Use simulator
(much more efficiently)

Learn simulator
(with deep learning)



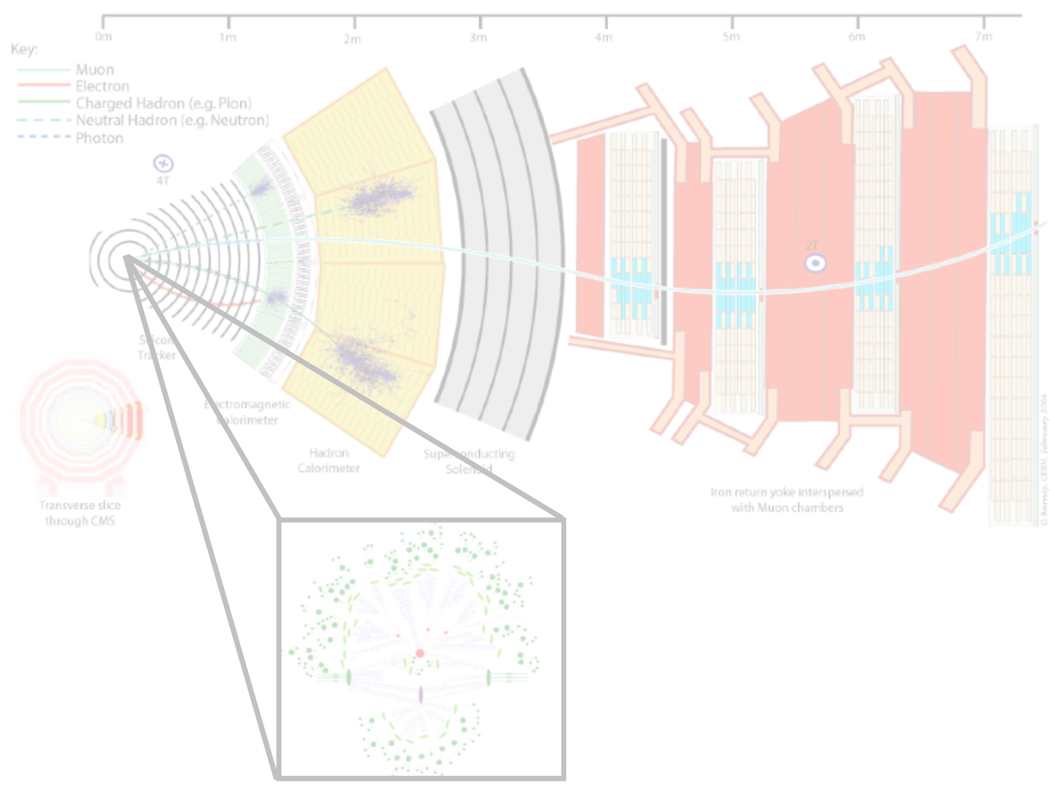
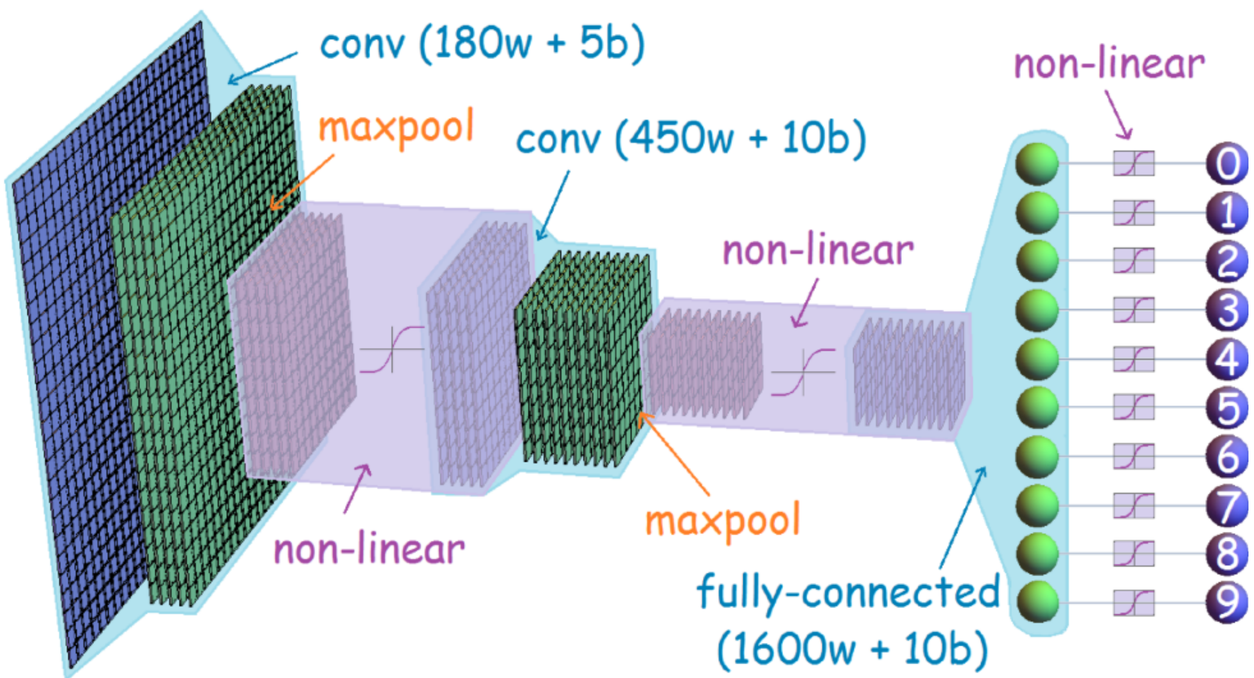


- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

- 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)	Learn simulator (with deep learning)
	
• Approximate Bayesian Computation (ABC)	• Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
• Probabilistic Programming	• Likelihood ratio from classifiers (CARL)
• Adversarial Variational Optimization (AVO)	• Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

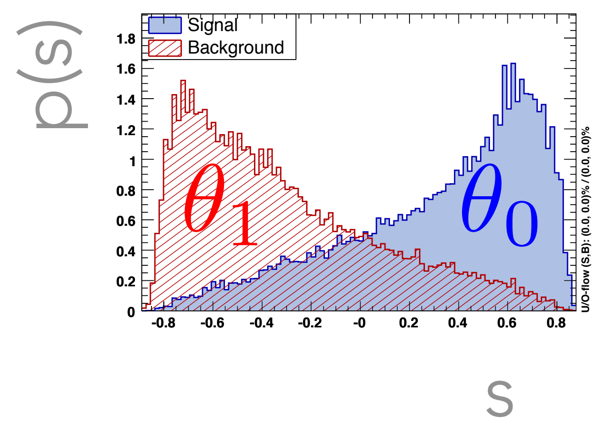
CARL

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: http://arxiv.org/abs/1506.02169

Binary classifier on balanced y=0 and y=1 labels learns

$$s(x) = \frac{p(x|y=1)}{p(x|y=0) + p(x|y=1)}$$

Which is one-to-one with the likelihood ratio

$$\frac{p(x|y=0)}{p(x|y=1)} = 1 - \frac{1}{s(x)}$$

Can do the same thing for any two points θ_0 & θ_1 in parameter space. 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: http://arxiv.org/abs/1506.02169

CARL SOFTWARE

Gilles Louppe

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

diana-hep.org

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

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

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

`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

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

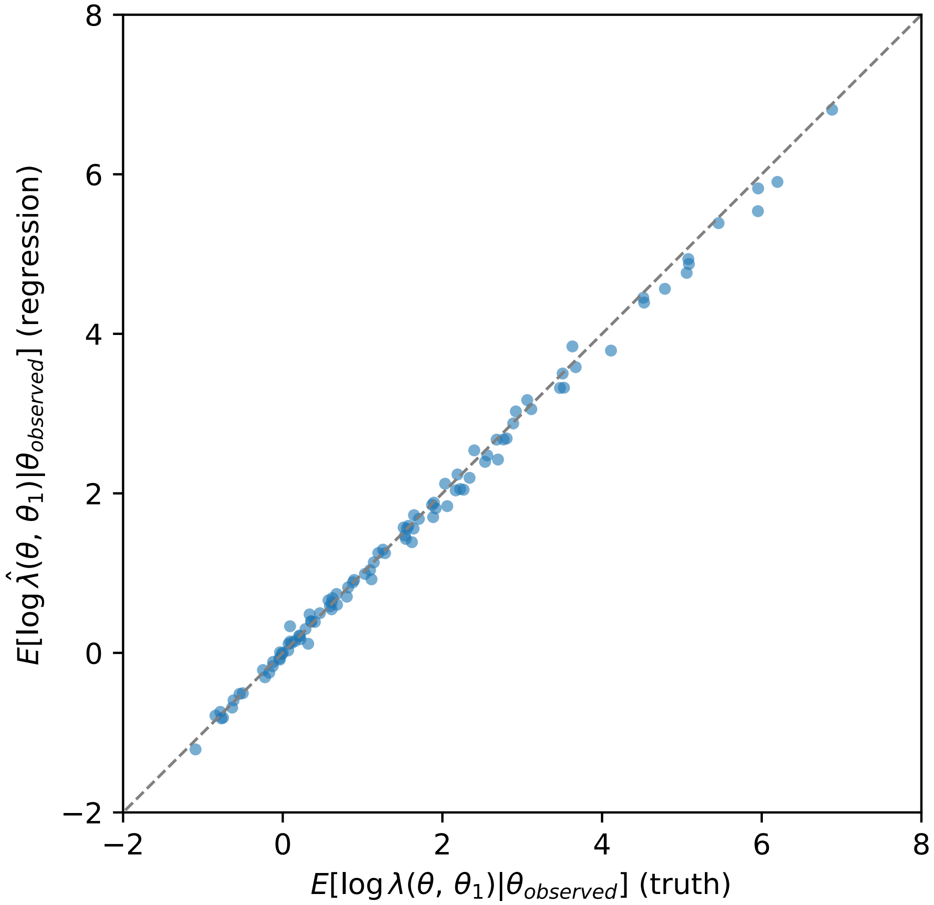
The following dependencies are required:

- Numpy >= 1.11

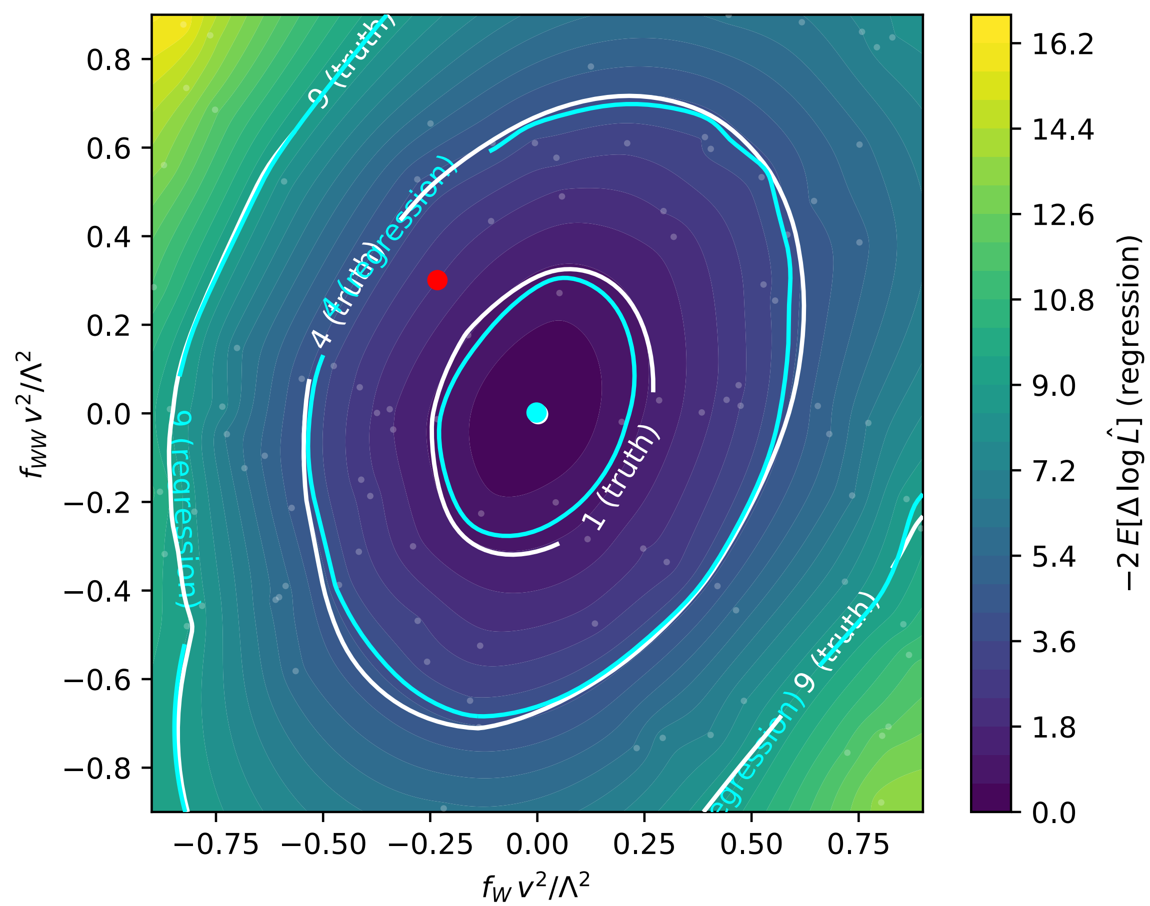
Display a menu

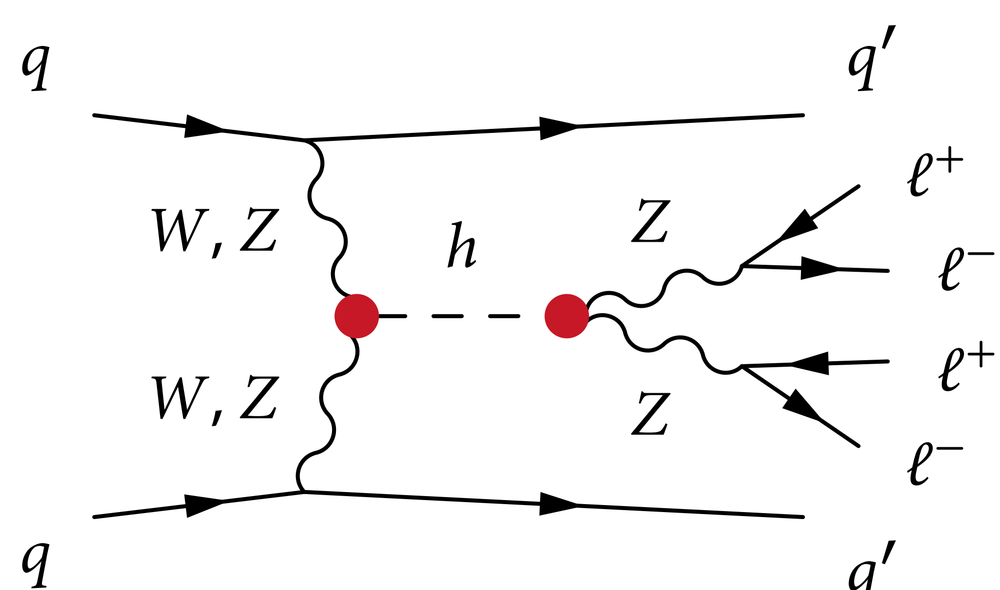
LEARNING A 16 DIM LIKELIHOOD

Estimated likelihood



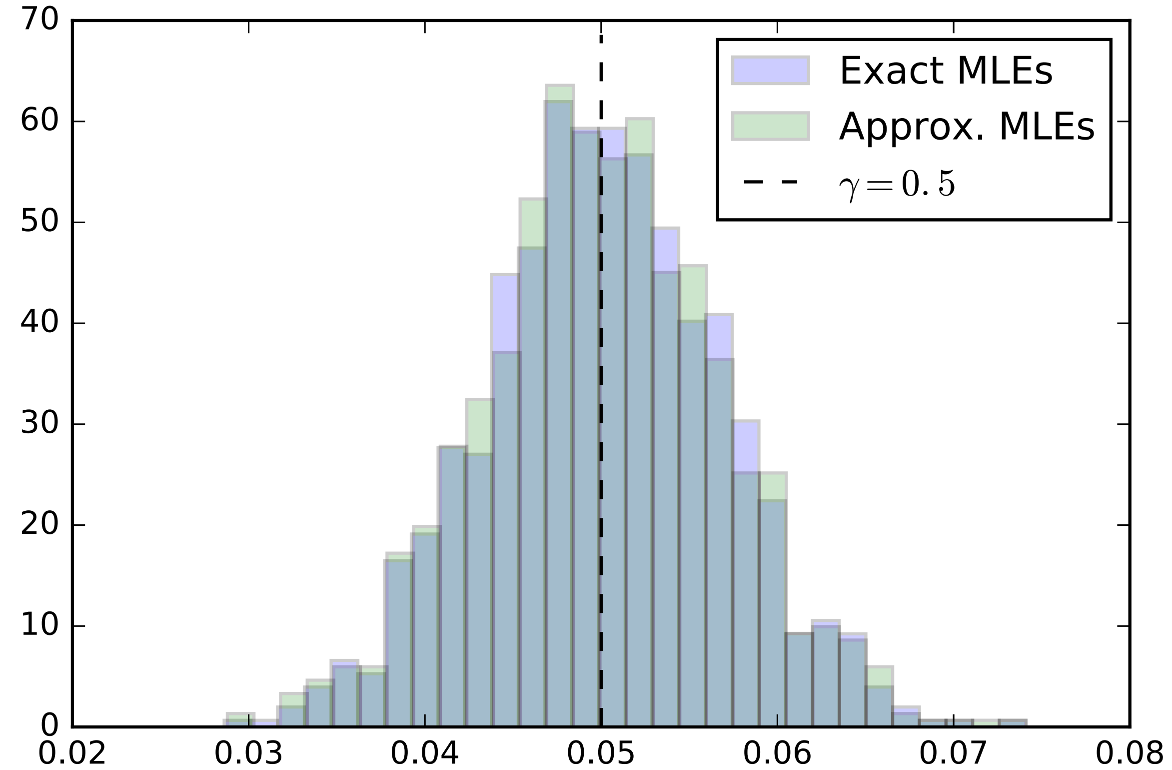
True likelihood



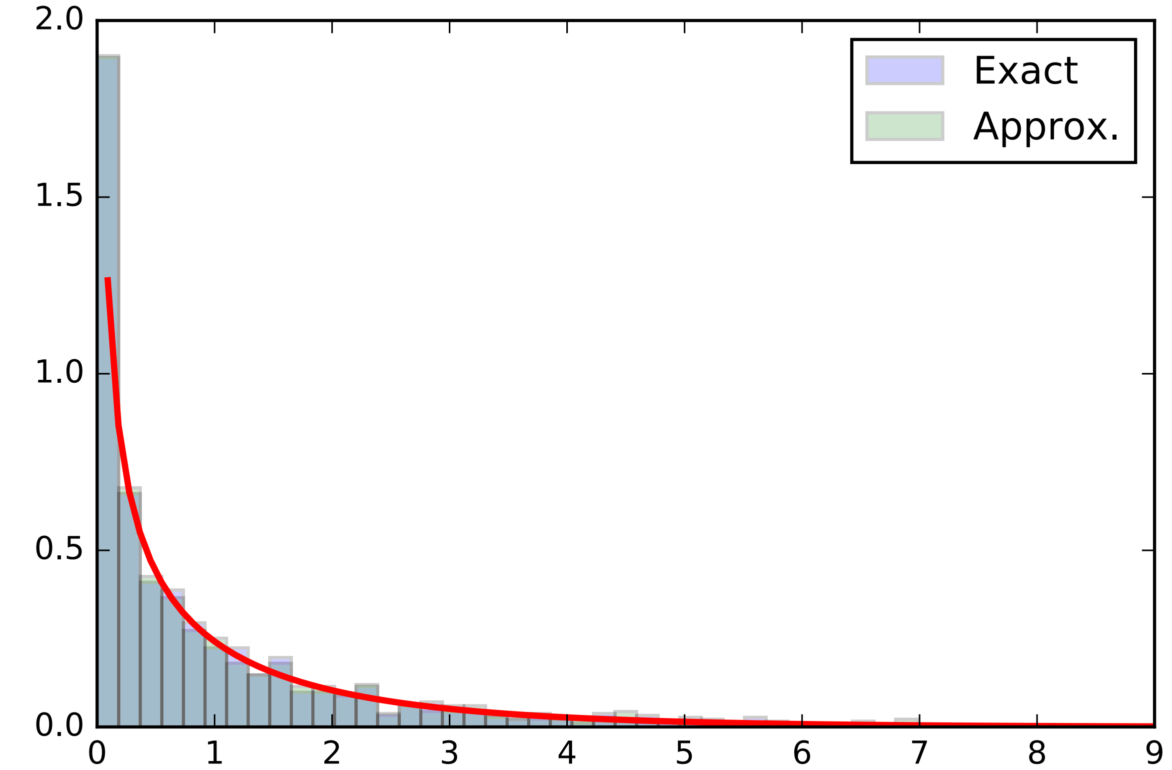


Once we've learned the function s(x; θ) to approximate the likelihood, we can apply it to any data x.

- 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.

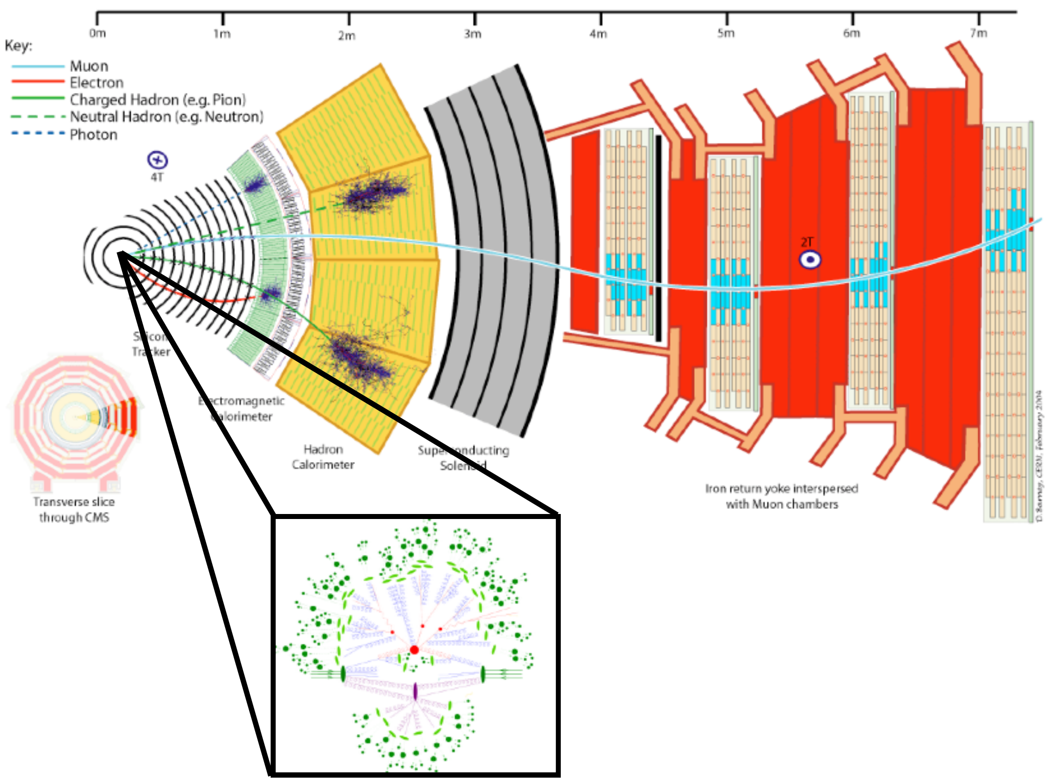


(b) $p(-2 \log \Lambda(\gamma = 0.05) \mid \gamma = 0.05)$

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

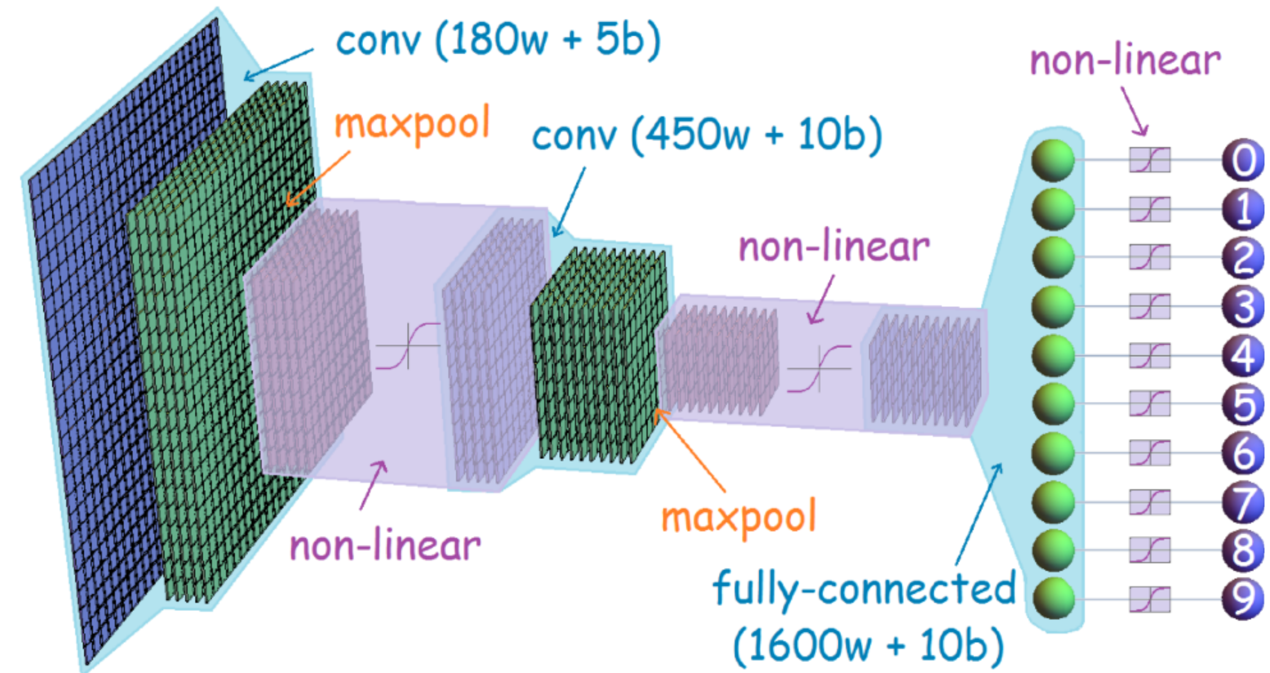
TWO APPROACHES

Use simulator
(much more efficiently)



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

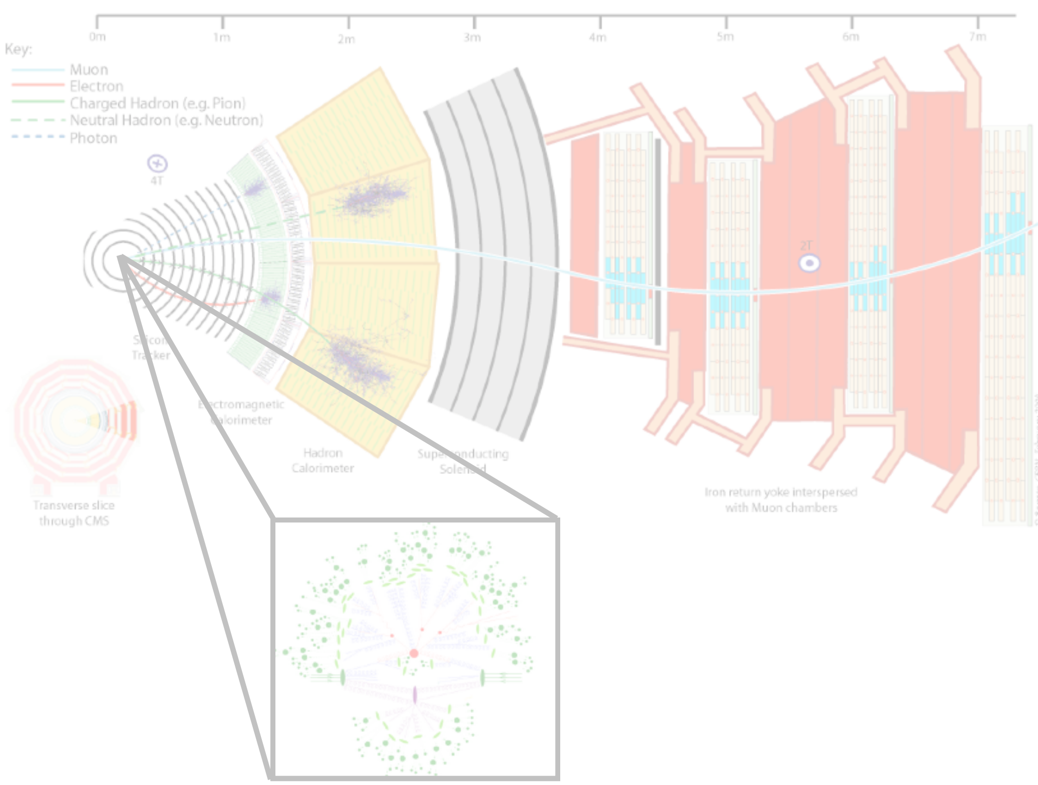
Learn simulator
(with deep learning)



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

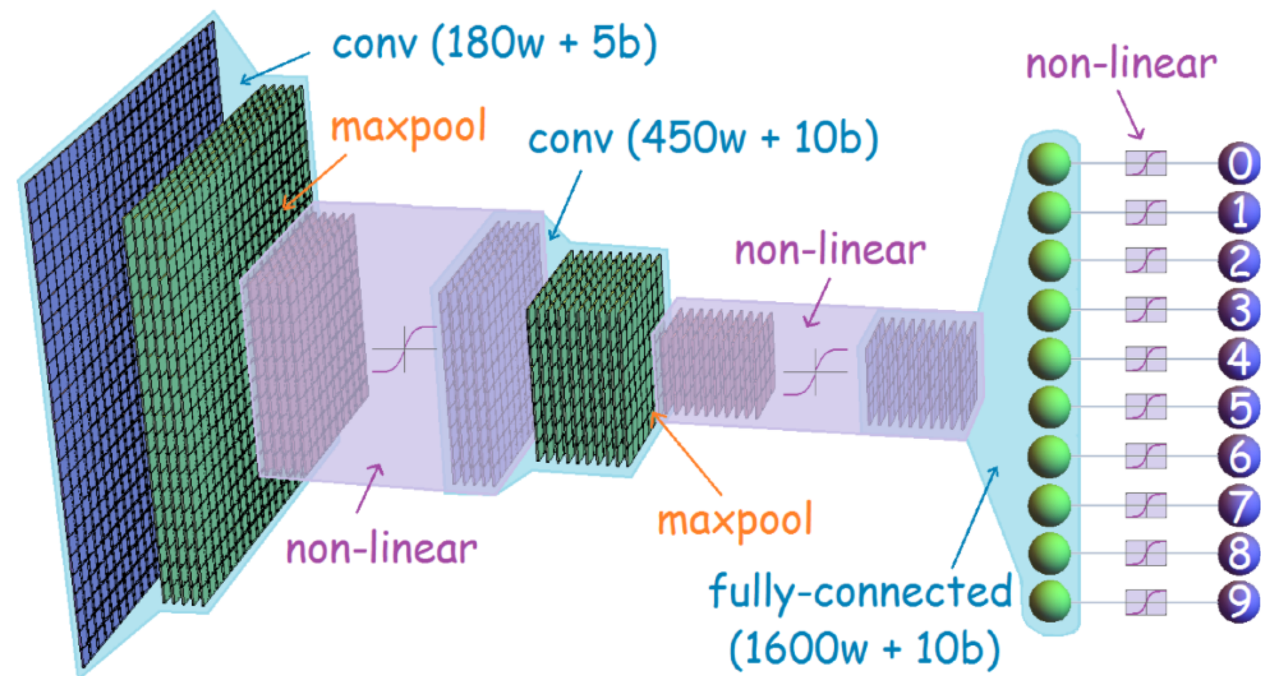
TWO APPROACHES

Use simulator
(much more efficiently)



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

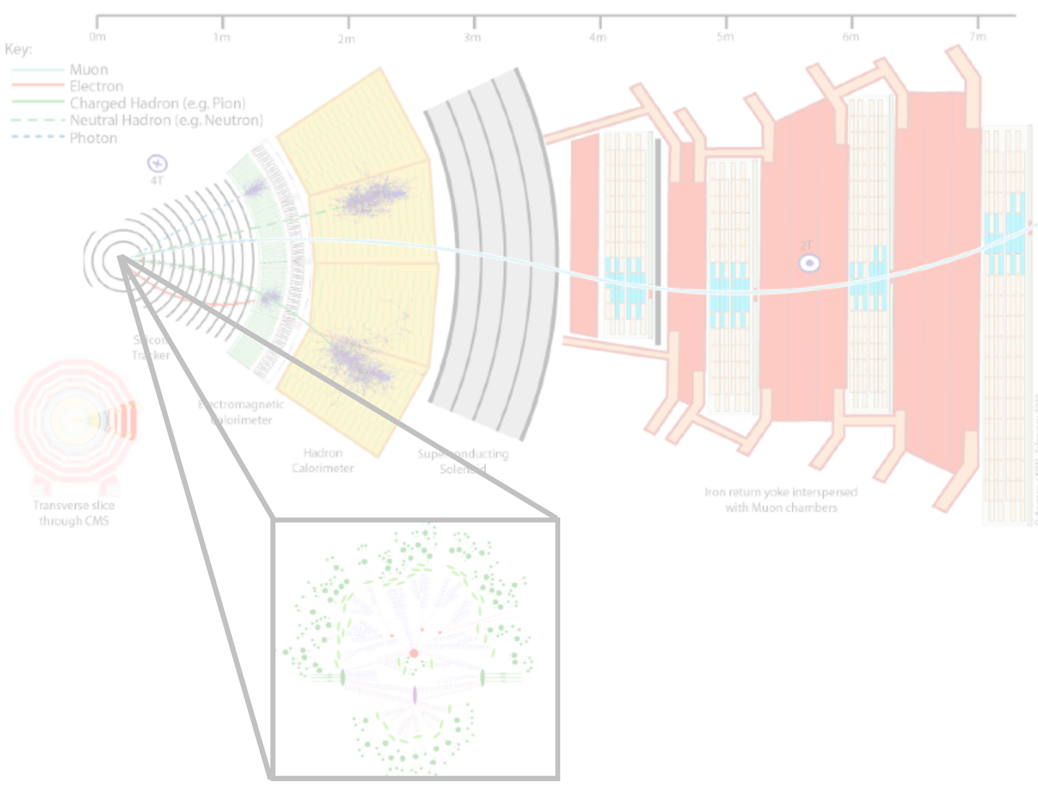
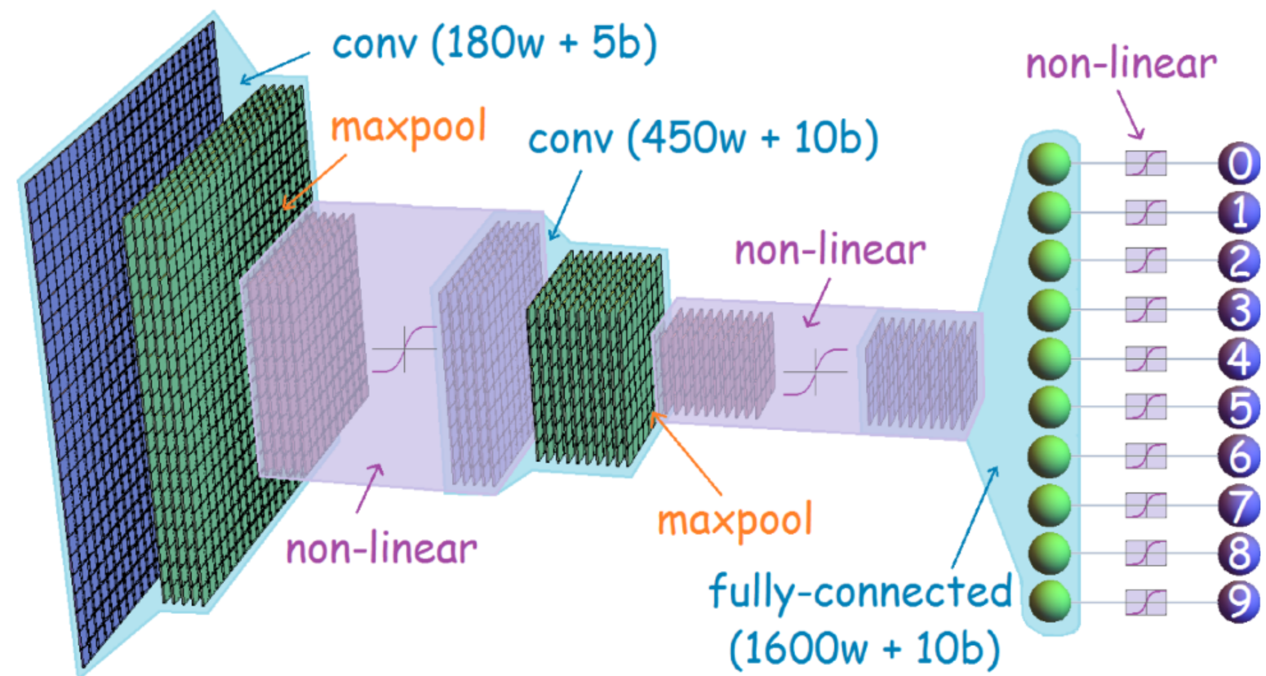
Learn simulator
(with deep learning)



- 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)	Learn simulator (with deep learning)
	

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

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

[image credit: A.P. Goucher]

AN ALTERNATIVE APPROACH

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

By Kyle Cranmer, Gilles Loupe

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

Actions

Authors

Kyle Cranmer, Gilles Loupe

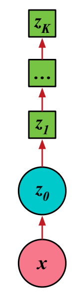
Metadata

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

Published: 8 Dec, 2016

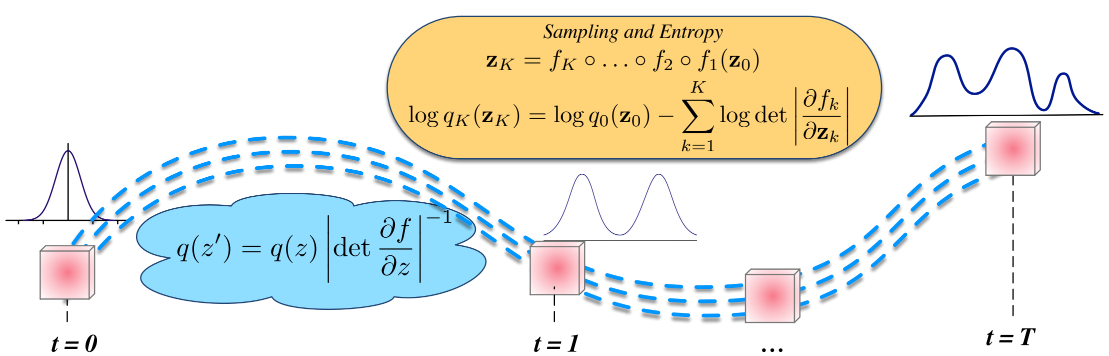
Normalizing flows and autoregressive models

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.



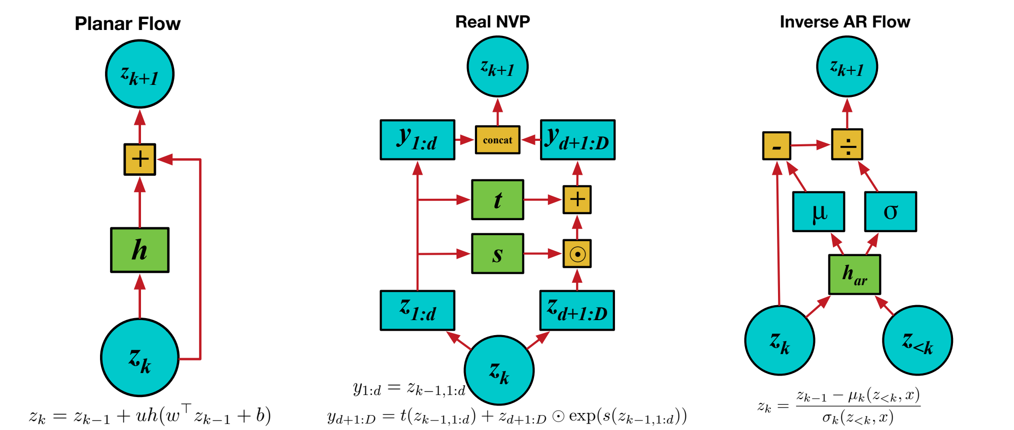
Distribution flows through a sequence of invertible transforms

[Rezende and Mohamed, 2015]

Choice of Transformation Function

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

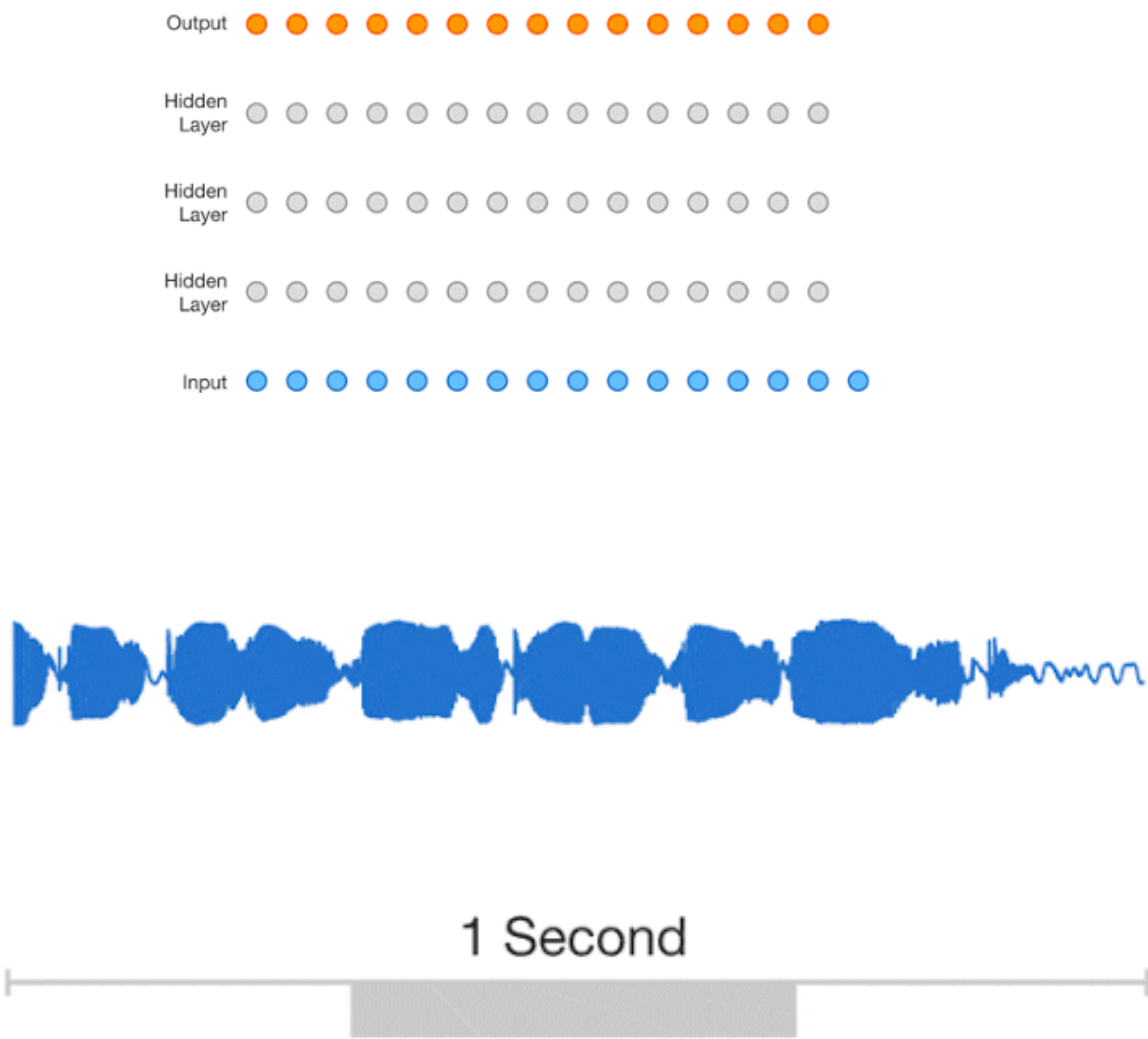
- Begin with a fully-factorised Gaussian and improve by change of variables.
- Triangular Jacobians allow for computational efficiency.



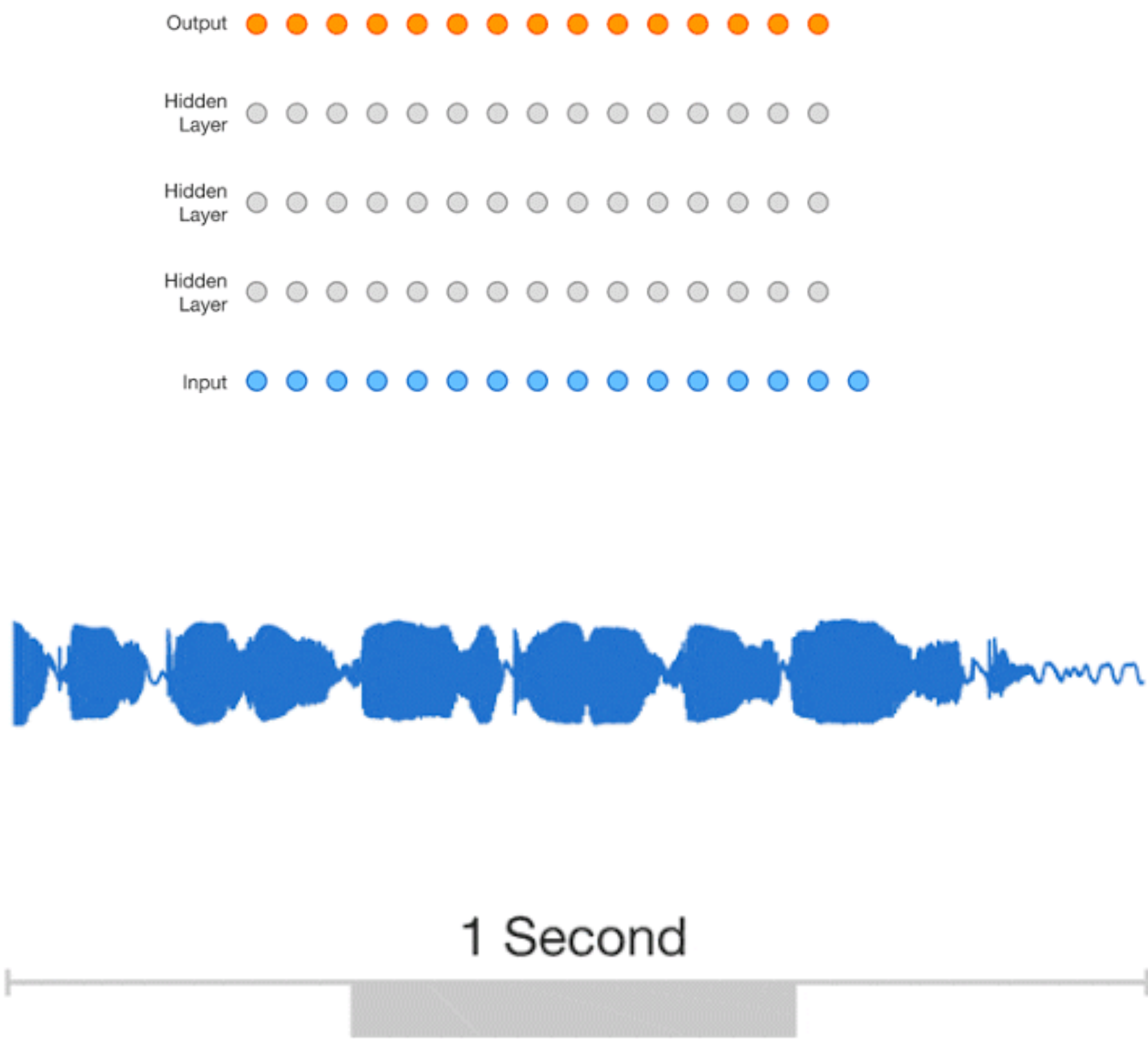
[Rezende and Mohamed, 2016; Dinh et al., 2016; Kingma et al., 2016]

Linear time computation of the determinant and its gradient.

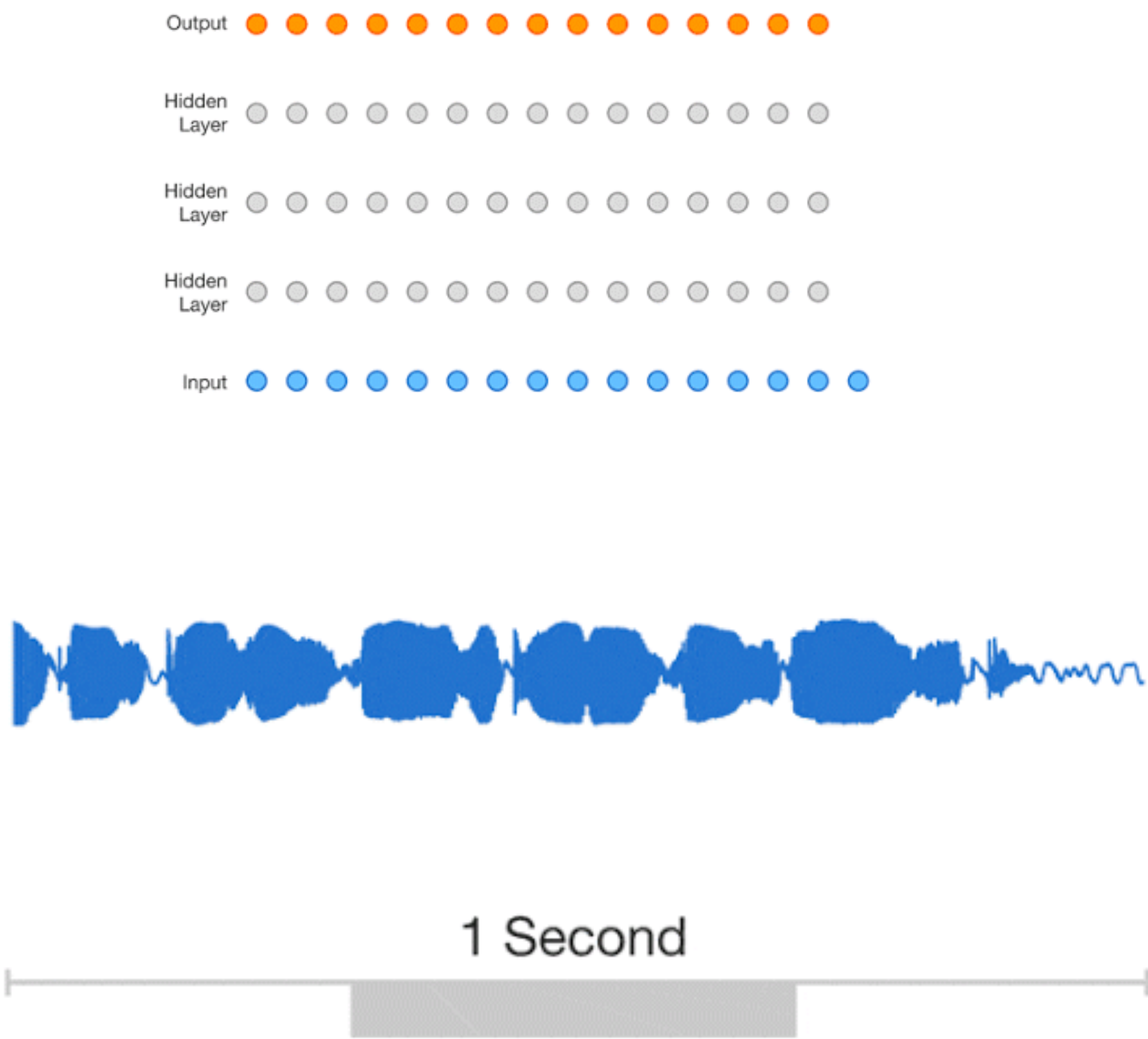
WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



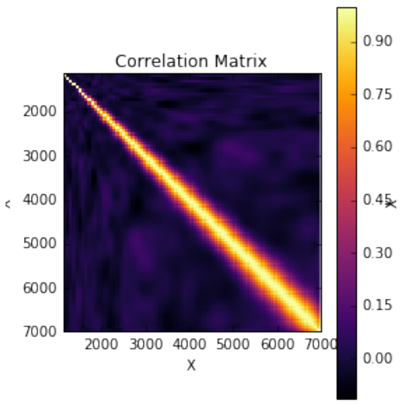
WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



We can inject our knowledge of physics into the variational family

Physics-aware Gaussian Processes

arXiv:1709.05681

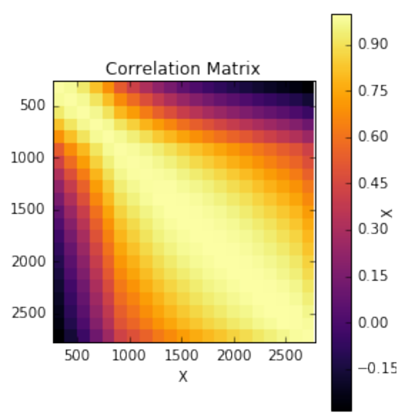


Final Kernel =

Poisson fluctuations

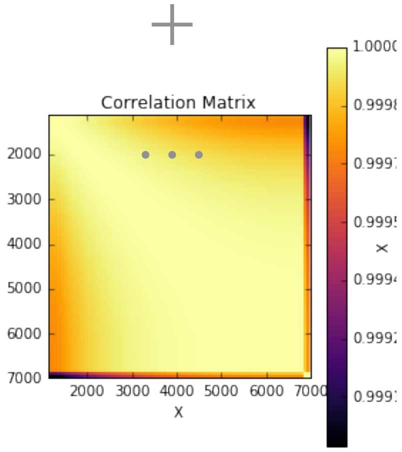
=

+ Mass Resolution



+ Parton Density Functions

+

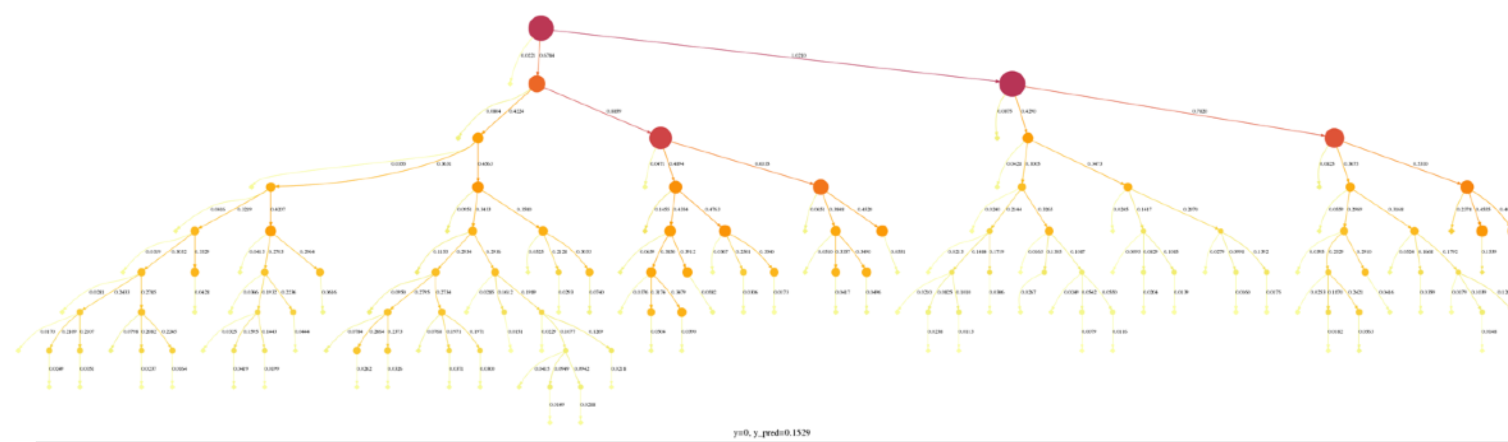


+ Jet Energy Scale

+

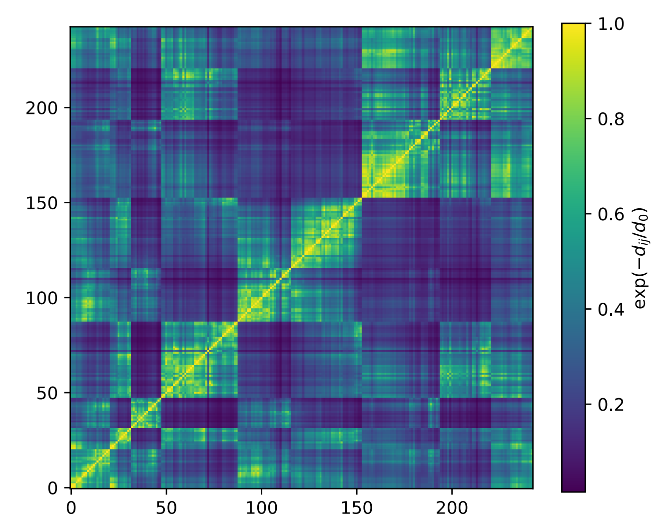
QCD-Aware recursive neural networks

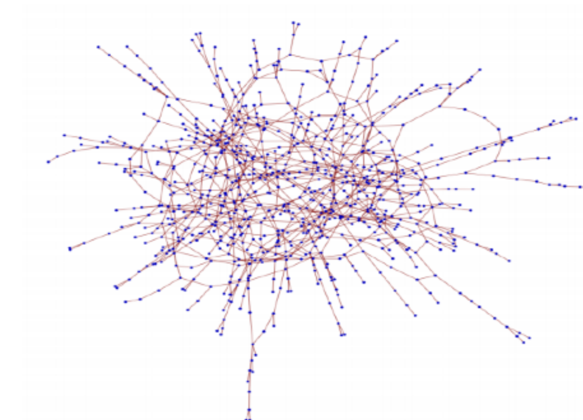
arXiv:1702.00748



QCD-Aware graph convolutional neural networks

NIPS2017 workshop





$$d^{\alpha}_{ii'} = \min(p^{2\alpha}_{ti}, p^{2\alpha}_{ti'}) \frac{\Delta R^2_{ii'}}{R^2}$$

Gaussian Processes

NEW PAPER https://arxiv.org/abs/1709.05681

Information | References (44) | Citations (0) | Files | Plots

Modeling Smooth Backgrounds and Generic Localized Signals with Gaussian Processes

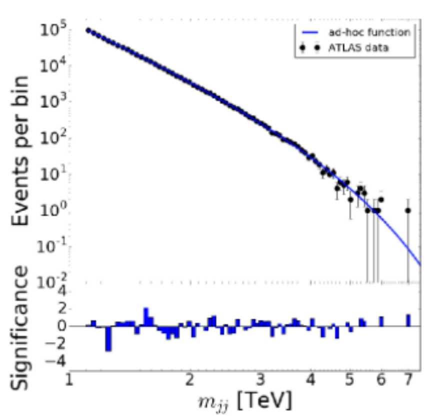
Meghan Frate, Kyle Cranmer, Saarik Kalia, Alexander Vandenberg-Rodes, Daniel Whiteson

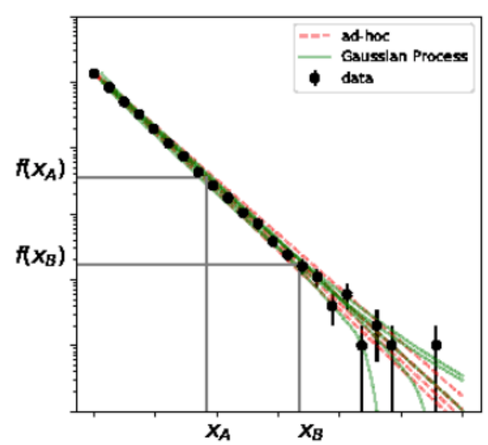
Sep 17, 2017 - 14 pages

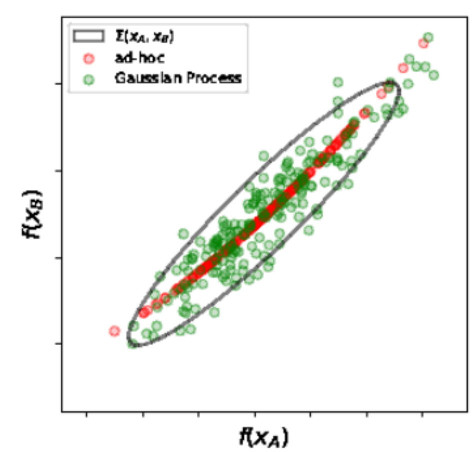
e-Print: arXiv:1709.05681 [physics.data-an] | PDF

Abstract (arXiv)
We describe a procedure for constructing a model of a smooth data spectrum using Gaussian processes rather than the historical parametric description. This approach considers a fuller space of possible functions, is robust at increasing luminosity, and allows us to incorporate our understanding of the underlying physics. We demonstrate the application of this approach to modeling the background to searches for dijet resonances at the Large Hadron Collider and describe how the approach can be used in the search for generic localized signals.

Note: *Temporary entry*
Note: 14 pages, 16 figures
Keyword(s): INSPIRE: background | CERN LHC Coll | dijet | resonance | data analysis method | Gauss model | statistics | statistical analysis







Show more plots

Record added 2017-09-19, last modified 2017-10-07 dianahep

Collaborative Analyses

Establish infrastructure for a higher-level of collaborative analysis, building on the successful patterns used for the Higgs boson discovery and enabling a deeper communication between the theoretical community and the experimental community

Reproducible Analyses

Streamline efforts associated to reproducibility, analysis preservation, and data preservation by making these native concepts in the tools

Interoperability

Improve the interoperability of HEP tools with the larger scientific software ecosystem, incorporating best practices and algorithms from other disciplines into HEP

Faster Processing

Increase the CPU and IO performance needed to reduce the iteration time so crucial to exploring new ideas

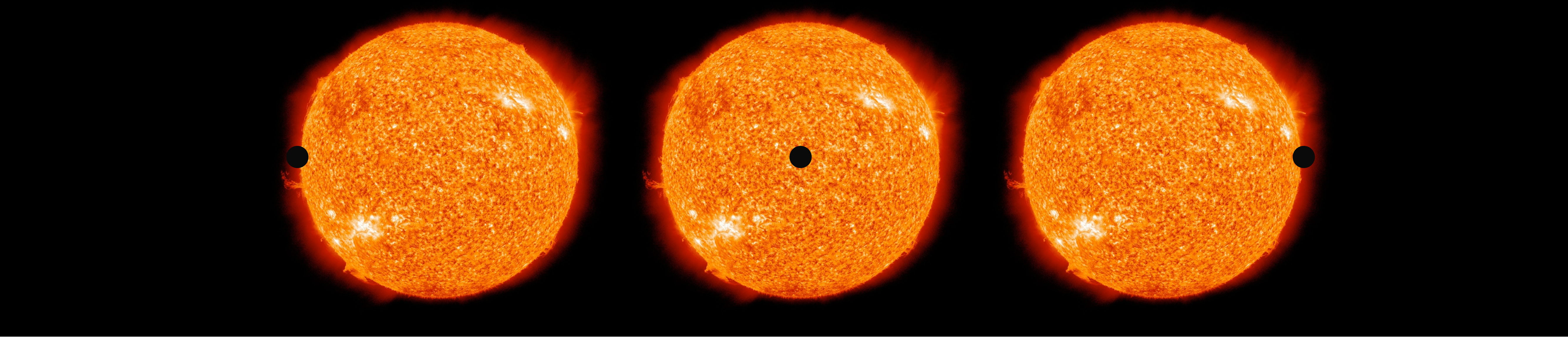
Better Software

Develop software to effectively exploit emerging many- and multi-core hardware.
Promote the concept of software as a research product.

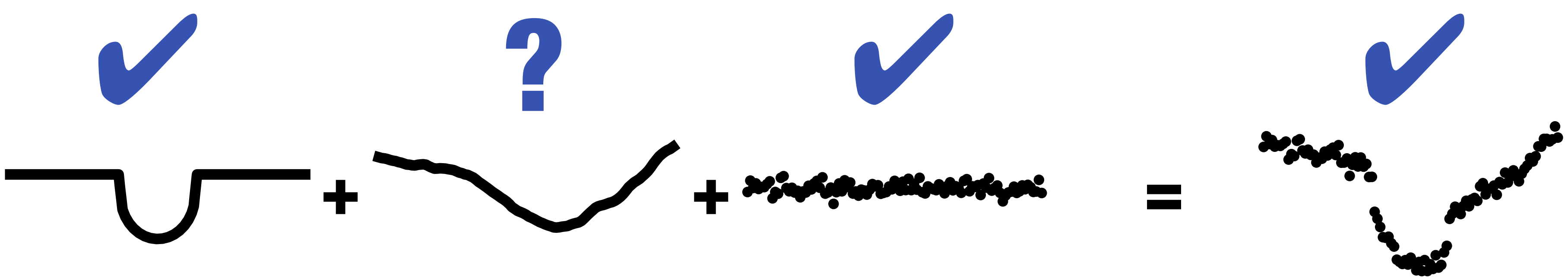
Training

Provide training for students in all of our core research topics.

http://diana-hep.org



The anatomy of a transit observation

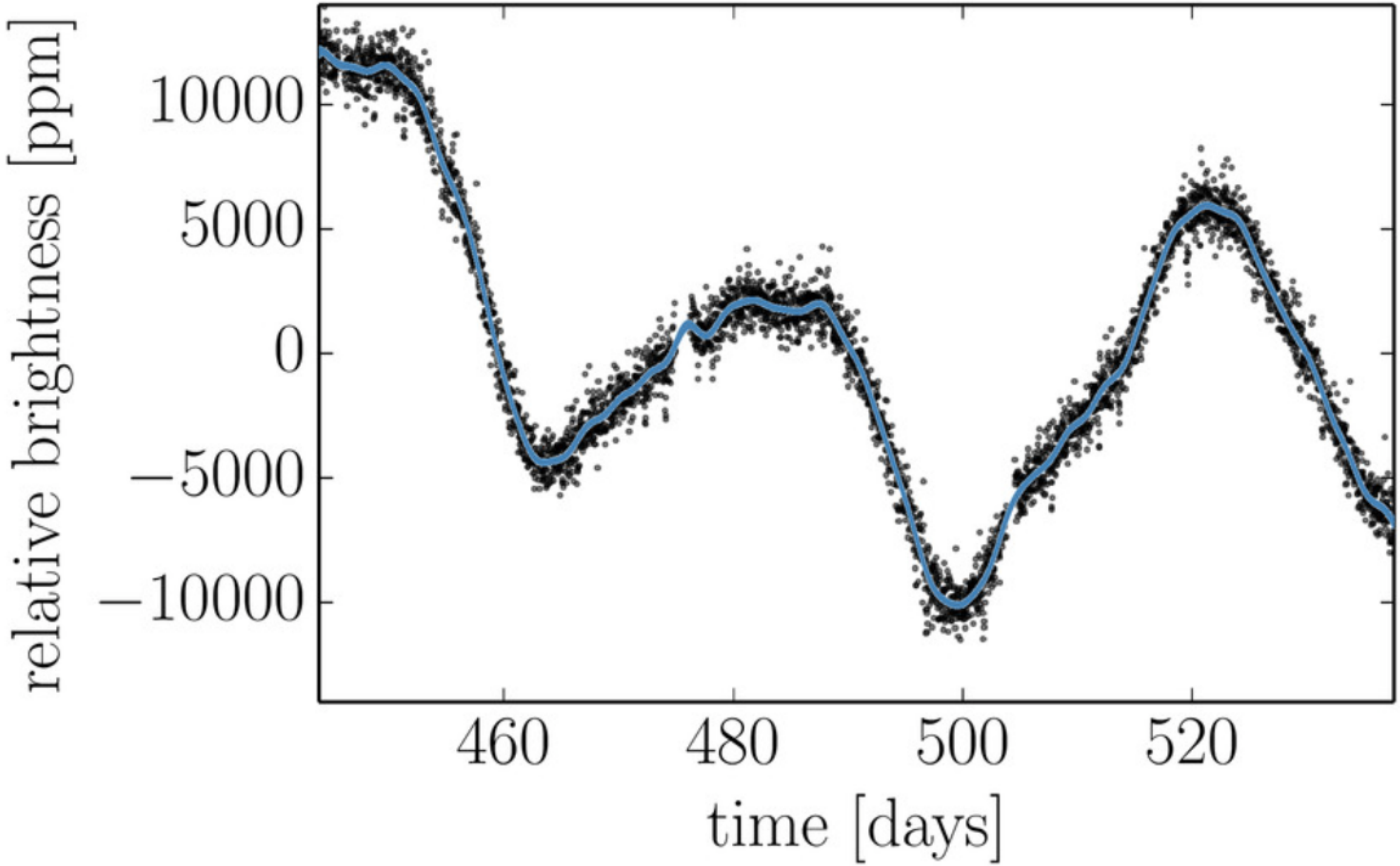


signal + variability + noise = data

variability → astrophysics and spacecraft

[Slides by Dan Foreman-Mackey]

AN EXOPLANET EXAMPLE



https://speakerdeck.com/dfm/pydata-time-series-analysis-gps-and-exoplanets

the data are drawn from one

HUGE*

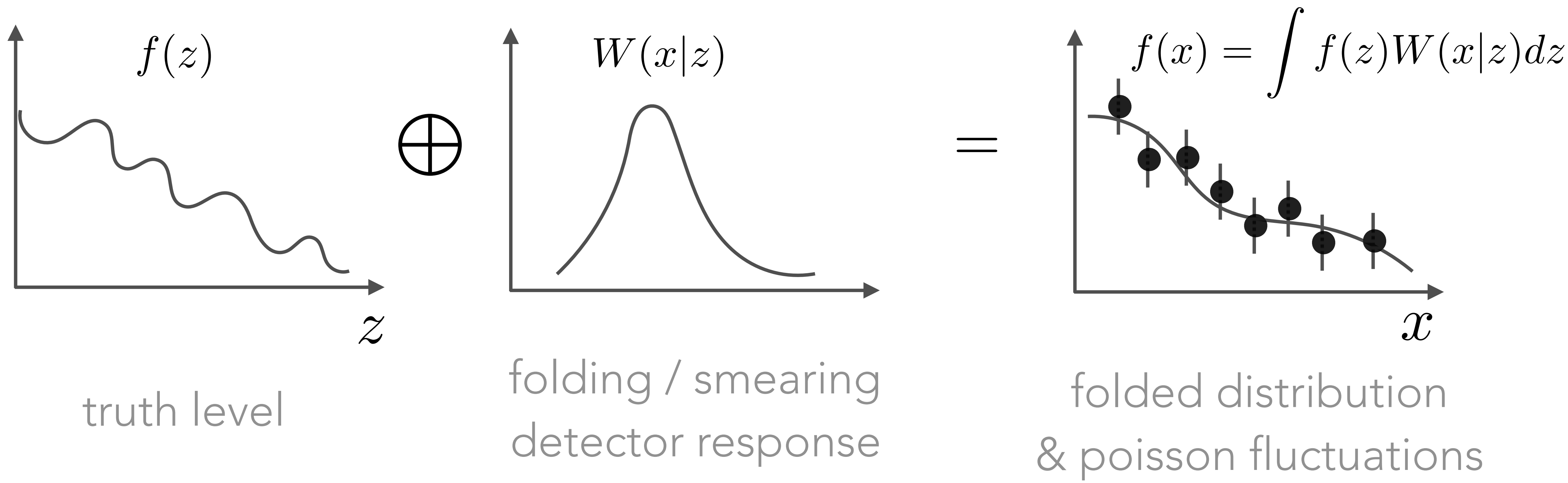
Gaussian

* the dimension is the number of data points.

A PARTICLE PHYSICS EXAMPLE OF A GAUSSIAN PROCESS

Consider unfolding when the detector response / "folding matrix" is known exactly (eg. no systematic uncertainty in detector response).

- the bin counts of observed distribution are uncorrelated Poisson fluctuations.



The unfolding process gives us a best estimate for unfolded distribution $f(z_i)$ and covariance matrix (eg. $f(z_i)$ and $f(z_{i+1})$ are usually highly correlated)

- the result of unfolding can be considered a Gaussian Process (GP).
- Gaussian Processes can be generalized to continuous z (unbinned distribution)

$$\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{f_\theta}(\boldsymbol{x}),\, K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})\right)$$

where

$$[K_{\boldsymbol{\alpha}}(\boldsymbol{x},\, \boldsymbol{\sigma})]_{ij} = {\sigma_i}^2\, \delta_{ij} + \underbrace{k_{\boldsymbol{\alpha}}(x_i,\, x_j)}_{\text{kernel function}}$$

kernel function
(where the magic happens)

see: gaussianprocess.org/**gpml** github.com/**dfm/george**

$$\log p(\boldsymbol{y}\,|\,\boldsymbol{x},\,\boldsymbol{\sigma},\,\boldsymbol{\theta},\,\boldsymbol{\alpha}) = -\frac{1}{2}\,[\boldsymbol{y} - \boldsymbol{f_\theta}(\boldsymbol{x})]^{\mathrm{T}}\,K_{\boldsymbol{\alpha}}(\boldsymbol{x},\,\boldsymbol{\sigma})^{-1}\,[\boldsymbol{y} - \boldsymbol{f_\theta}(\boldsymbol{x})] - \frac{1}{2}\log\det K_{\boldsymbol{\alpha}}(\boldsymbol{x},\,\boldsymbol{\sigma}) - \frac{N}{2}\log 2\,\pi$$

where

$$[K_{\boldsymbol{\alpha}}(\boldsymbol{x},\,\boldsymbol{\sigma})]_{ij} = {\sigma_i}^2\,\delta_{ij} + k_{\boldsymbol{\alpha}}(x_i,\,x_j)$$

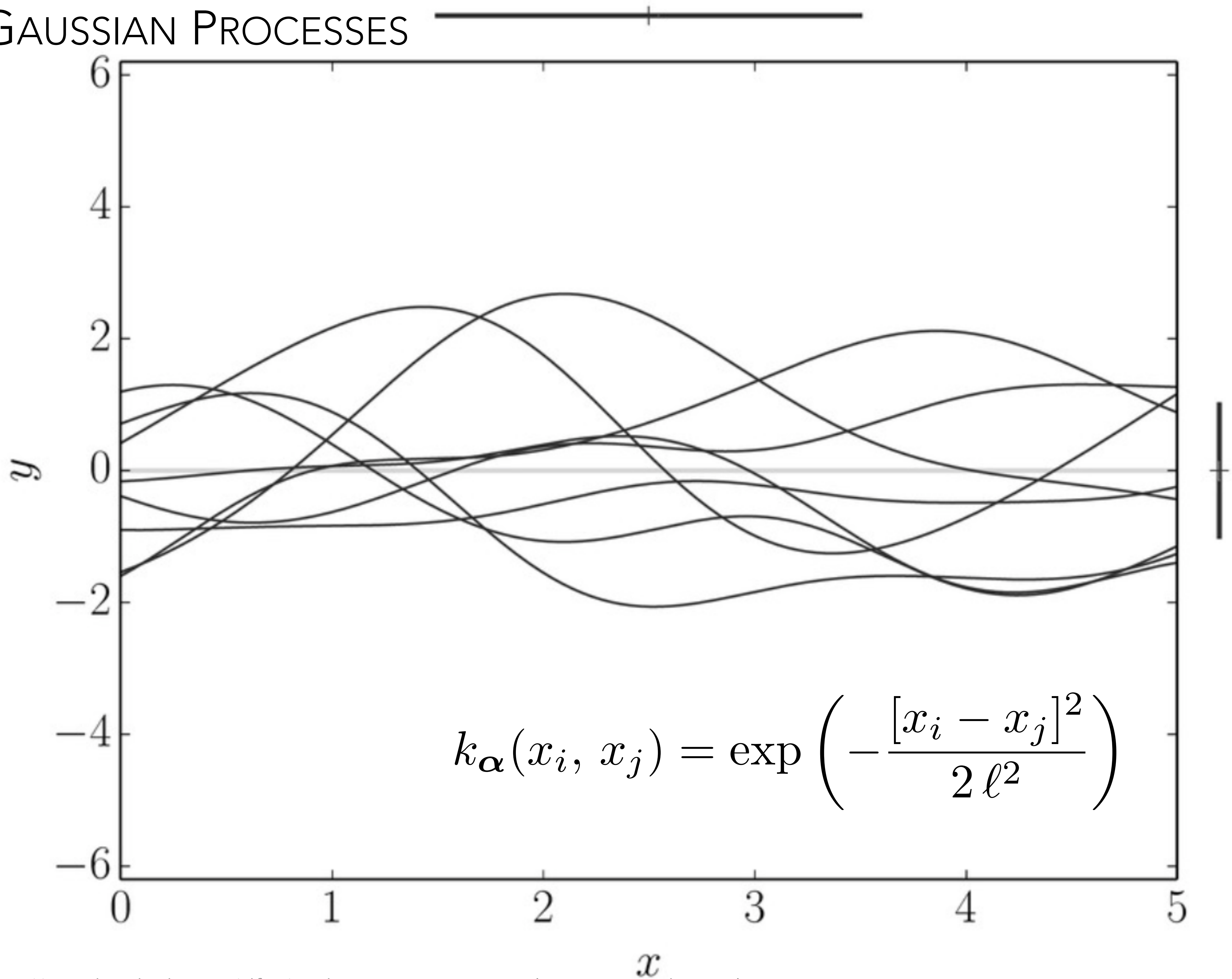
kernel function

(where the magic happens)

see: gaussianprocess.org/**gpml** github.com/**dfm/george**

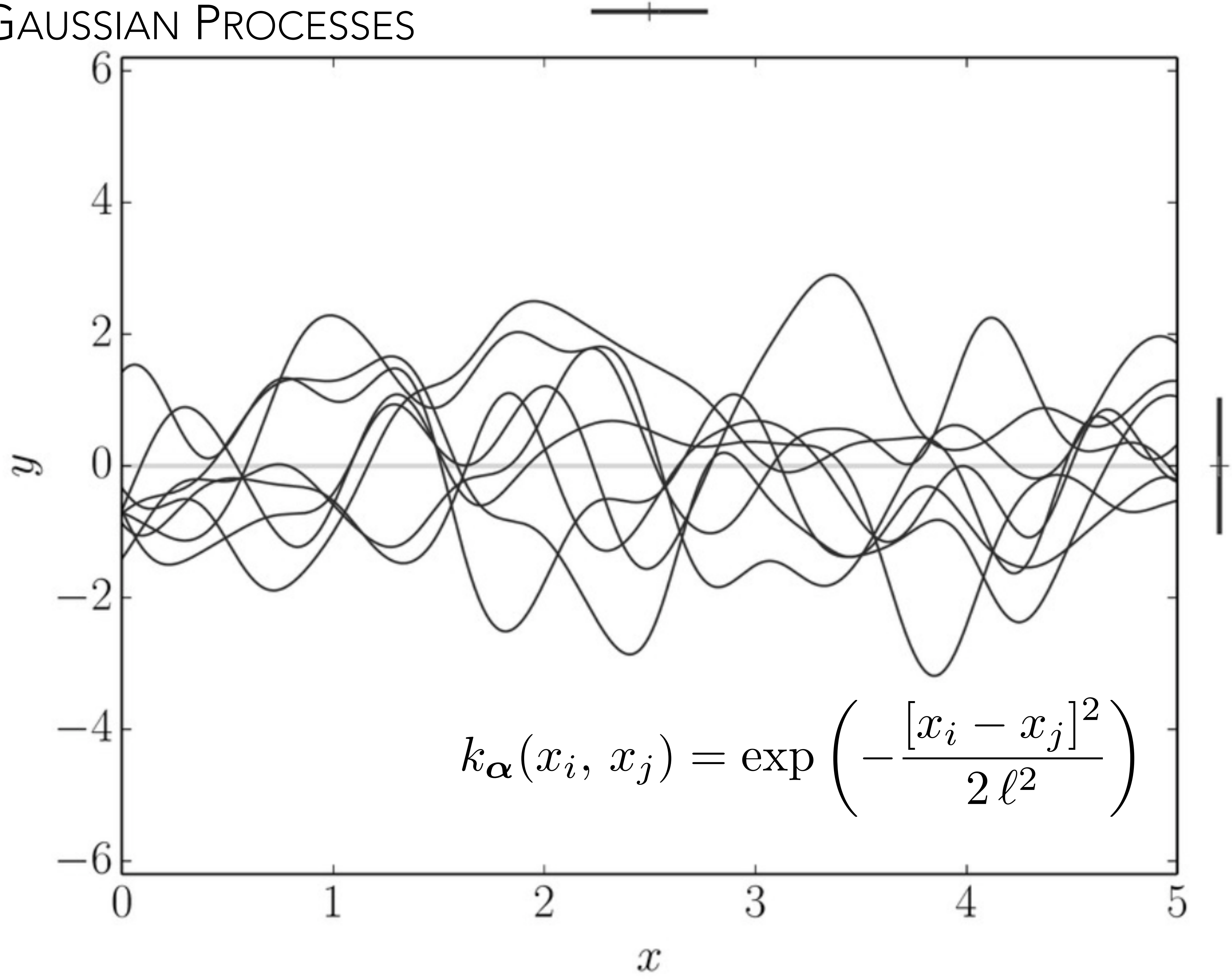
Gaussian Processes

$$k_{\boldsymbol{\alpha}}(x_i,\, x_j) = \exp\left(-\frac{[x_i - x_j]^2}{2\,\ell^2}\right)$$



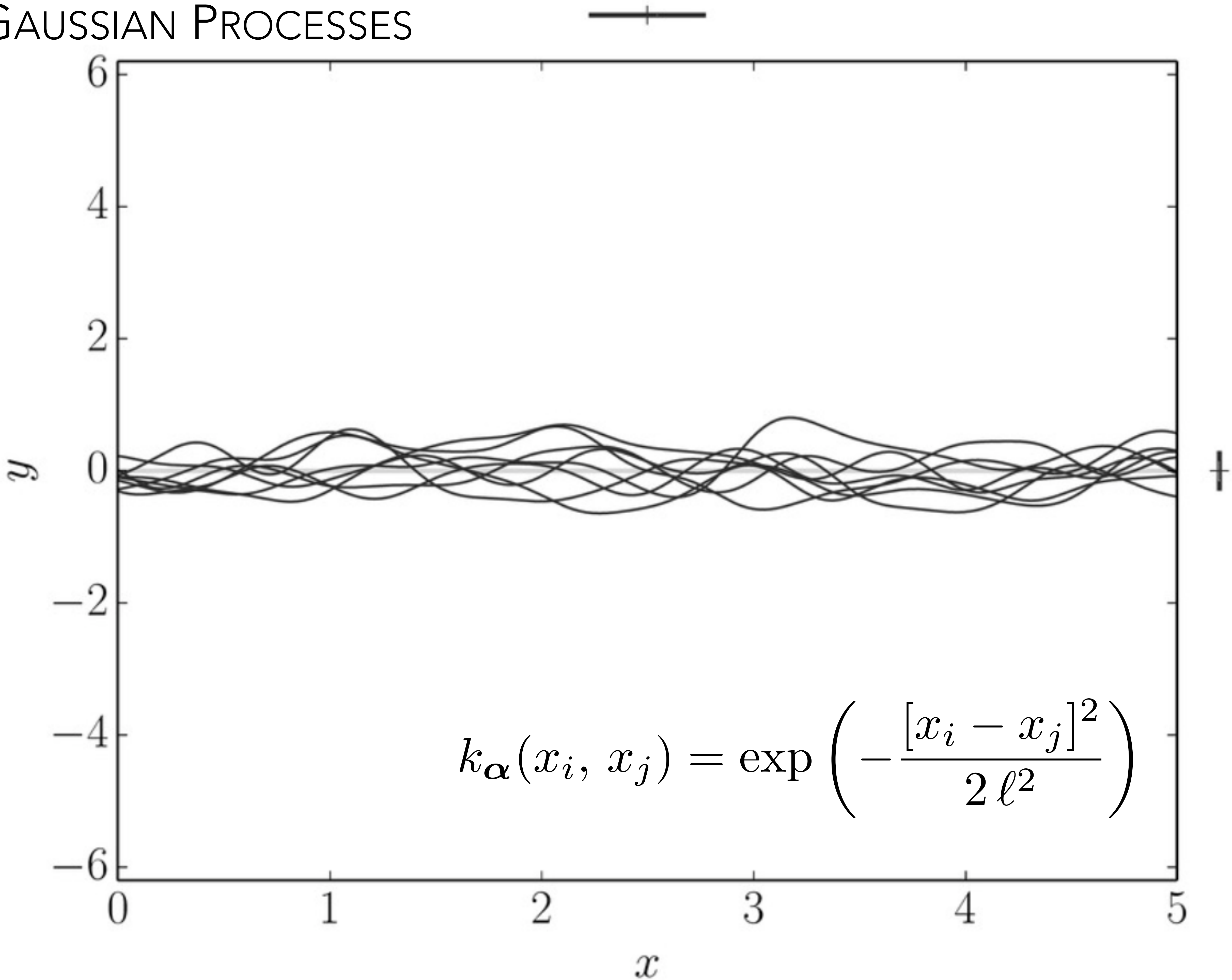
https://speakerdeck.com/dfm/pydata-time-series-analysis-gps-and-exoplanets

GAUSSIAN PROCESSES



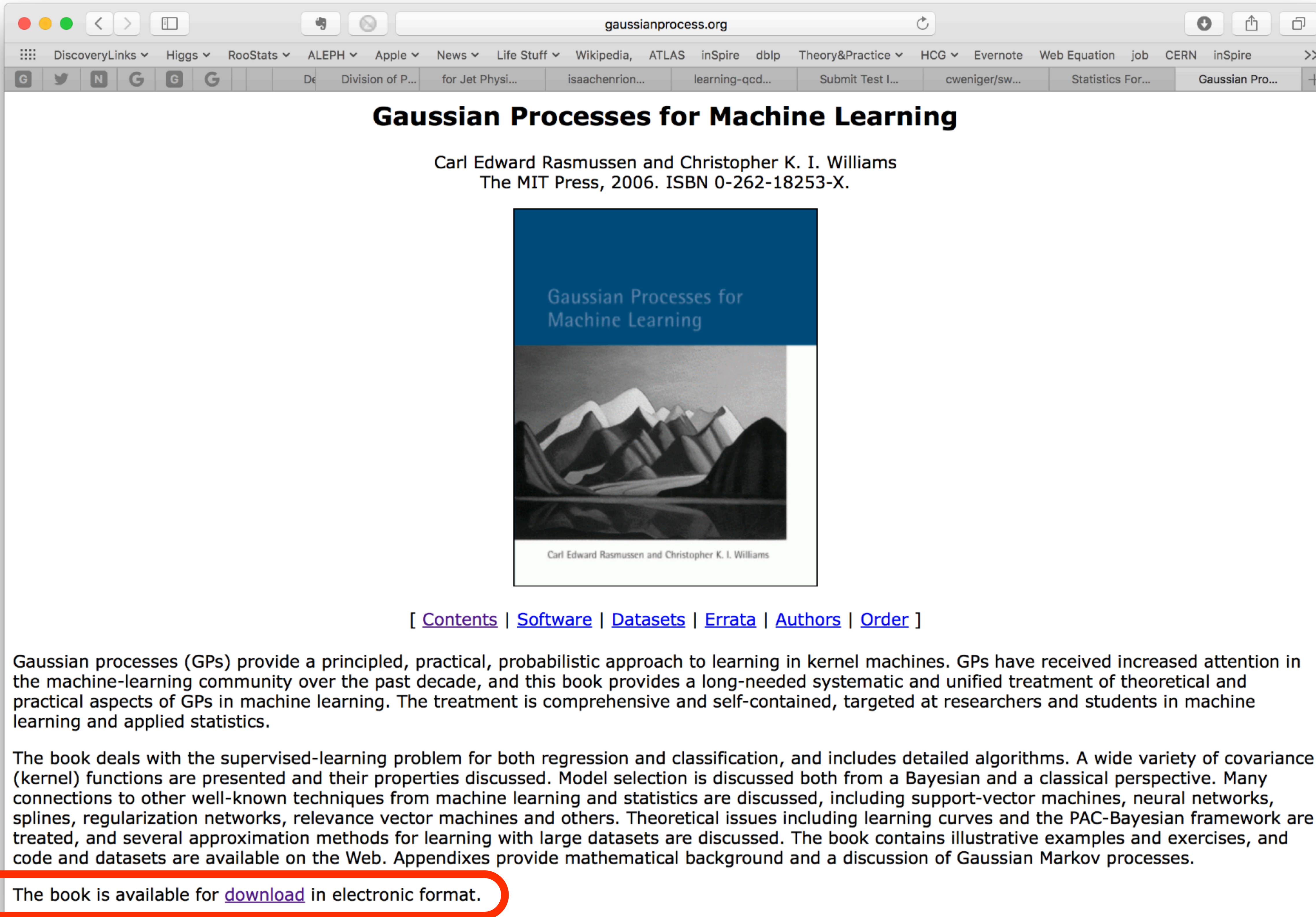
https://speakerdeck.com/dfm/pydata-time-series-analysis-gps-and-exoplanets

Gaussian Processes



https://speakerdeck.com/dfm/pydata-time-series-analysis-gps-and-exoplanets

LEARN MORE



Gaussian Processes for Machine Learning

Carl Edward Rasmussen and Christopher K. I. Williams
The MIT Press, 2006. ISBN 0-262-18253-X.

[Contents | Software | Datasets | Errata | Authors | Order]

Gaussian processes (GPs) provide a principled, practical, probabilistic approach to learning in kernel machines. GPs have received increased attention in the machine-learning community over the past decade, and this book provides a long-needed systematic and unified treatment of theoretical and practical aspects of GPs in machine learning. The treatment is comprehensive and self-contained, targeted at researchers and students in machine learning and applied statistics.

The book deals with the supervised-learning problem for both regression and classification, and includes detailed algorithms. A wide variety of covariance (kernel) functions are presented and their properties discussed. Model selection is discussed both from a Bayesian and a classical perspective. Many connections to other well-known techniques from machine learning and statistics are discussed, including support-vector machines, neural networks, splines, regularization networks, relevance vector machines and others. Theoretical issues including learning curves and the PAC-Bayesian framework are treated, and several approximation methods for learning with large datasets are discussed. The book contains illustrative examples and exercises, and code and datasets are available on the Web. Appendixes provide mathematical background and a discussion of Gaussian Markov processes.

The book is available for download in electronic format.

http://www.gaussianprocess.org/gpml/

Parametrized Function

vs.

Gaussian Process

PARAMETRIC FUNCTION VS. GP

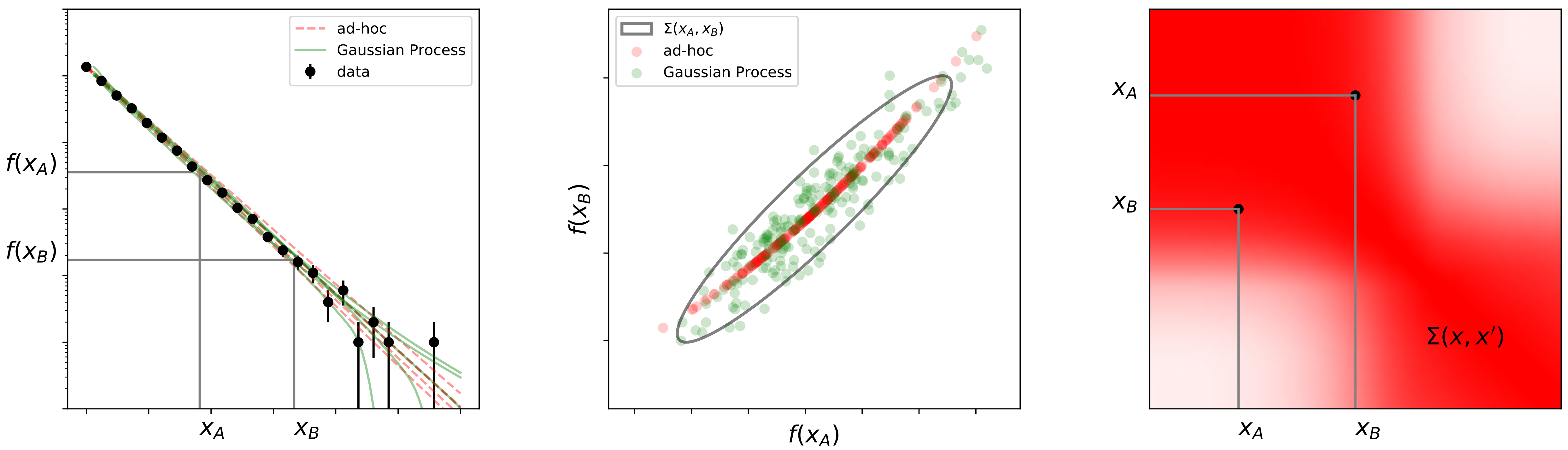


FIG. 2: Schematic of the relationship between an ad-hoc function and the GP. An example toy dataset is shown (left) with samples from the posterior for an ad-hoc 1-parameter function (red) and a GP (green). Each posterior sample is an entire curve $f(x)$, which corresponds to a particular point in the (center) plane of $f(x_A)$ vs. $f(x_B)$. The red dots for the ad-hoc 1-parameter function trace out a 1-dimensional curve, which reveals how the function is overly-rigid. In contrast, the green dots from the GP relax the assumptions and fill a correlated multivariate Gaussian (with covariance indicated by the black ellipse). The covariance kernel $\Sigma(x, x')$ for the GP is shown (right) with $\Sigma(x_A, x_B)$ corresponding to the black ellipse of the center panel.

MOAR DATA!

GP fits the background well, and continues to as we add more data. Parametric function no longer fits well

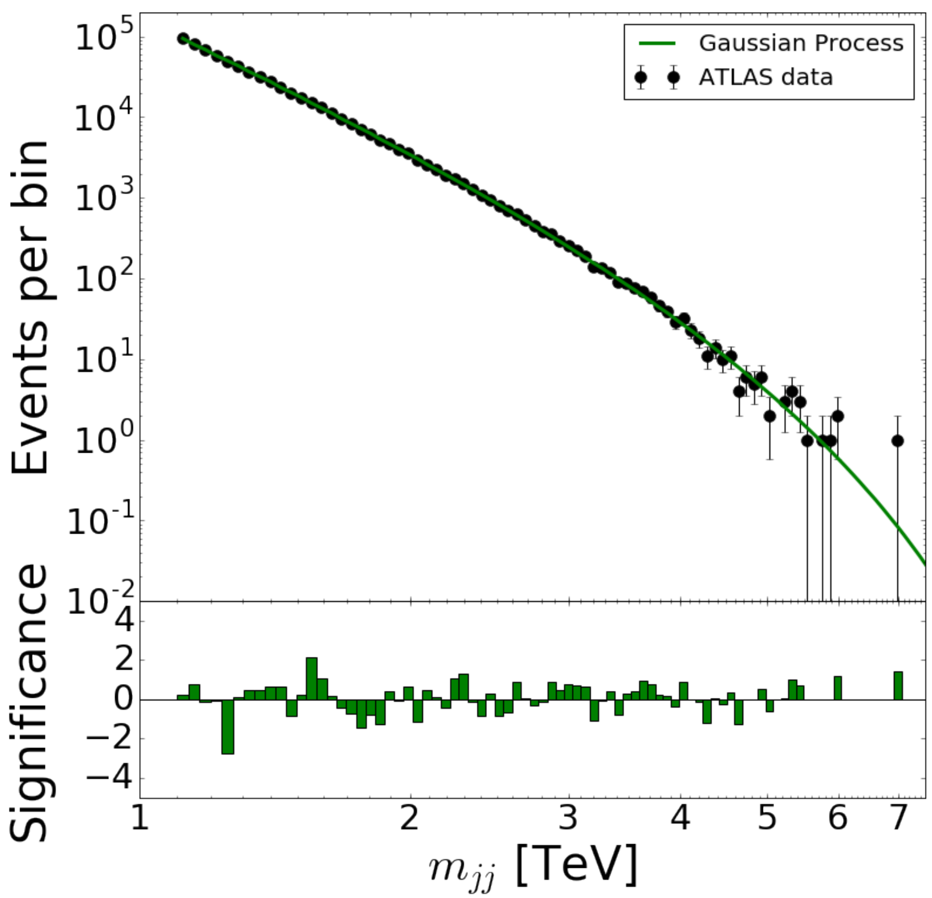
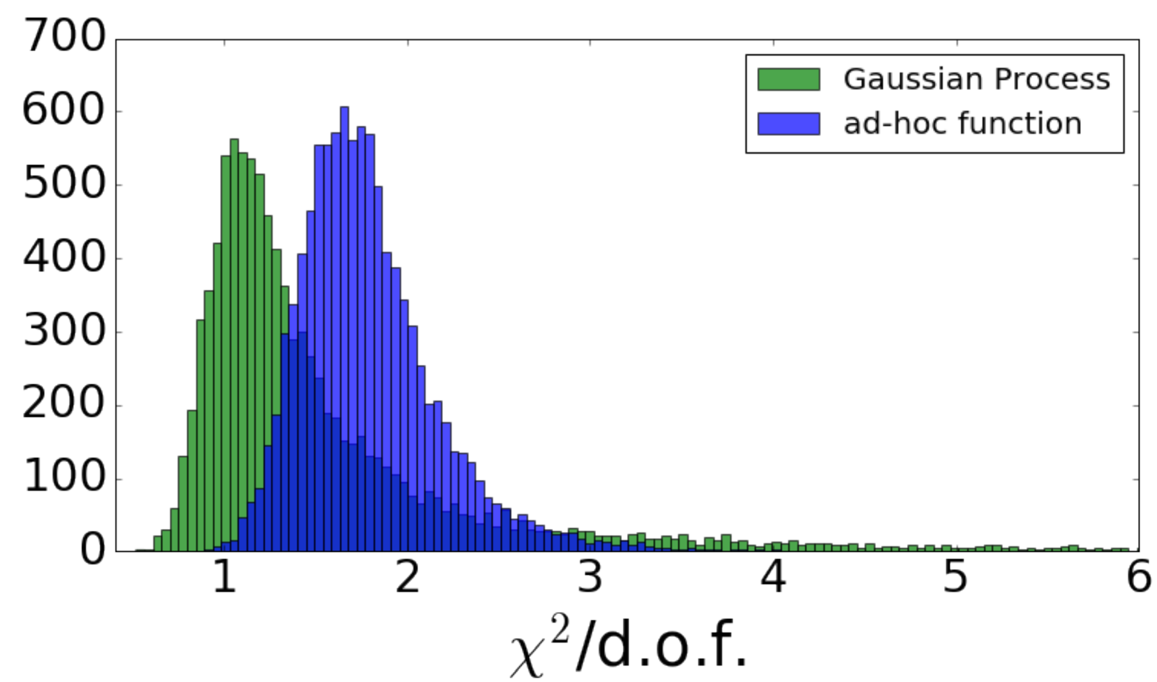
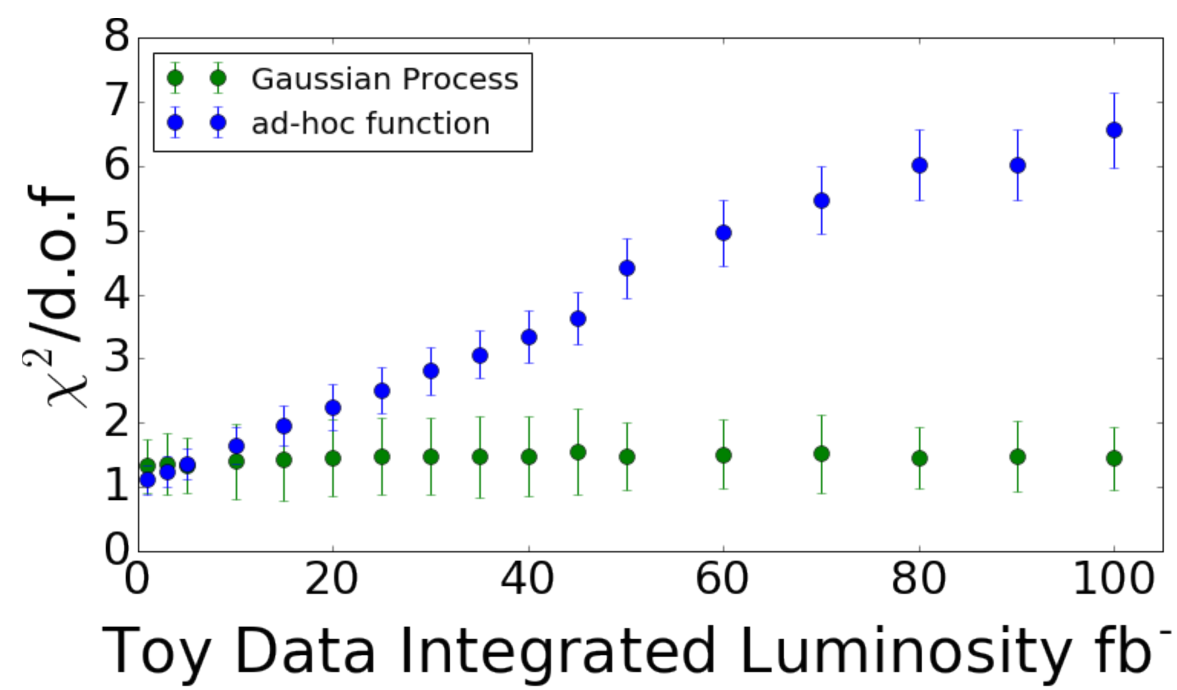


FIG. 5: Invariant mass of dijet pairs reported by ATLAS [15] in proton-proton collisions at $\sqrt{s} = 13$ TeV with integrated luminosity of 3.6 fb^{-1}. The green line shows the resulting Gaussian process background model. The bottom pane shows the significance of the residual between the data and the GP model.

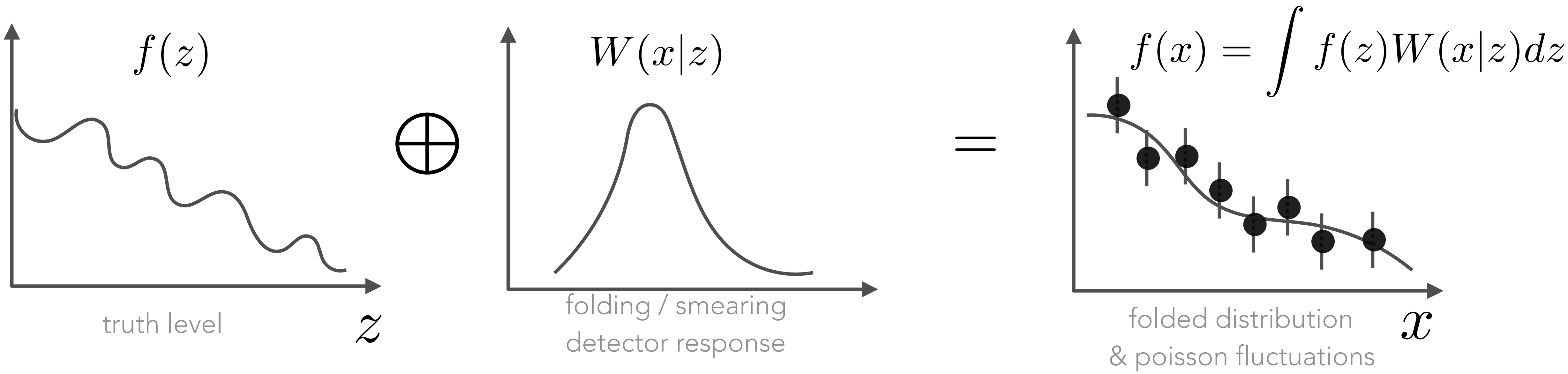




Physically Motivated Kernels

If the truth level distribution f(z) is a Gaussian Process with kernel $\Sigma(z, z')$, then the reconstructed distribution f(x) is also a Gaussian process with $\Sigma(x,x')$

$$\Sigma(x,x') = \int\int dz dz' \Sigma(z,z') W(x,z) W(x',z') \qquad (8)$$



If we are making predictions with Monte Carlo, truth level distribution f(z) is usually known exactly.

To think of f(z) as a Gaussian Process, we need some notion of uncertainty (eg. parton density functions, higher-order corrections, renormalization/factorization scales)

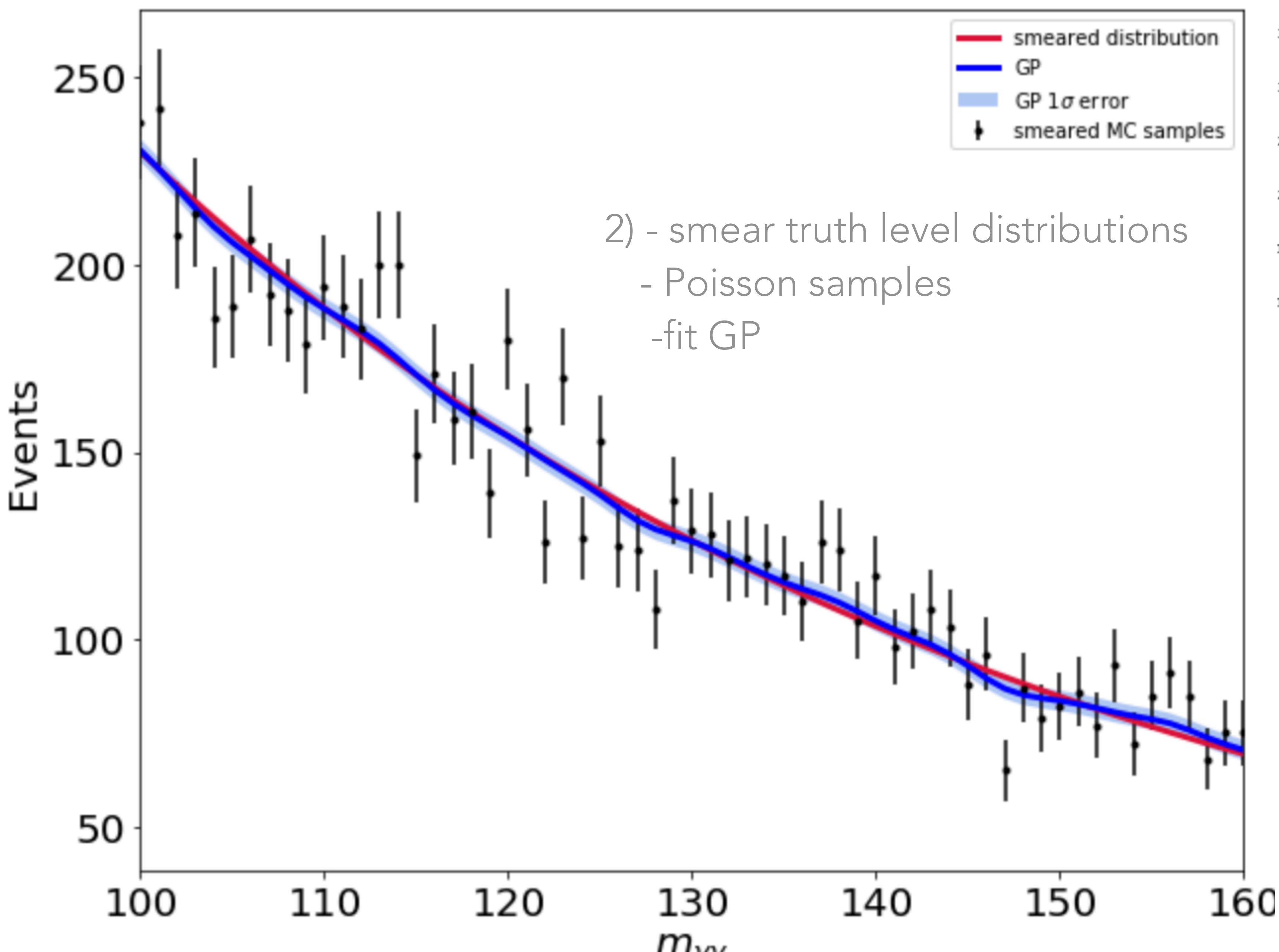
In unfolding, we often don't want to make assumptions about f(z)… it could be anything. But regularization in unfolding is equivalent to choosing a kernel for f(z).

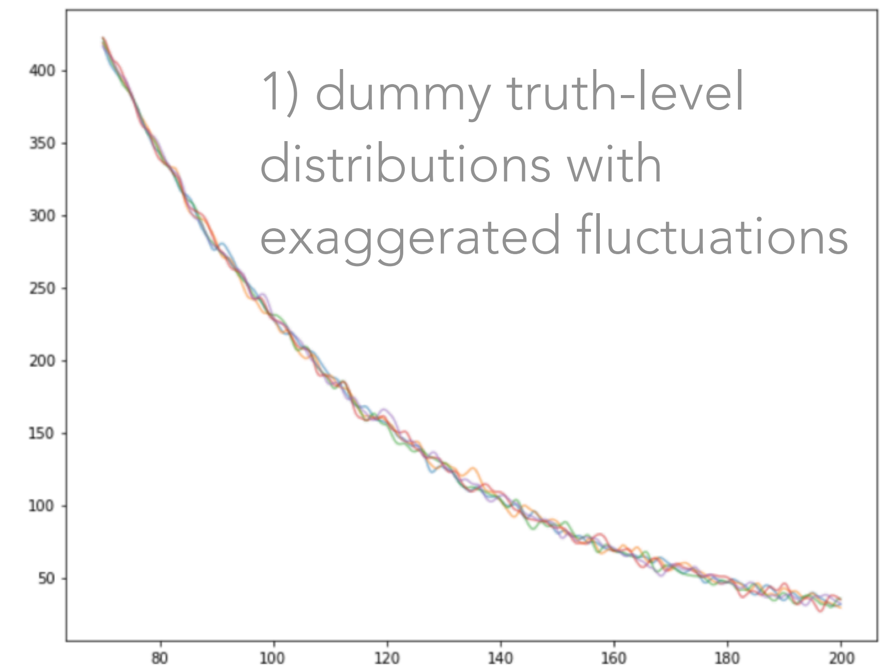
Even in extreme case where we assume no smoothness in f(z), f(x) has to be smooth due to detector resolution.

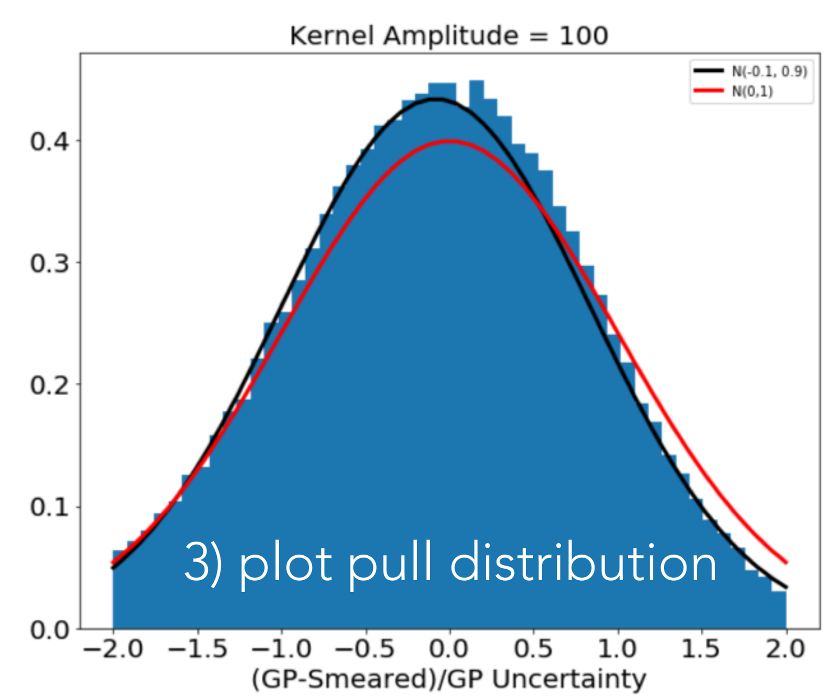
MC-TEMPLATE SMOOTHER

In $H \to \gamma\gamma$, we have used functional forms, like Bernstein polynomial. We "trust" the Monte Carlo, and assign "spurious signal" to account for differences between MC and functional form, but MC Stat error is a limiting factor for spurious signal etc.

Alternate idea: fit GP to MC histogram. No functional form assumed. Here only assume length scale must be > $\sqrt{2}$ mass resolution.

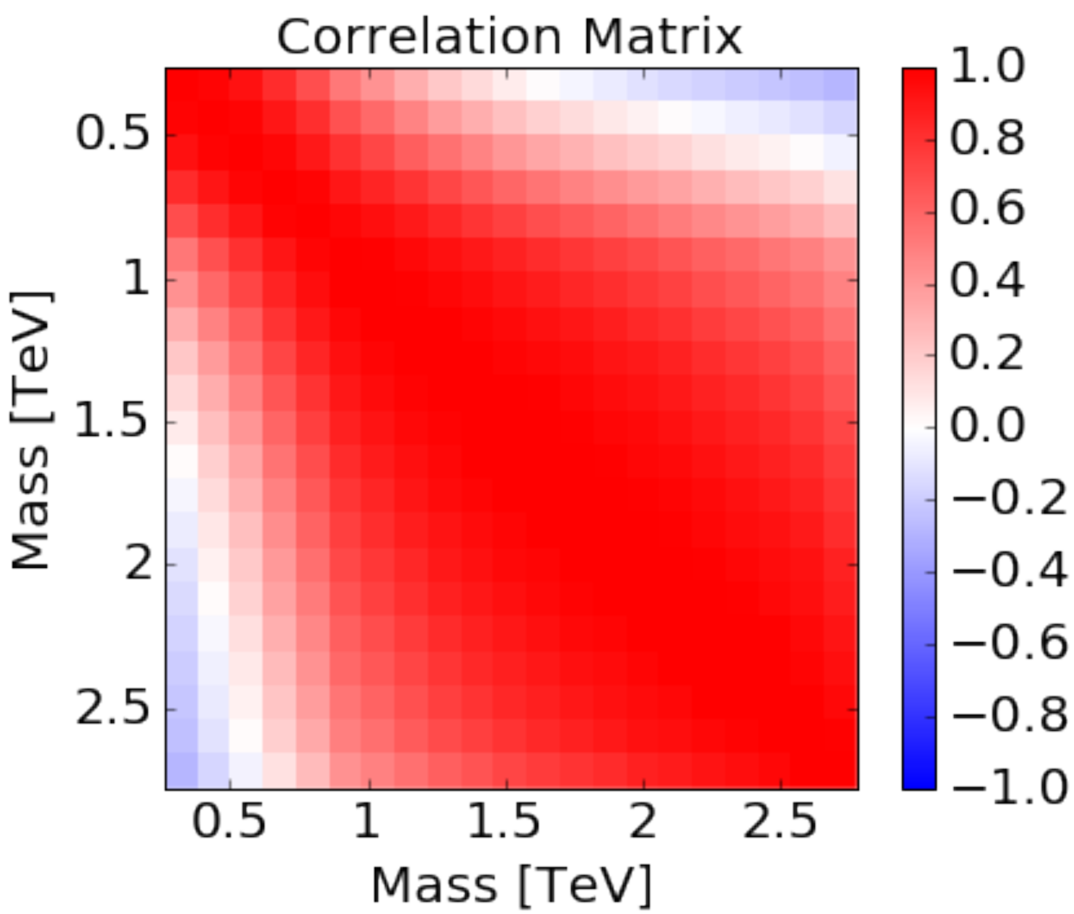


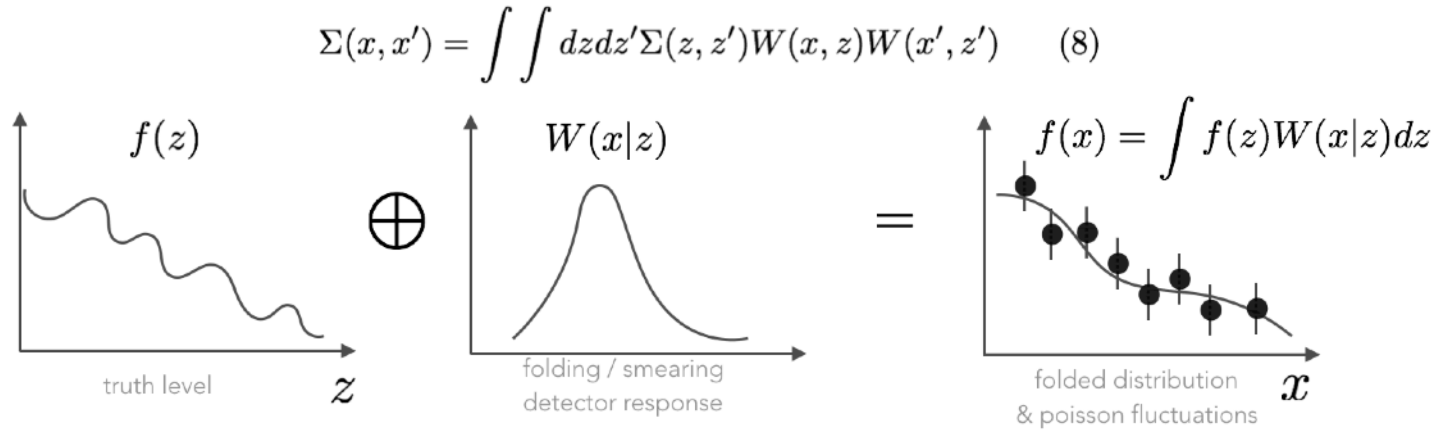




https://github.com/mfrate28/gp-mc-template-smoother/blob/master/GPsmearedFitting.ipynb

EXAMPLE: PDF UNCERTAINTIES



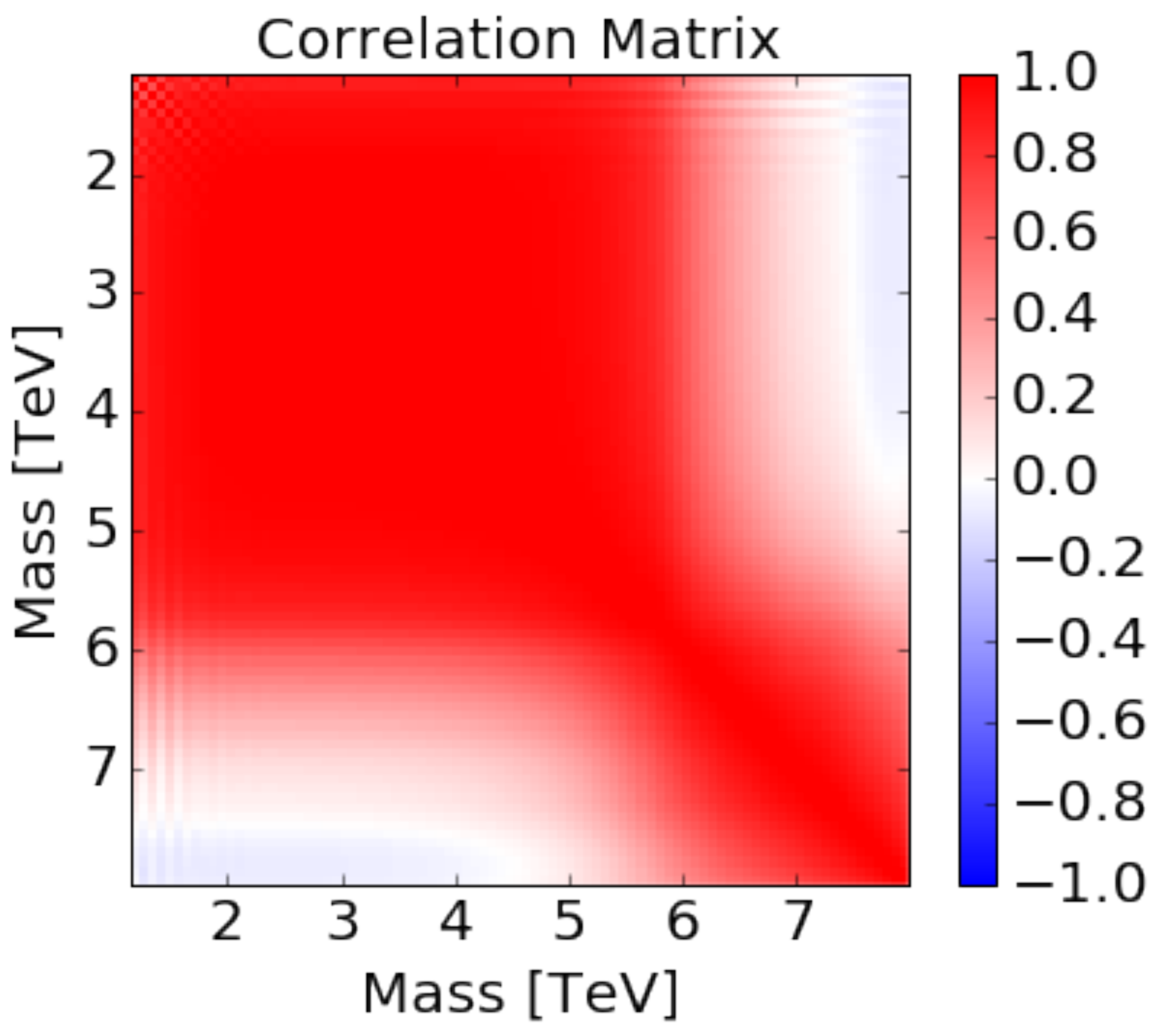


Here we focused on truth level distribution f(z).

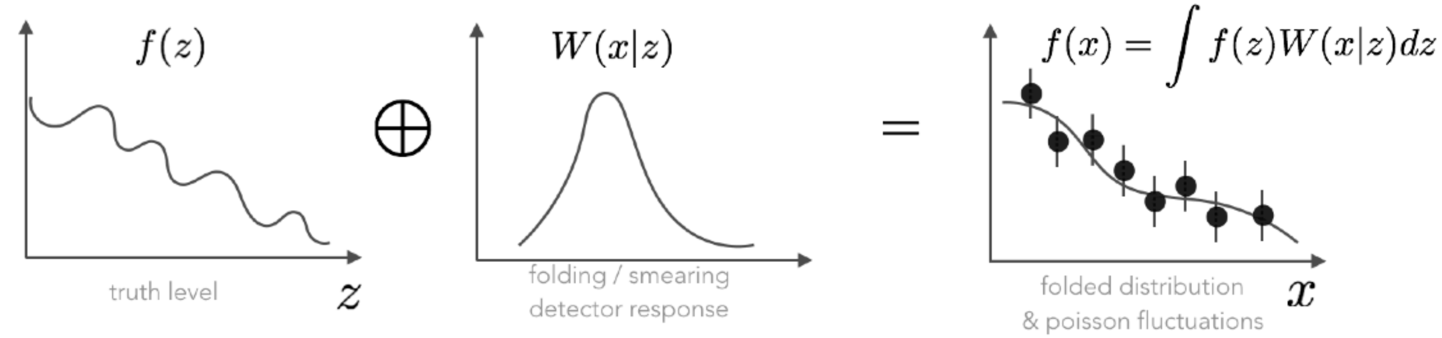
Used dijet spectrum predicted at NLO with POWHEG-BOX and look into PDF uncertainties from NNPDF3

This is the PDF uncertainty in the truth distribution expressed as a Gaussian Process Kernel.

EXAMPLE: JET ENERGY SCALE



$$\Sigma(x,x') = \int\int dz dz' \Sigma(z,z') W(x,z) W(x',z') \quad (8)$$

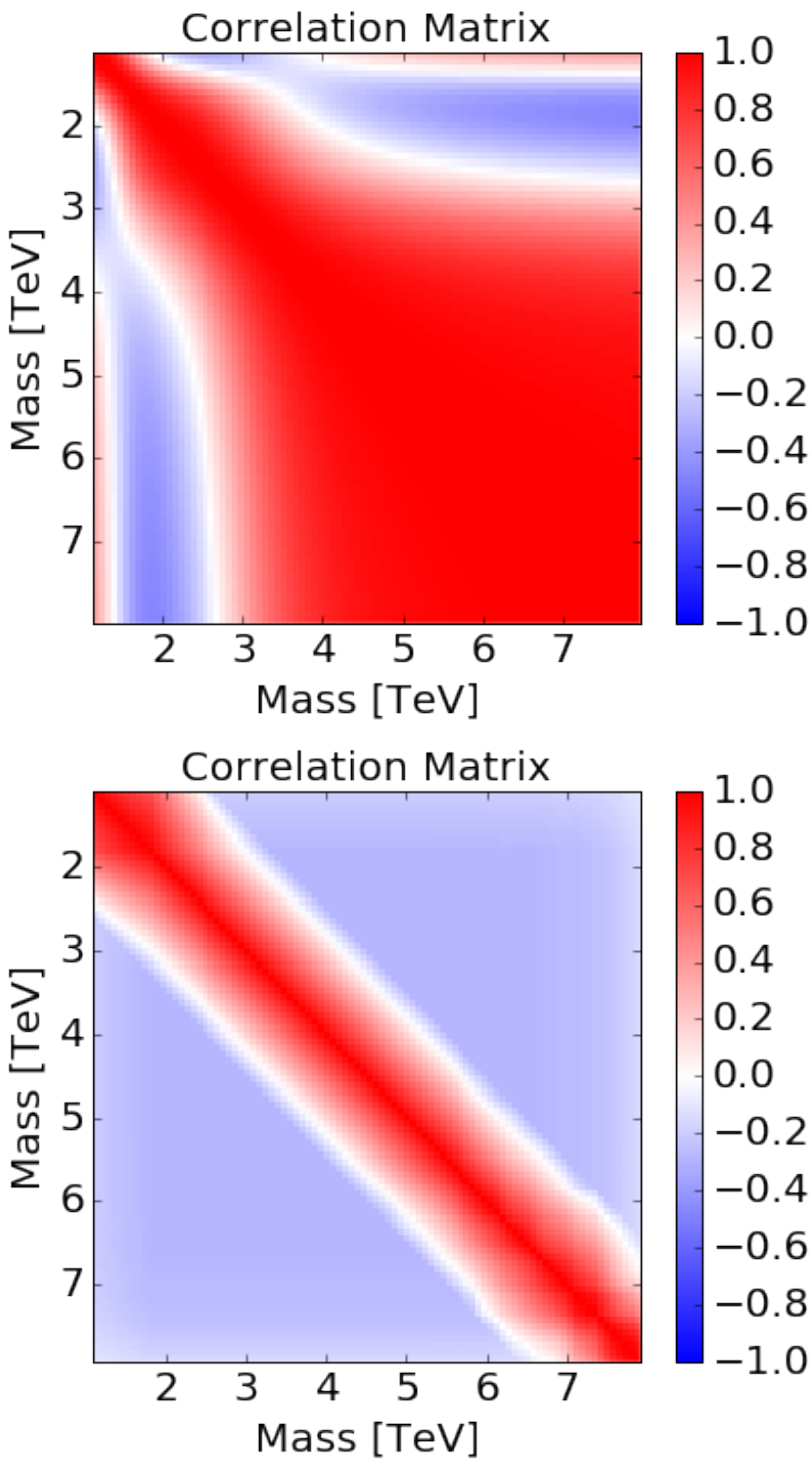


Take for example, the jet energy scale (JES) uncertainty. As described in Refs. [17, 27] the ATLAS JES uncertainty is only a few percent for jets with p_T of around 1 TeV where data are plentiful, while the the limited size of observed examples for higher-p_T jets requires an alternate approach to estimating the JES. The resulting JES uncertainty therefore grows rapidly with m_{jj} and has an impact of at most 15% [27]. To illustrate the covariance due to the JES uncertainty, consider a simplified two-parameter model for the impact on the m_{jj} distribution: $J(z,\theta) = 1 + 15\%\,\theta_1 z^4 + 5\%\,\theta_2(1-z)$, where z is the true dijet invariant mass and $z_{\max} = 7$ TeV. We use the best fit 3-parameter fit as a proxy for $f(z)$ and fold in the smearing $W(x|z,\theta) = \text{Gaus}(x|z\,J(z/z_{\max},\theta),\sigma_x)$, where $\sigma_x = 2\%z$ is the dijet invariant mass resolution [17]. By assuming a uniform prior and an appropriate scaling for θ, we sample from the posterior $\text{Gaus}(\theta_1|0,1)\text{Gaus}(\theta_2|0,1)$ and propagate the uncertainty in θ through to the predicted bin counts $\bar{f}(\mathbf{x}|\theta)$ as in Eqs. 4 and 5. This allows us to explicitly build the covariance matrix Σ using the simulation shown in Fig. 3. As expected, we see a roughly block-diagonal structure defined by low and high mass regions.

Even if the truth-level distribution is known, the folding matrix may not be known exactly.

Example: consider a jet energy scale with 2 nuisance parameters, where one parameter dominantly affects low-pT jets (in situ) and the other high-pT jets (limited stats for in situ).

Propagate uncertainty in jet energy scale to reconstructed mjj spectrum, obtain covariance kernel.

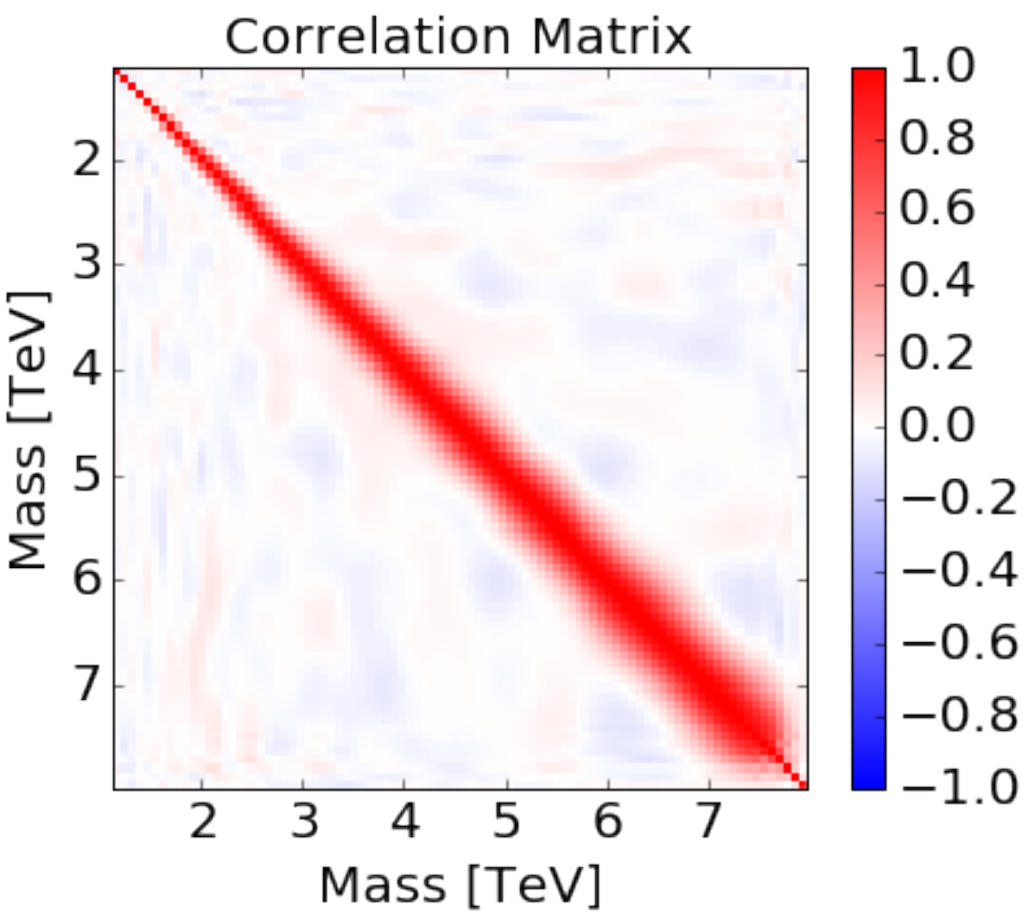


We can also think of the covariance structure for current fitting strategies.

- top: 3-parameter dijet function
- bottom: sliding window (SWIFT)

These are post-fit covariance plots.

POST-FIT PARAMETRIZED DIJET KERNEL



$$\Sigma(x,x') = \; Ae^{\frac{d-(x+x')}{2a}} \sqrt{\frac{2l(x)l(x')}{l(x)^2+l(x')^2}} e^{\frac{-(x-x')^2}{l(x)^2+l(x')^2}}$$

In addition to kernels constructed bottom-up from first-principles, we can also construct parametrized kernels using some intuition.

GPs adapt to the data very well, so even simple exponential-squared kernels often work fine.

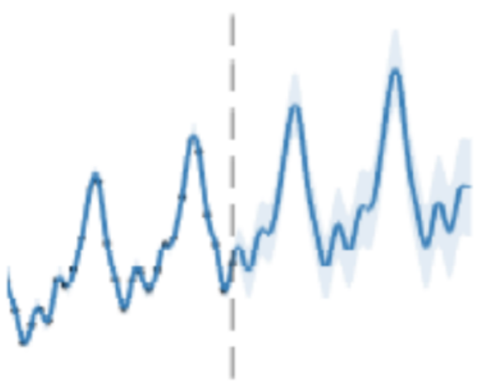
For our dijet studies, we used a "Gibbs kernel", which has length scale $l(x)$ and amplitude vary with x

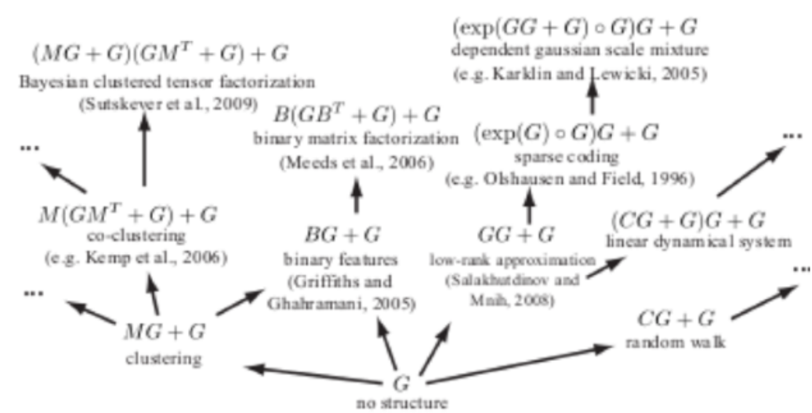
- plot shows post-fit covariance kernel

FUTURE DIRECTIONS

Vocabulary of kernels + grammar for composition

- physics goes into the construction of a "Kernel" that describes covariance of data

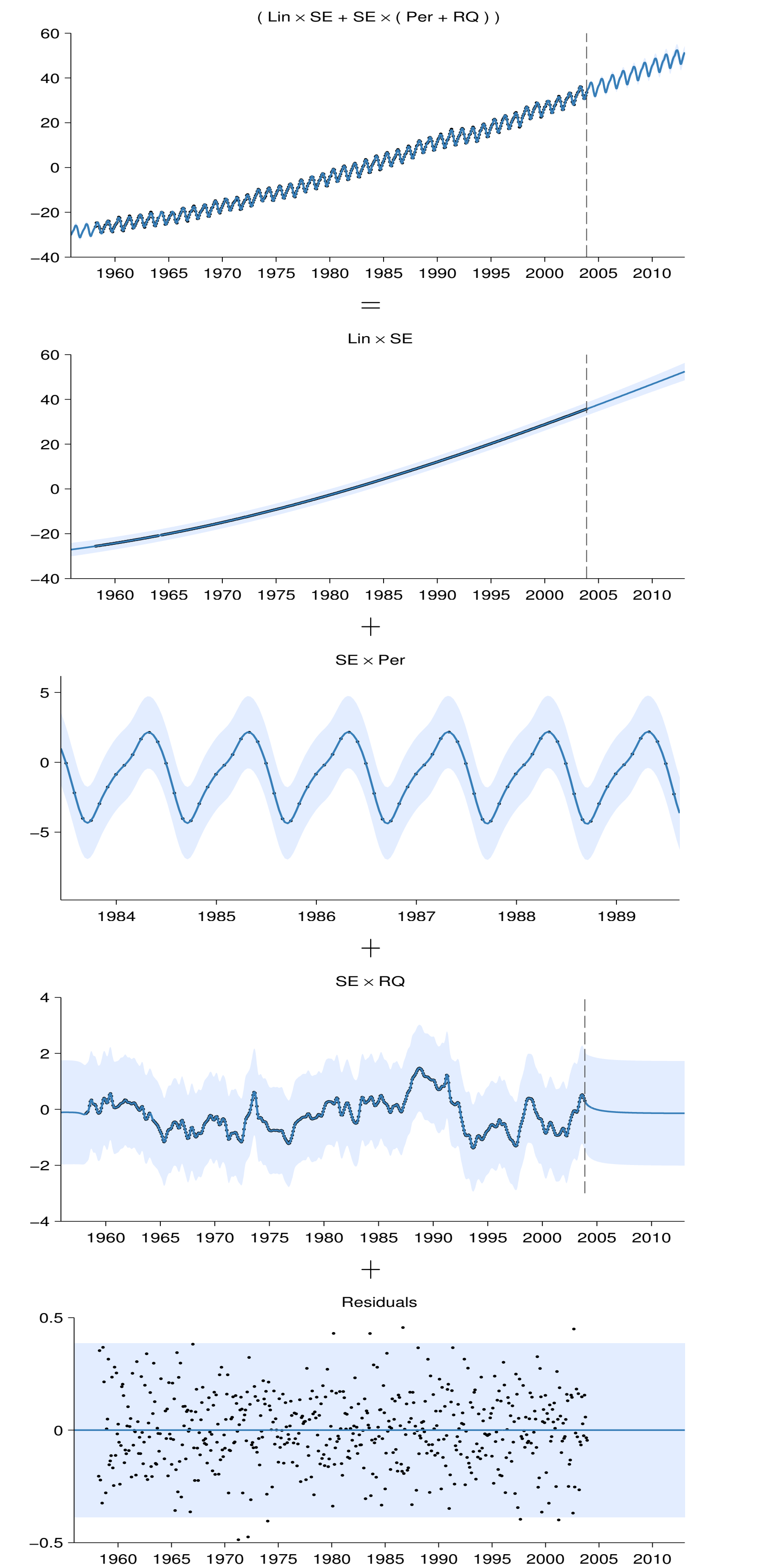




Structure Discovery in Nonparametric Regression through Compositional Kernel Search
David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, Zoubin Ghahramani
International Conference on Machine Learning, 2013
pdf | code | poster | bibtex

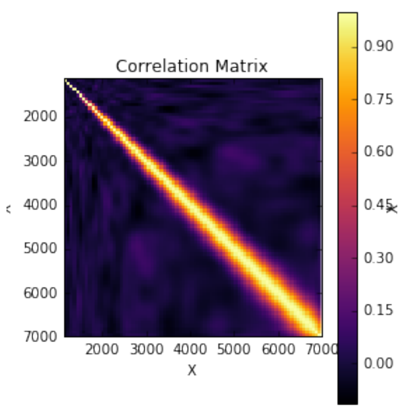
Exploiting compositionality to explore a large space of model structures
Roger Grosse, Ruslan Salakhutdinov, William T. Freeman, Joshua B. Tenenbaum
Conference on Uncertainty in Artificial Intelligence, 2012
pdf | code | bibtex

Mauna Loa atmospheric CO_2



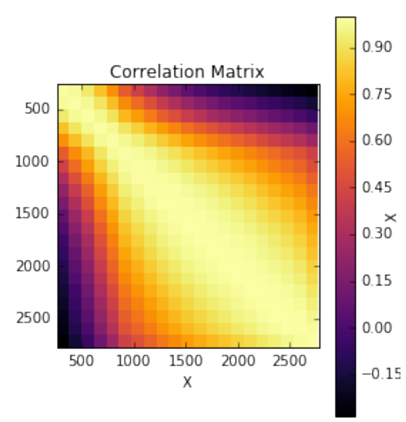
with Meghan Frate
& Daniel Whiteson

Instead of fitting the dijet spectrum with an ad hoc 3-5 parameter function, use GP with kernel motivated from physics



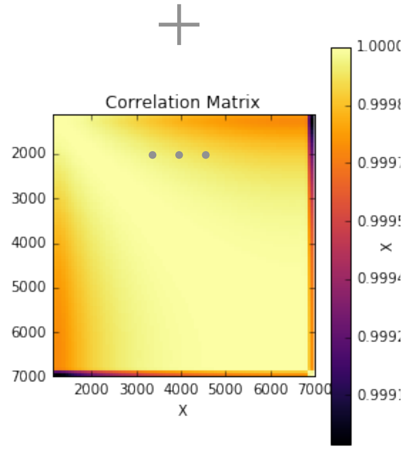
Final Kernel =

=



Poisson stats
+ Mass Resolution

+

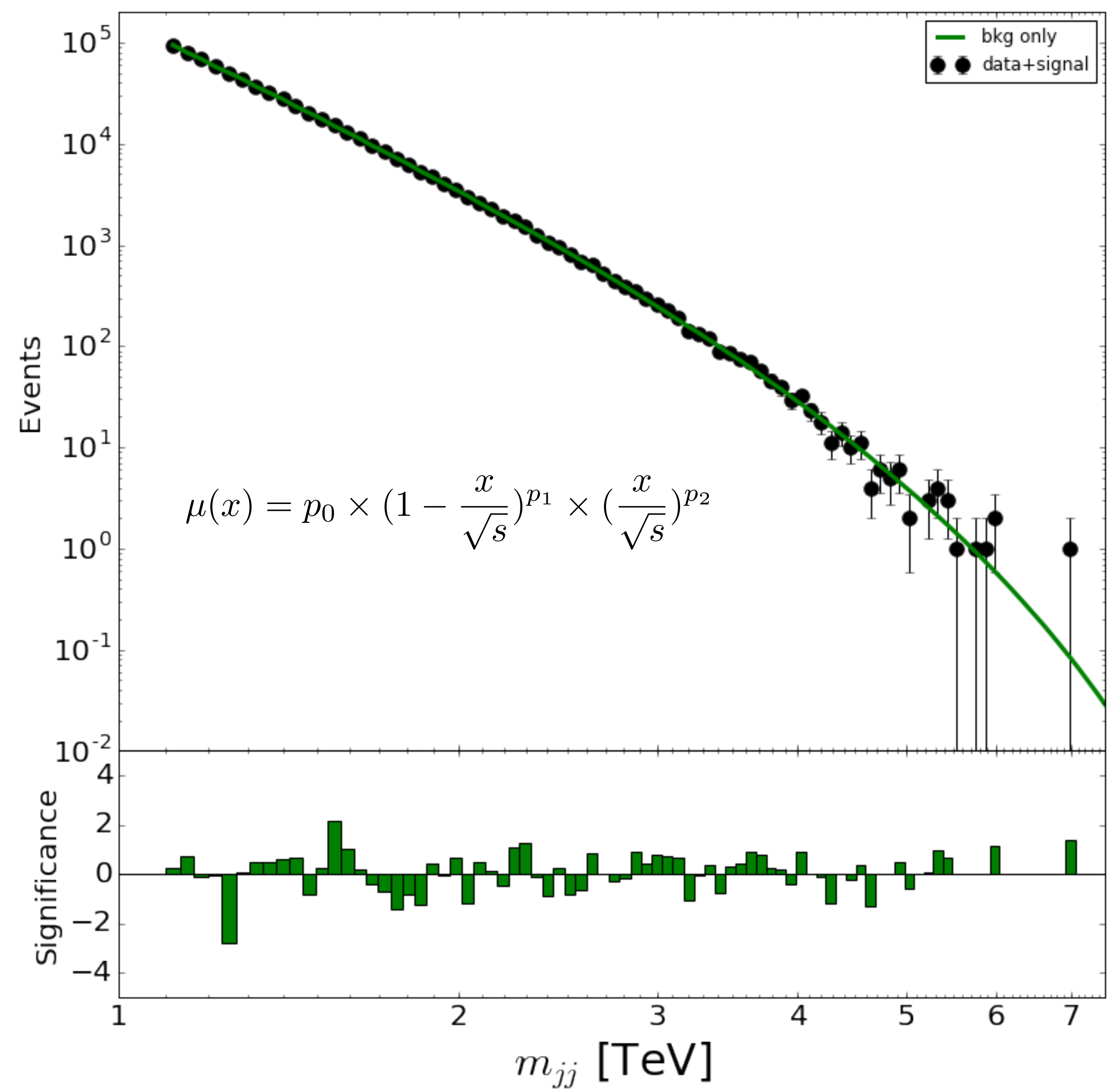


+ Parton Density Functions

+ Jet Energy Scale

+

...



Integration into our Statistical Procedures

The statistical interpretation of GPs can be a bit subtle. Specifically Bayesian vs. Frequentist issues

- Most GP literature is presented in a Bayesian formalism
- the GP is usually thought of as a prior over functions, and the result of the fit is posterior given observations
- then usually fit "hyperparameters" of kernel using marginal likelihood

However, also consistent to think of the GP likelihood where the kernel represents auxiliary measurements / constraint terms.

A third interpretation is that the kernel represents the penalty term in "penalized maximum likelihood" in the spirit of regularization in unfolding

Integration of GPs into our statistical procedures can be done in a few ways.

- start with our typical extended maximum likelihood for a statistical model parametrized by θ

$$p(\mathcal{D}, \mathbf{a}|\theta) = \text{Pois}(N|\nu(\theta)) \prod_{e=1}^{N} p(x_e|\theta) \cdot p_{\text{constr.}}(\mathbf{a}|\theta) \ .$$

- If we are using a binned distribution in a high-statistics regime, and we approximate the **effect** of the constraint terms on the bin counts as a Gaussian, then we can approximate this as

$$\begin{aligned} p(\mathbf{y}, \mathbf{a}|\theta) = & \quad \textstyle\prod_{i=1}^{n} \text{Pois}(y_i|\bar{f}(x_i|\theta)) \cdot p_{\text{constr.}}(\mathbf{a}|\theta) \\ \approx & \quad \text{Gaus}(\mathbf{y}|\bar{f}(\mathbf{x}|\theta), \sigma^2) \cdot \text{Gaus}(\bar{f}(\mathbf{x}|\theta)|\mu, \mathbf{\Sigma}) \ , \end{aligned}$$

- The Poisson mean $\bar{f}(\mathbf{x}|\theta)$ can be a parametrized signal + a Gaussian Processs for the background.

Integration of GPs into our statistical procedures can be done in a few ways.

1. Fully Bayesian analysis using Poisson fluctuations about a GP mean. This is called a Cox process. Cumbersome to implement because it is "doubly stochastic"

2. Fit the total model (parametrized signal + background GP) to the data (assuming stat errors are Gaussian), use result as the mean $\mu(\mathbf{x}_*|\mathbf{y}) = \mu(\mathbf{x}_*) + \Sigma(\mathbf{x}_*, \mathbf{x})[\Sigma(\mathbf{x}, \mathbf{x}) + \sigma^2(\mathbf{x})\mathbf{I}]^{-1}(\mathbf{y} - \mu(\mathbf{x}))$ in standard likelihood Poisson likelihood

3. Fit the GP, use the posterior mean and covariance of the GP as a simple Gaussian likelihood / chi-square with covariance matrix.

We used option 2., most consistent with our existing statistical procedures

Here true hypothesis has no signal, but is neither the ad-hoc function nor the GP, so we don't expect it to be a chi-square exactly.

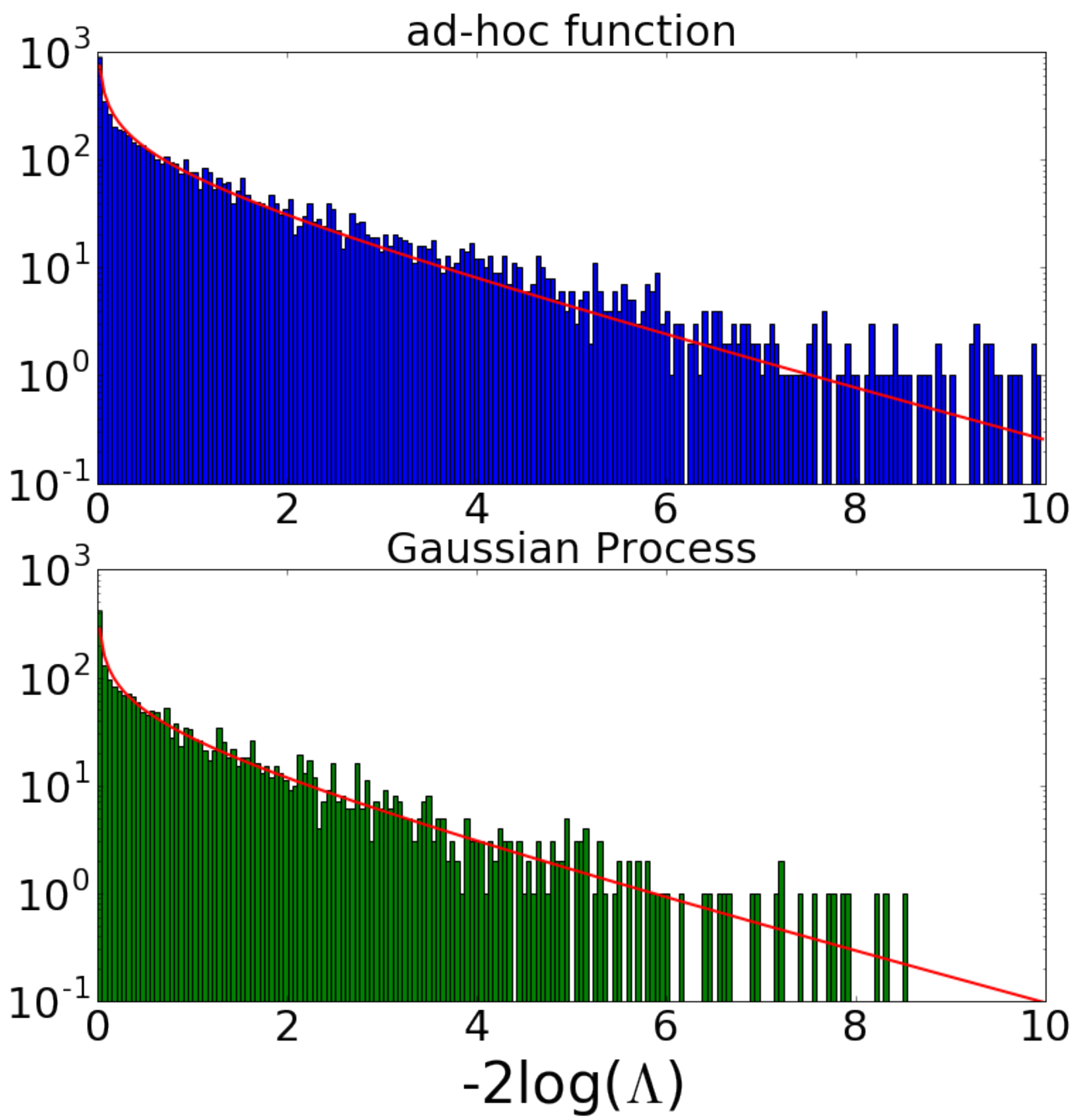
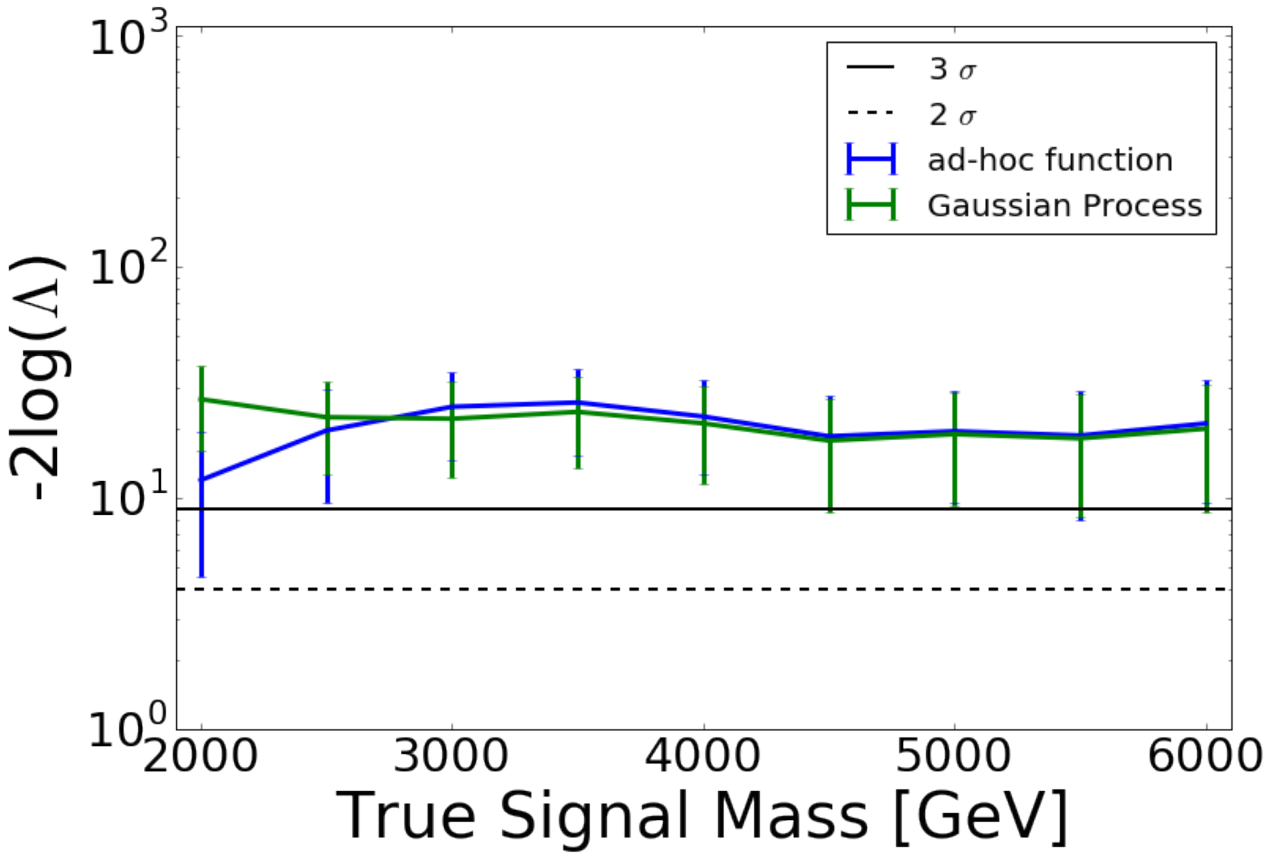


FIG. 11: Distribution of $-2\log(\Lambda)$, where Λ is the likelihood ratio between the background-only and the background-plus-signal hypotheses, for toy data with no signal present, shown for both the ad-hoc fit (top) and the Gaussian process background model (bottom). Overlaid in red is a χ^2 distribution with one degree of freedom.

Worry is GP might be too flexible.

So need to check expected significance (power) by injecting signal.

(Result depends on kernel used)



Modeling Generic Localized Signals

(related to spurious signal)

In many exotics searches, we don't want to assume a specific signal model.

- difficult to do likelihood-ratio based tests using shape information, since we don't know the signal's shape
- Instead, typically use **BumpHunter** and look for a localized signal in some mass window.
  - difficulties here because BumpHunter needs a global background estimate to do background-only toys to correct for look elsewhere effect
  - If we are fitting background from data, this is circular do we do this?

An alternative is to use a **GP for the signal**

- Use a kernel that looks for an excess only in a localized excess in a window around mass m with width t (keeping length scale l for smoothness)

$$\Sigma(x, x') = Ae^{-\frac{1}{2}(x-x')^2/l^2} e^{-\frac{1}{2}((x-m)^2+(x'-m)^2)/t^2}, \quad (14)$$

- Now we have a signal shape, so we can do likelihood-ratio tests between signal and background

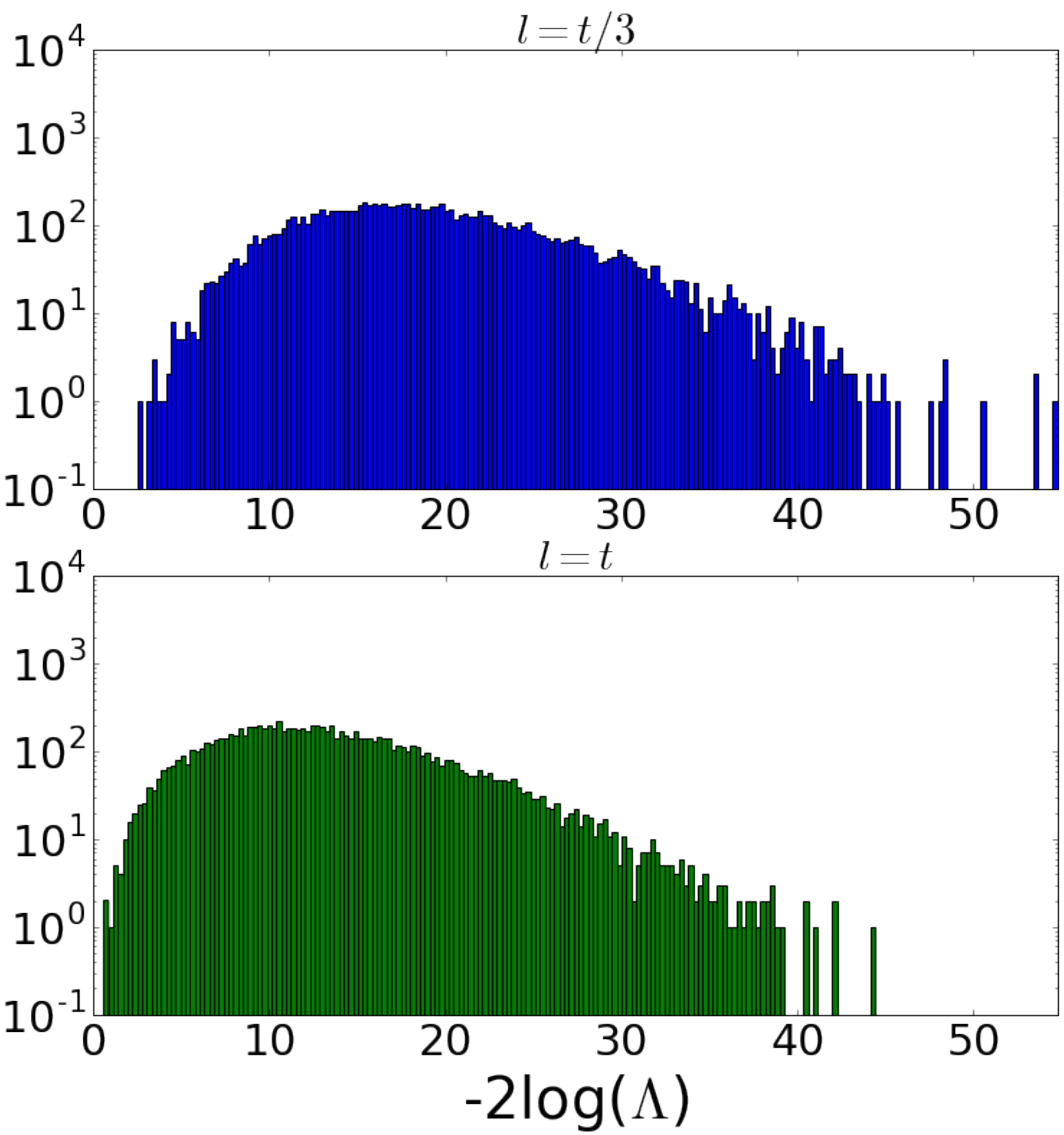
The issue now is that the signal has many free parameters, so these tests will have a look-elsewhere effect.

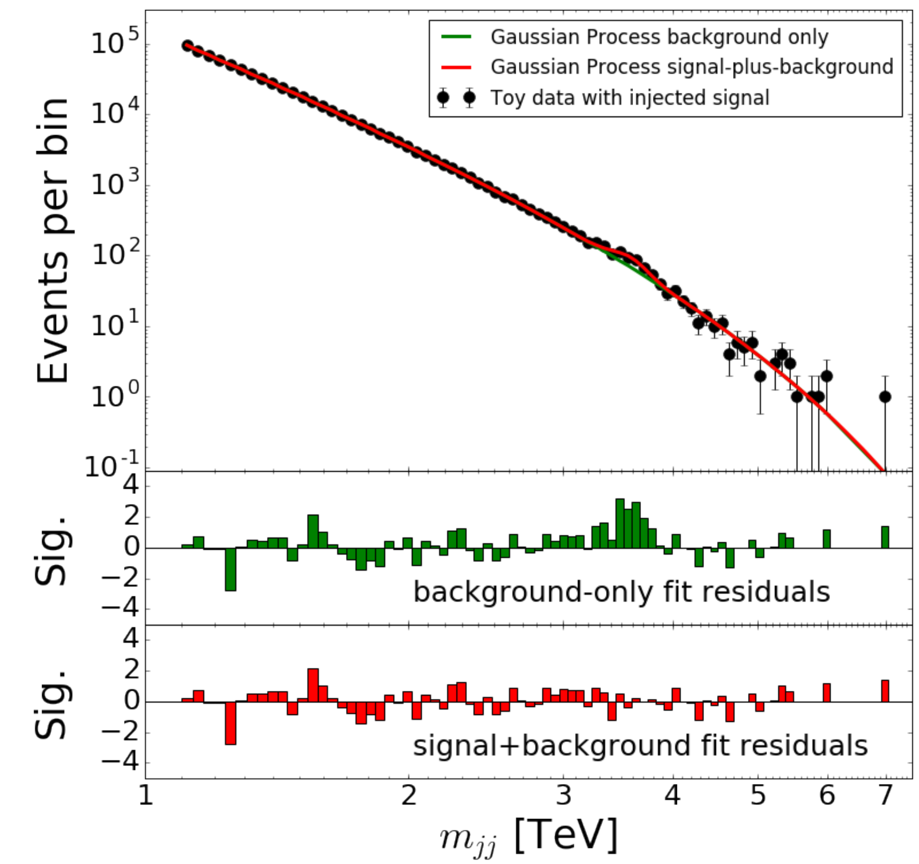
- this isn't a problem though, we still do background-only fits to get the "global p-value"

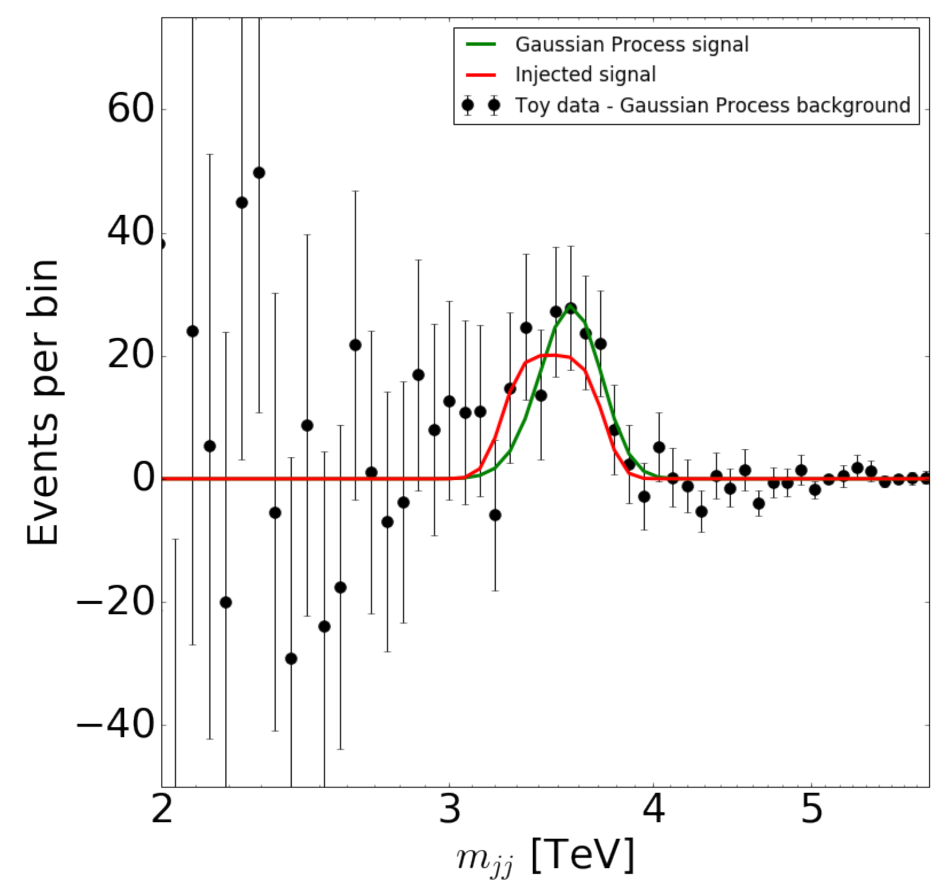
LOOK-ELSEWHERE EFFECT

The plot below shows 2logΛ(μ=0) for the background-only.Use this for global p-value.

(depends on kernel hyper parameters)







Software & Examples

You don't need to do this yourself, there's many Gaussian

Process packages that do this for you

- See github.com/mfrate28/ComparingGPpackages for a comparison of GP packages

Meghan worked on some tutorials that help with common HEP use cases

- https://github.com/mfrate28/GP_Tutorial

We are investigating a RooFit interface.

Physics-Aware Machine Learning

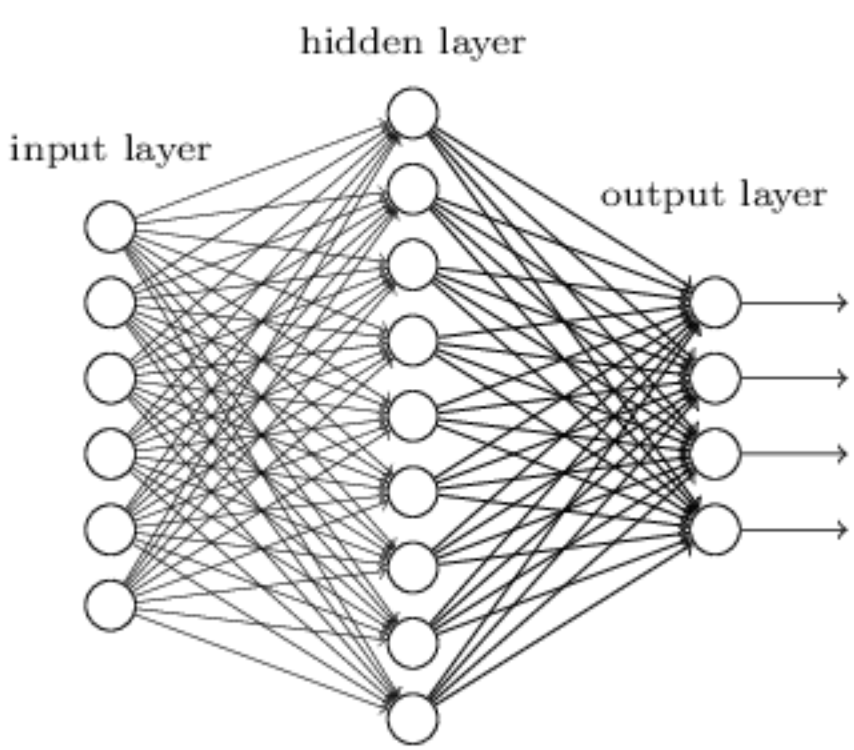
(choosing the variational family)

NN = A HIGHLY FLEXIBLE FAMILY OF FUNCTIONS

In calculus of variations, the optimization is over all functions: $\hat{s} = \underset{s}{\operatorname{argmin}}\, L[s]$

- In applied calculus of variations, we consider a highly flexible family of functions s_θ and optimize: i.e. $\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, L[s_\theta]$ $\quad \hat{s} \approx s_{\hat{\theta}}$
- Think of neural networks as a highly flexible family of functions
- Machine learning also includes non-convex optimization algorithms that are effective even with millions of parameters!

Shallow neural network



Deep neural network

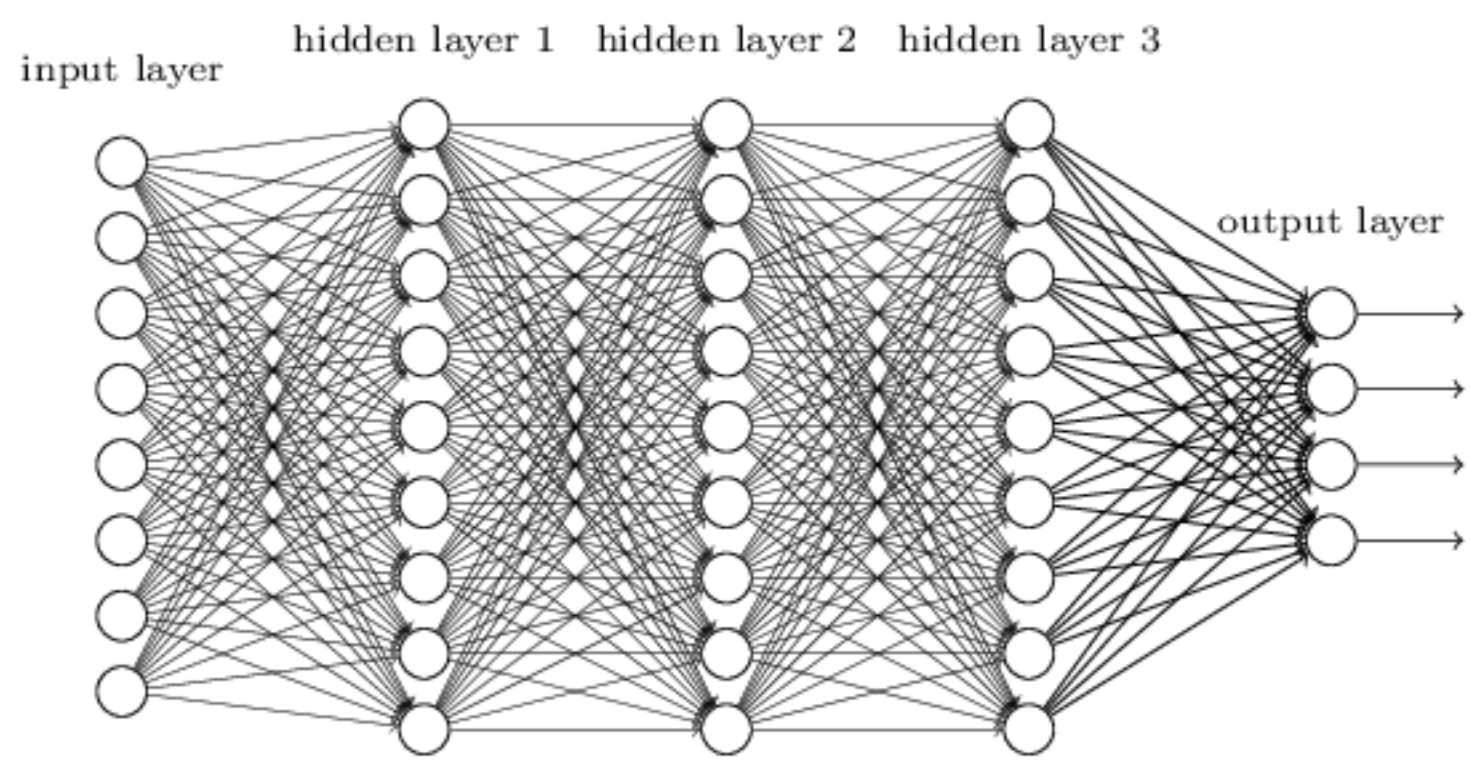


image credit: Michael Nielsen

Another major idea of deep learning: convolutional filters

- the world is compositional ⇒ hierarchical architecture
- images are translationally invariant ⇒ shared weights

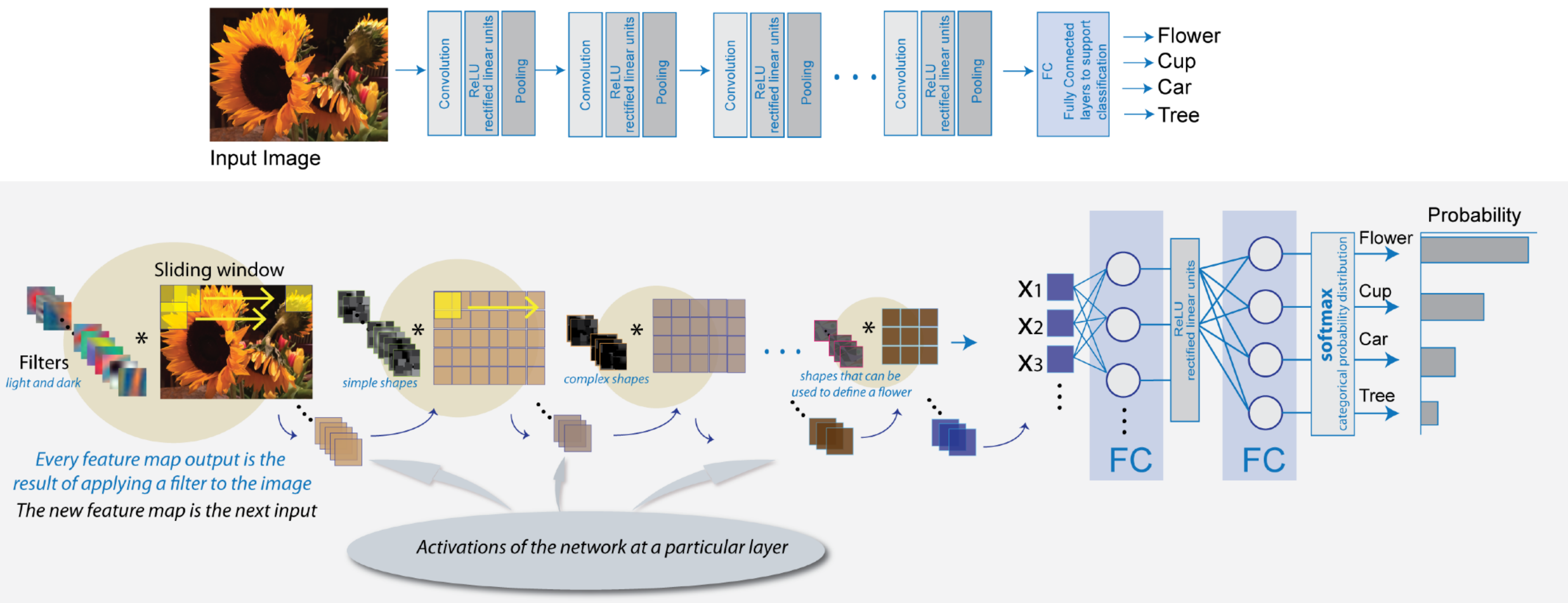
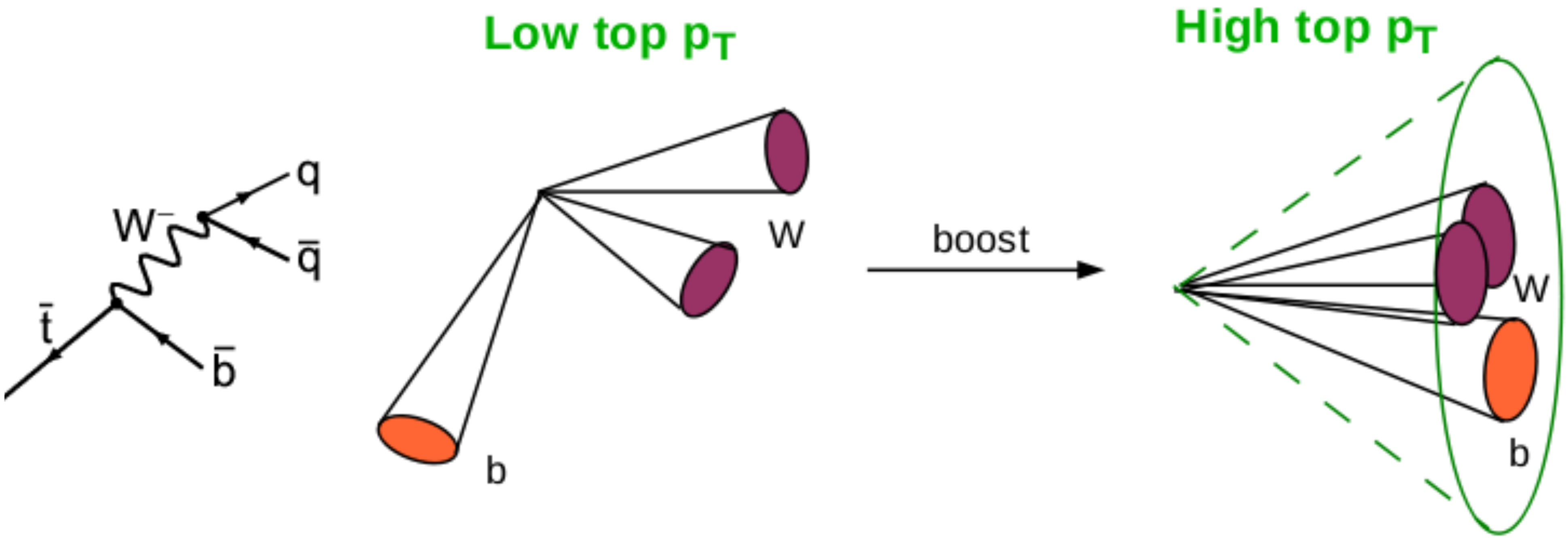


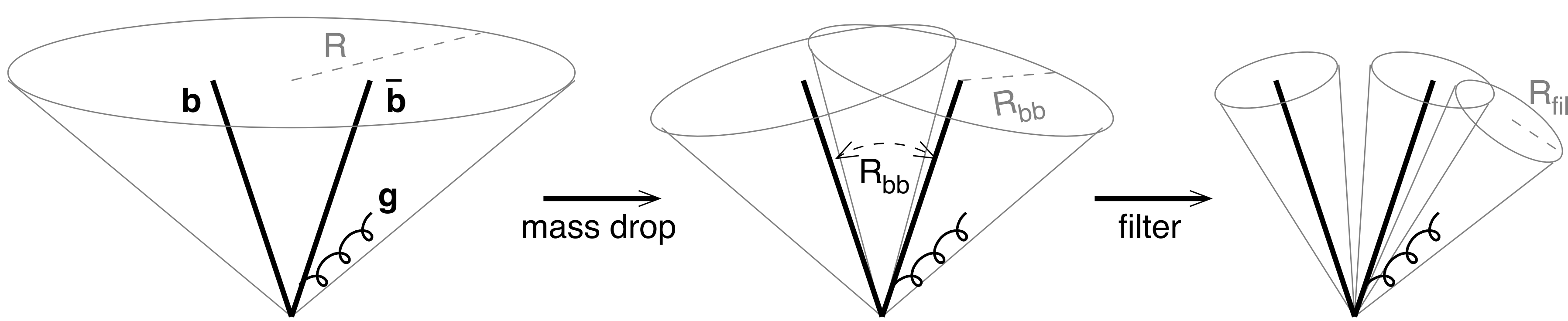
image credit: MathWorks

Many scenarios for physics Beyond the Standard Model include highly boosted W, Z, H bosons or top quarks



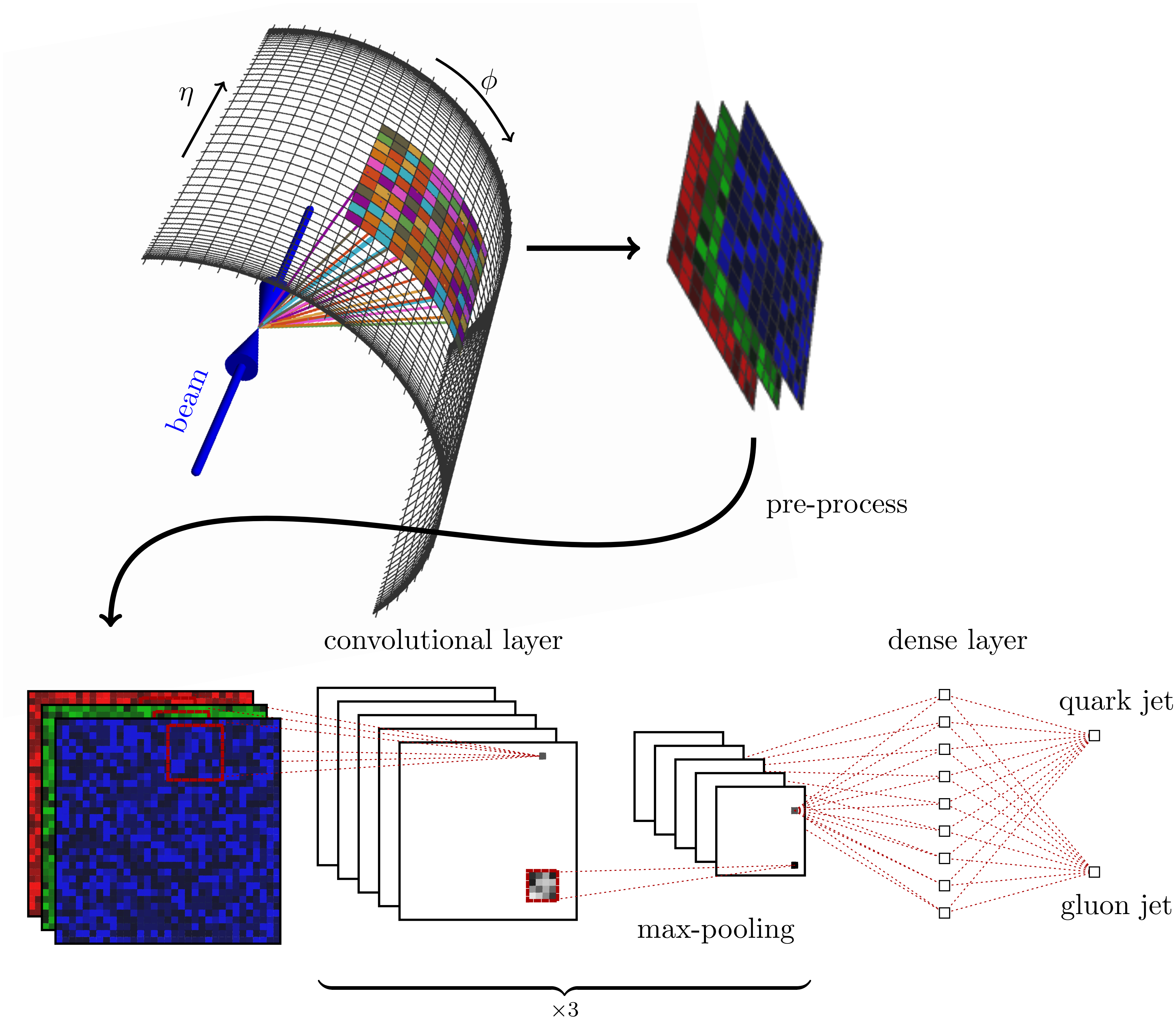
Identifying these rests on subtle substructure inside jets

- an enormous number of theoretical effort in developing observables and techniques to tag jets like this



JET IMAGES

image: Komiske, Metodiev, Schwartz arxiv:1612.01551
Oliveira, et. al arXiv:1511.05190
Whiteson, et al arXiv:1603.09349
Barnard, et al arXiv:1609.00607



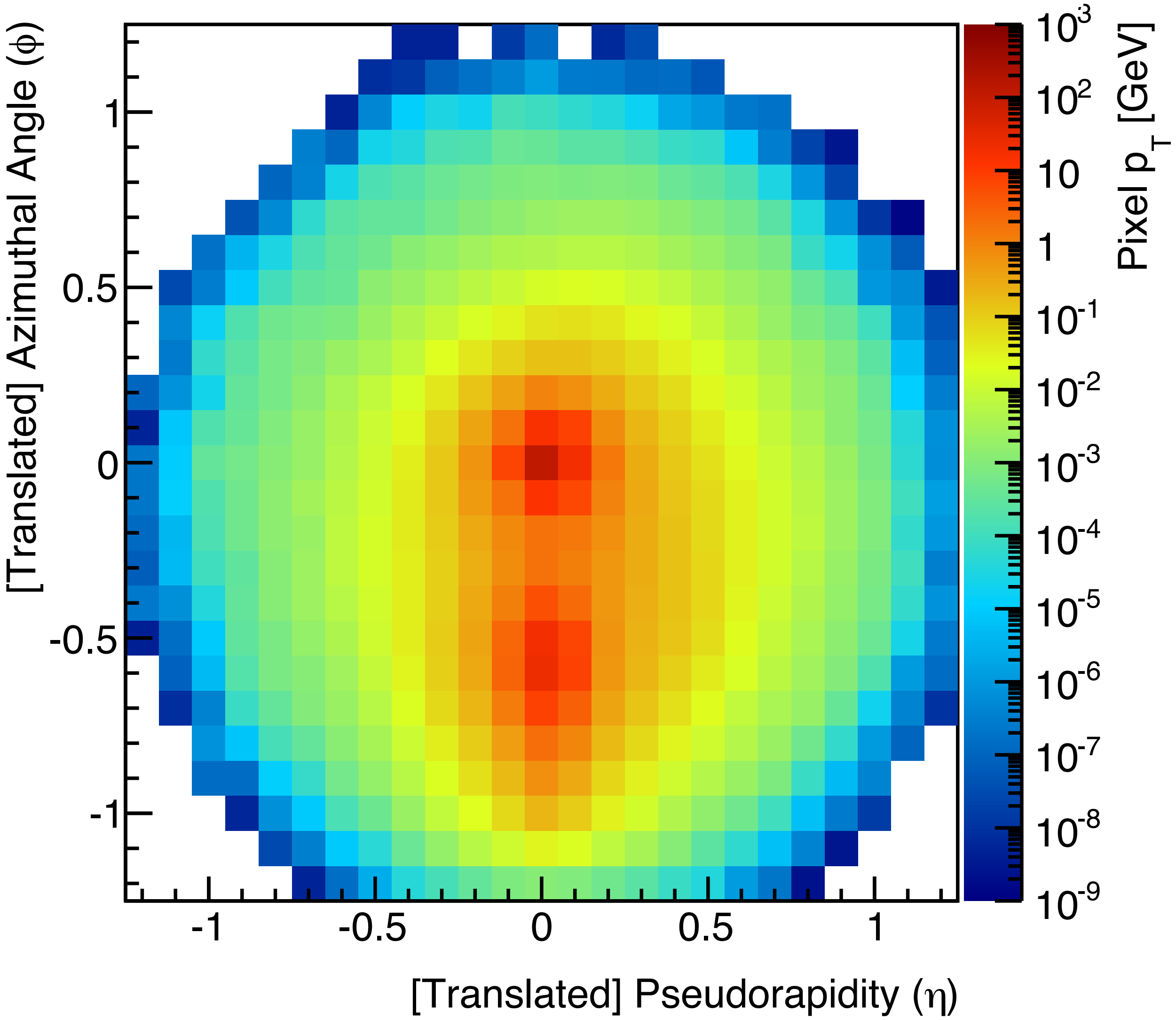
JET IMAGES

Oliveira, et. al arXiv:1511.05190
Whiteson, et al arXiv:1603.09349
Dawe, et al arXiv:1609.00607

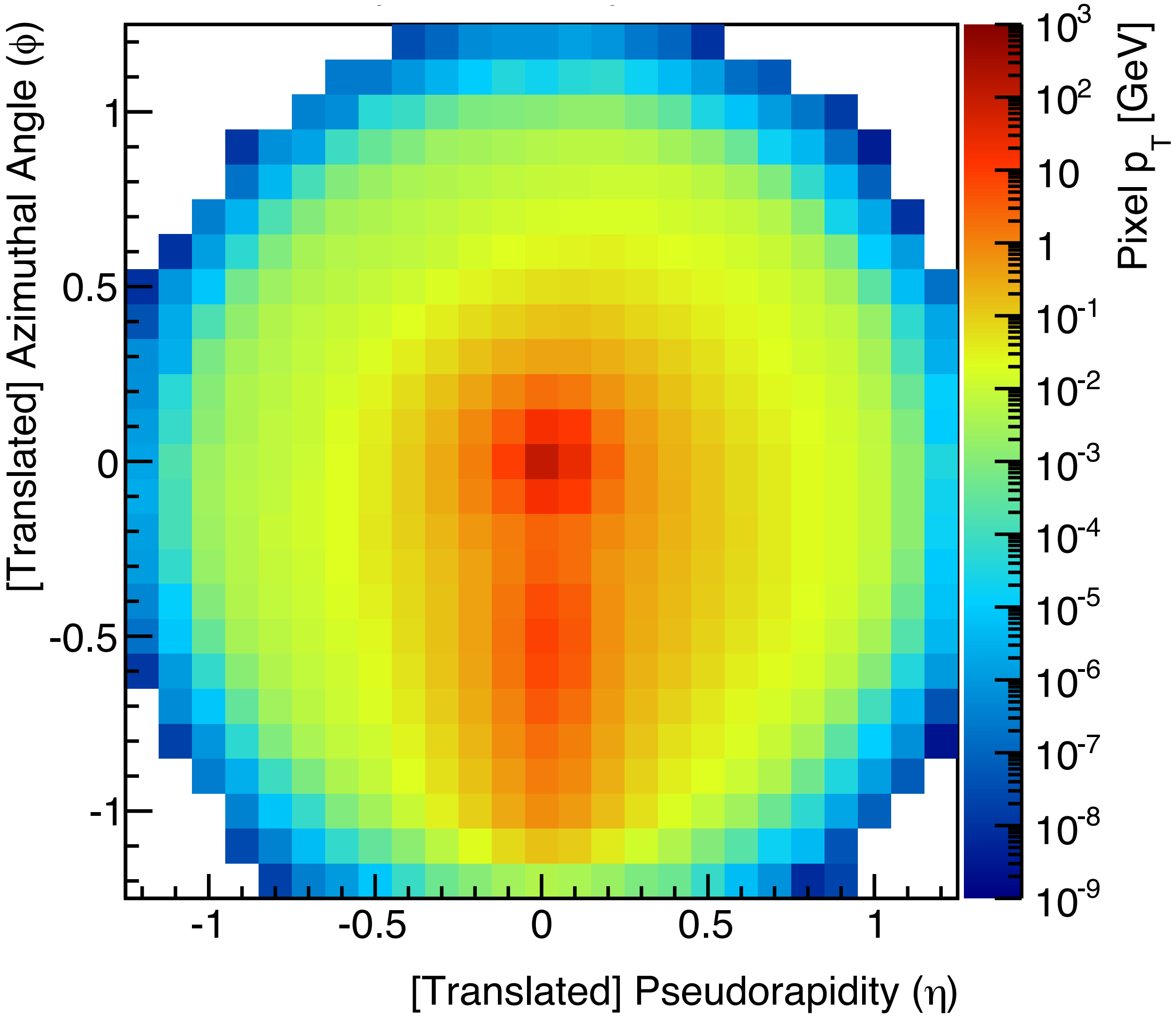
Apply deep learning algorithms to classify to "jet images"

- good results (based on fast simulation & idealized uniform calorimeter)
- preprocessed to mod out symmetries in the data
- discretization into images looses information

Average Boosted W Jet (y=1)



Average QCD Jet (y=0)



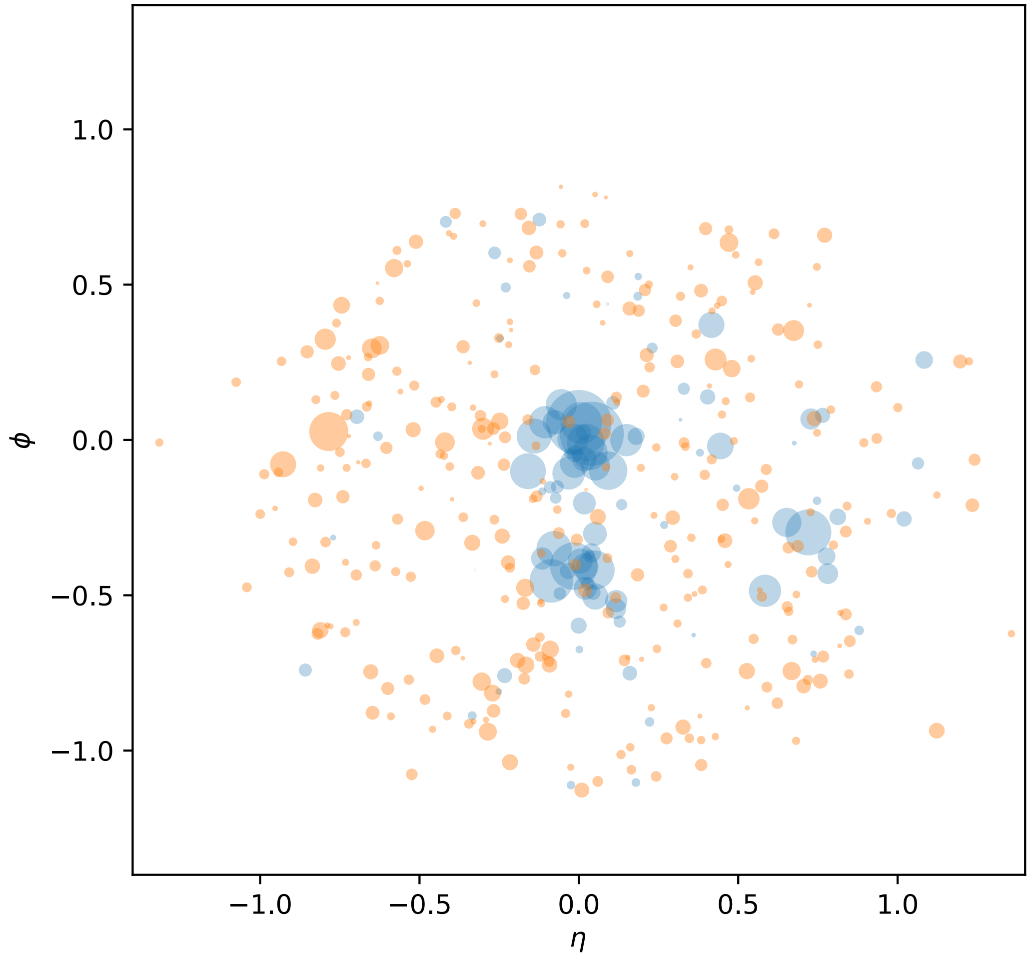
JETS AS A GRAPH

Isaac Henrion

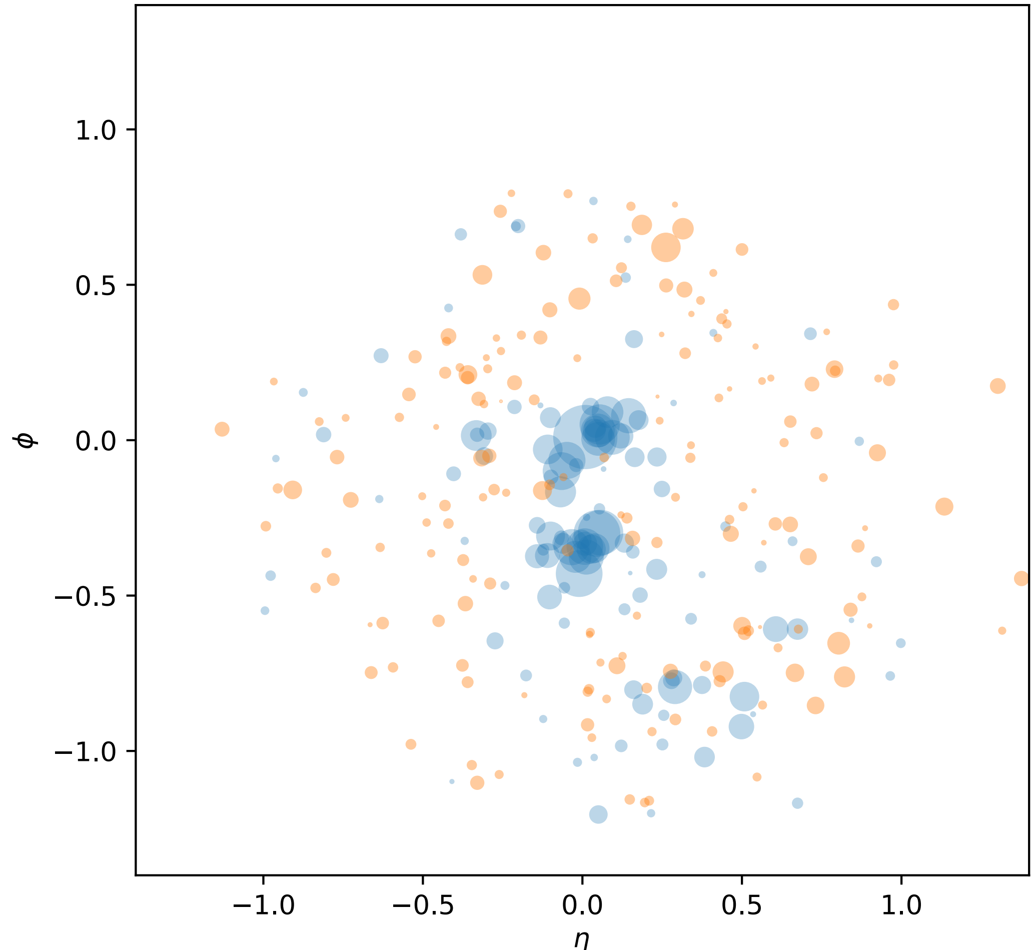
Using message passing neural networks over a fully connected graph on the particles

- Two approaches for adjacency matrix for edges
  - inject physics knowledge by using d_{ij} of jet algorithms
  - learn adjacency matrix and export new jet algorithm

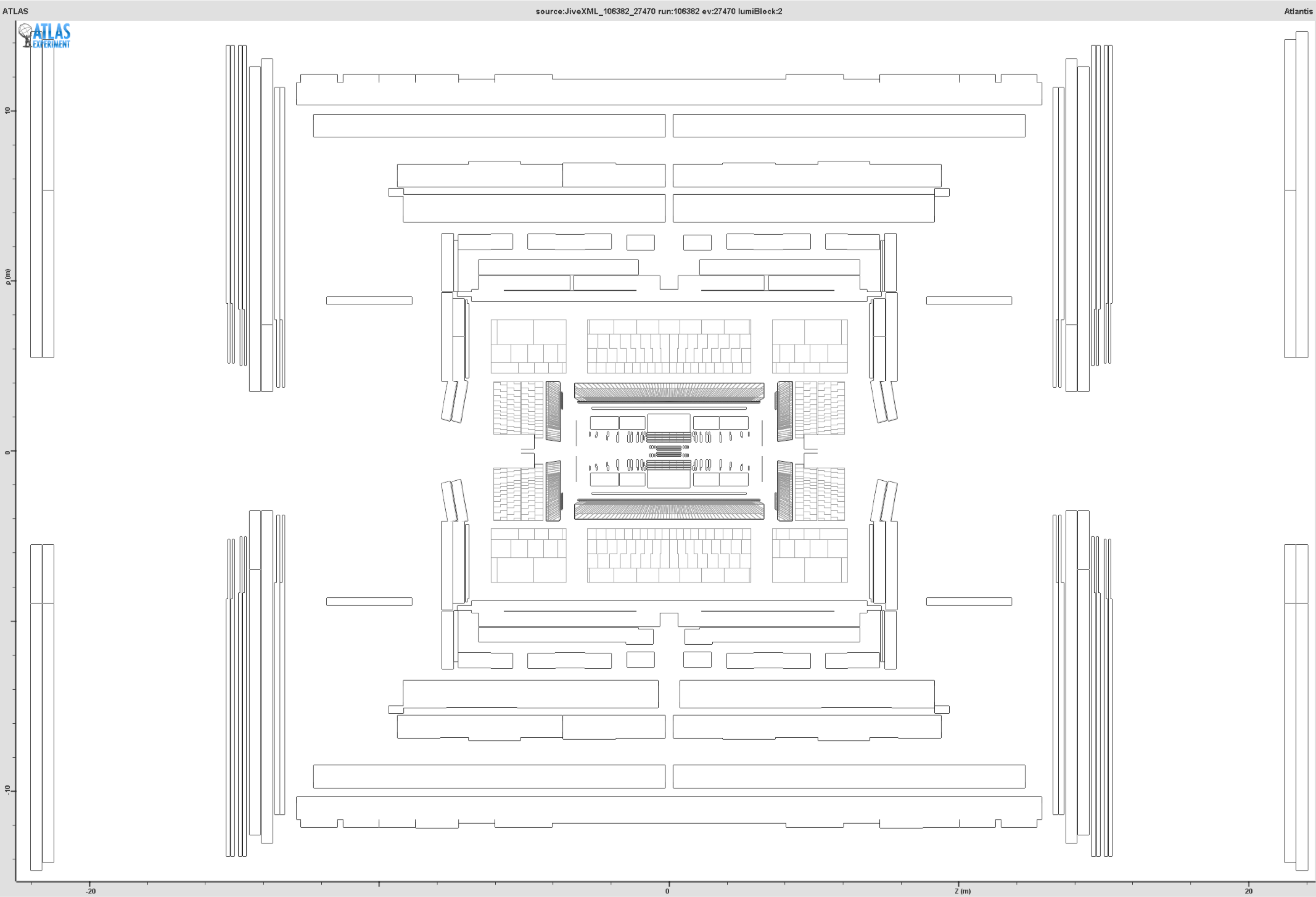
Example Boosted W Jet (y=1)



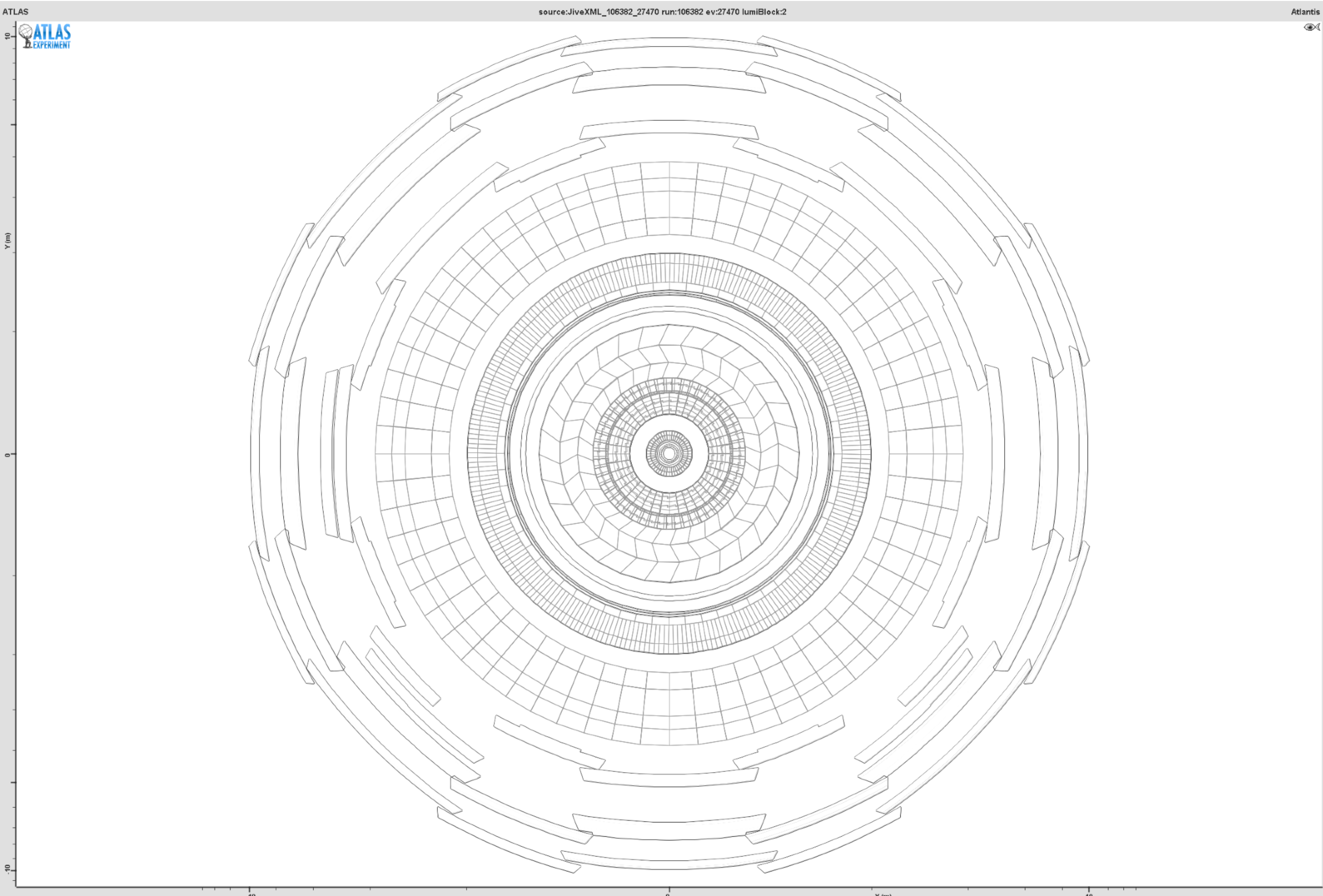
Example QCD Jet (y=0)



NON-UNIFORM GEOMETRY



NON-UNIFORM GEOMETRY



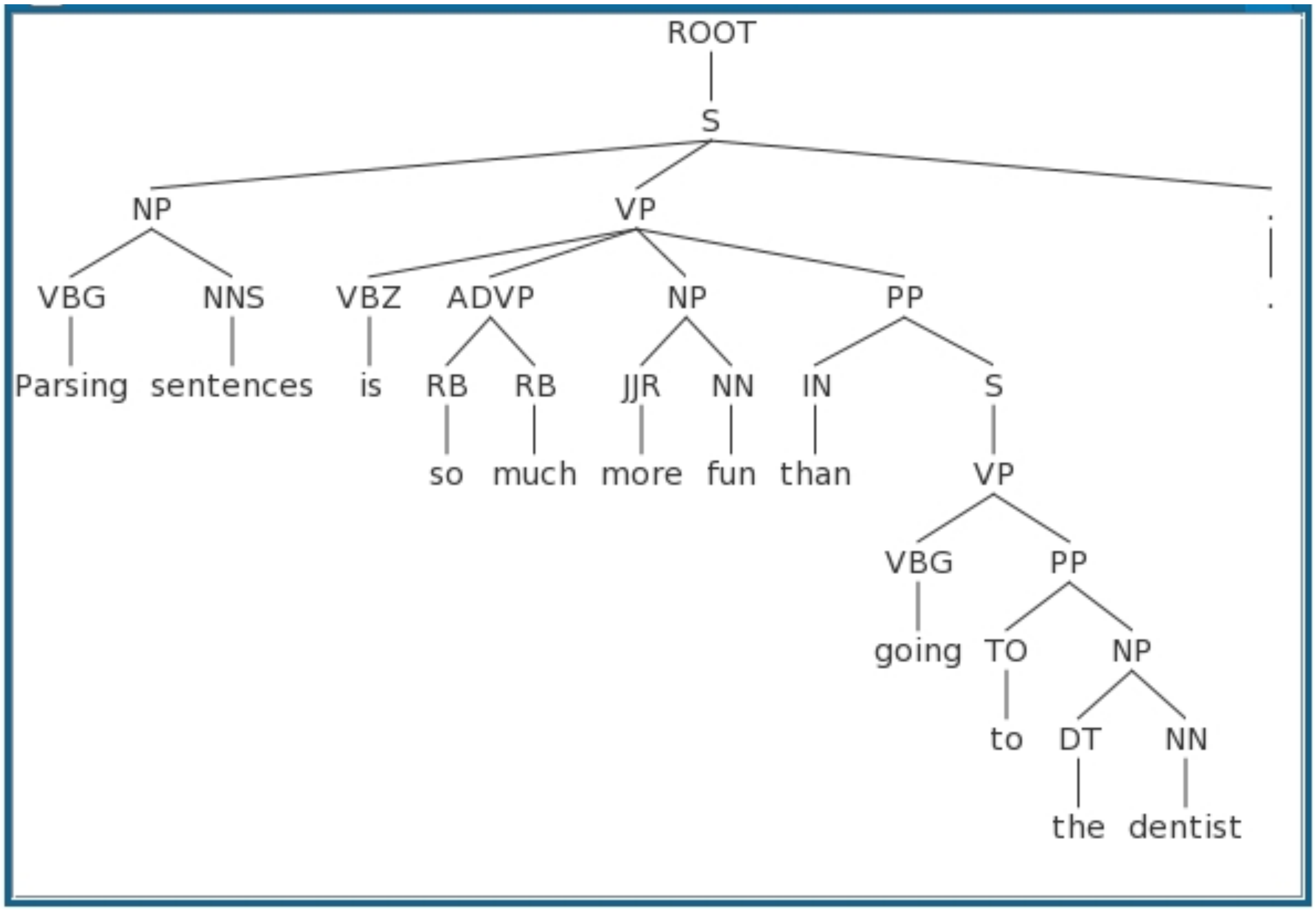
HOW CAN WE IMPROVE?

Image based approaches are doing well, but….

- would be nice to be able to work with a variable length input
  - avoid pre-processing into a regular-grid (eg. non-uniform calorimeters)
  - avoid representing empty pixels (sparse input)
- would be nice if classifier had nice theoretical properties
  - infrared & collinear safety, robustness to pileup, etc.
- would be nice to be more data efficient, most image-based networks use a LOT of training data.

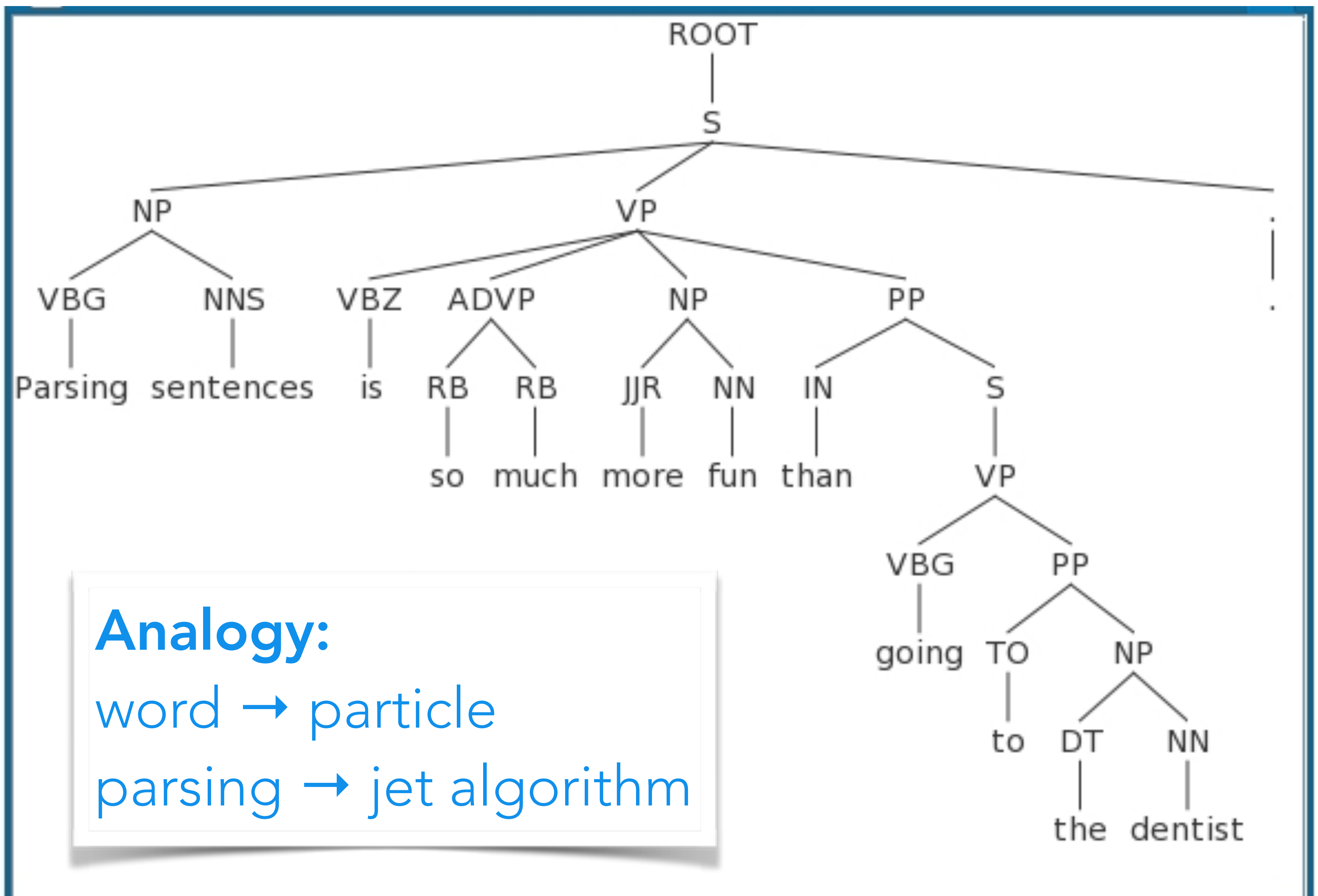
Recursive Neural Networks showing great performance for Natural Language Processing tasks

- neural network's topology given by parsing of sentence!



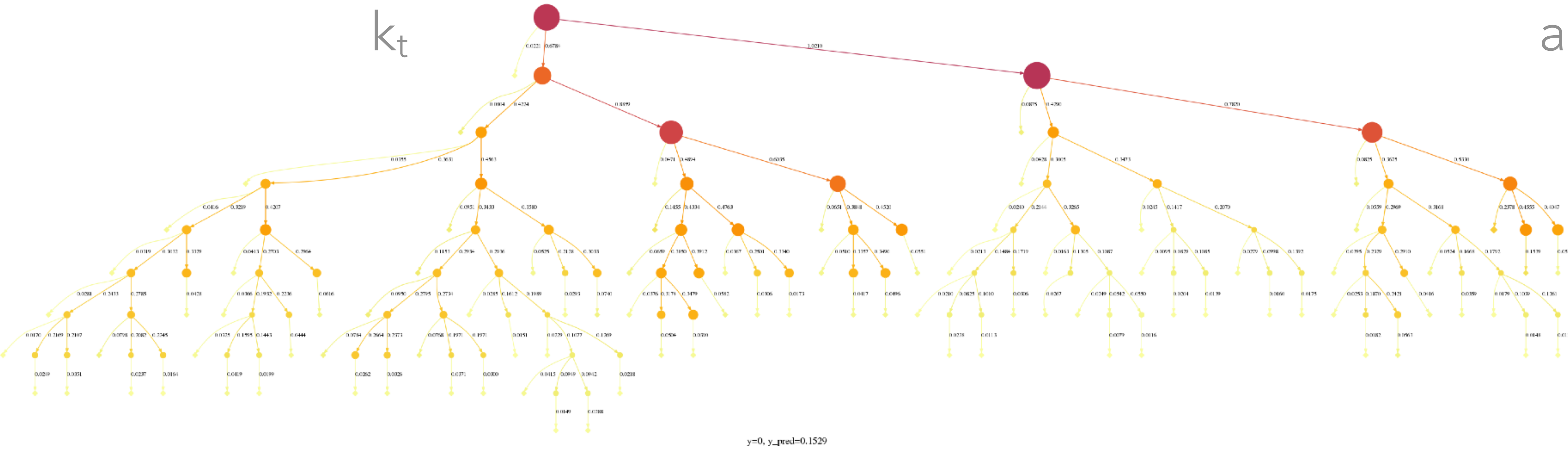
Recursive Neural Networks showing great performance for Natural Language Processing tasks

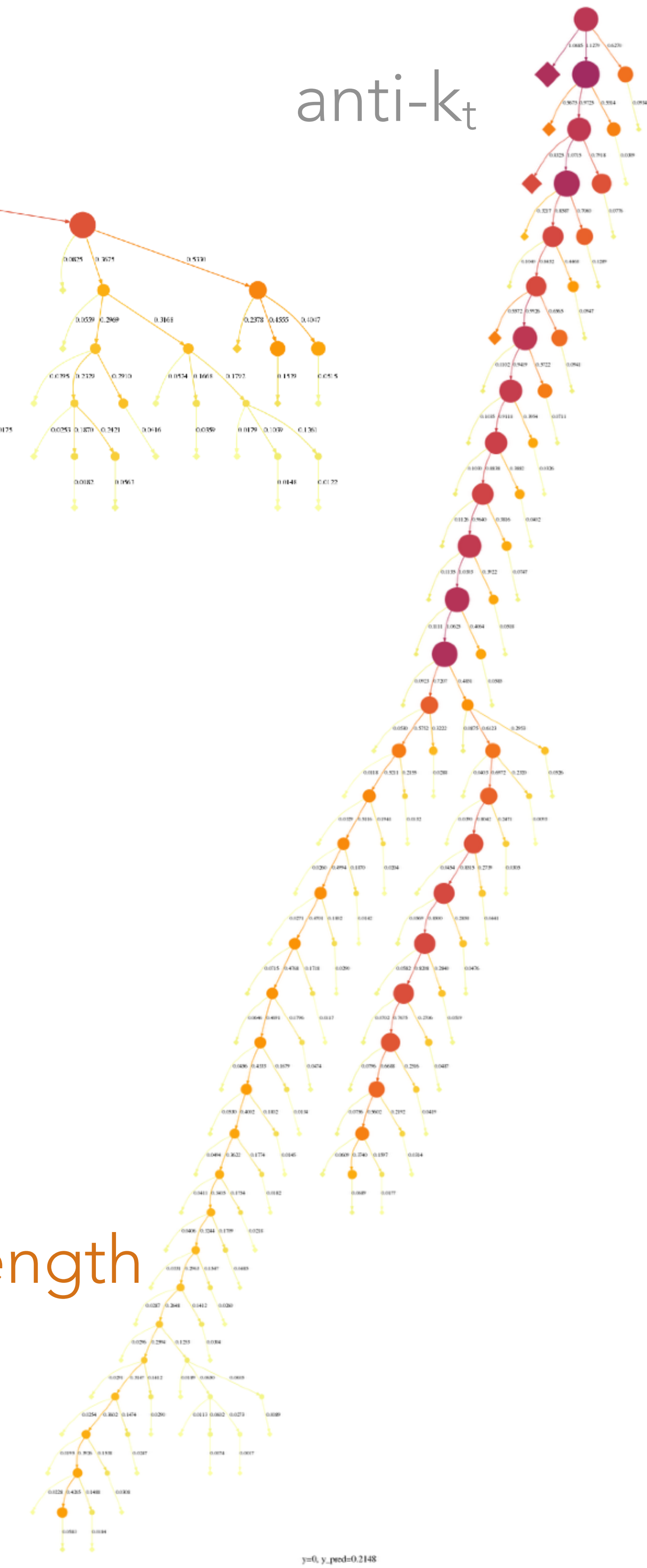
- neural network's topology given by parsing of sentence!



Analogy:
word → particle
parsing → jet algorithm

QCD-INSPIRED RECURSIVE NEURAL NETWORKS





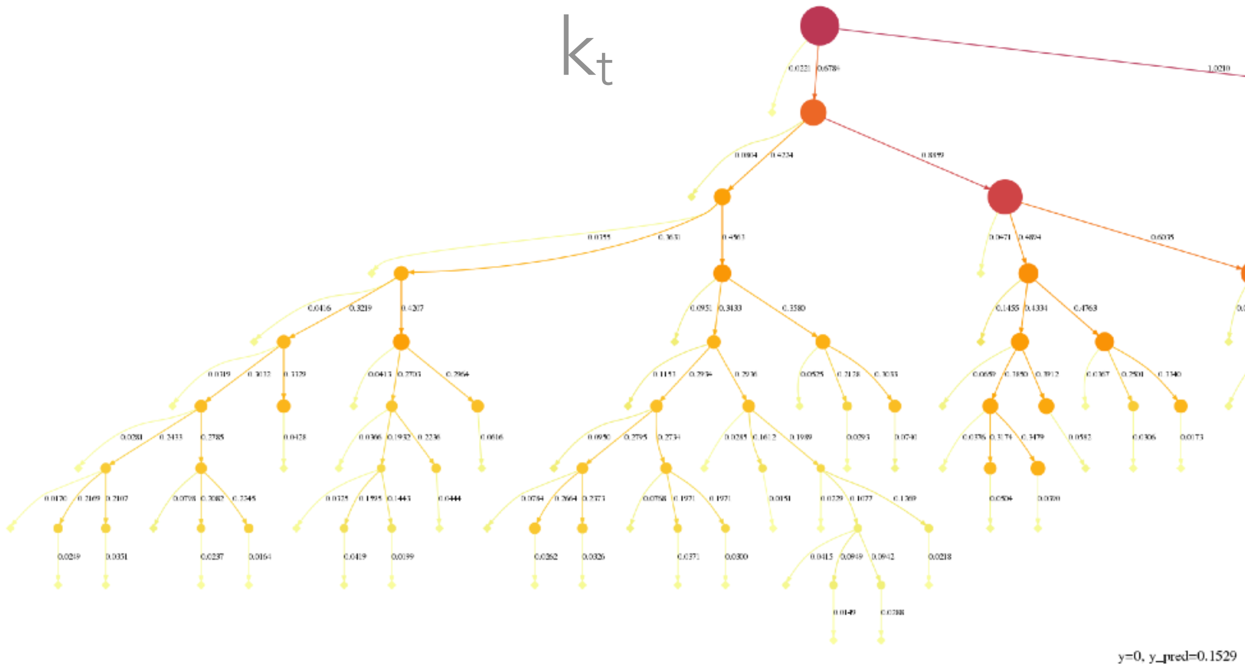
Work with Gilles Louppe, Kyunghyun Cho, Cyril Becot

- Use sequential recombination jet algorithms to provide network topology (**on a per-jet basis**)
- path towards ML models with good theoretical properties
- Top node of recursive network provides a fixed-length **embedding** of a jet that can be fed to a classifier

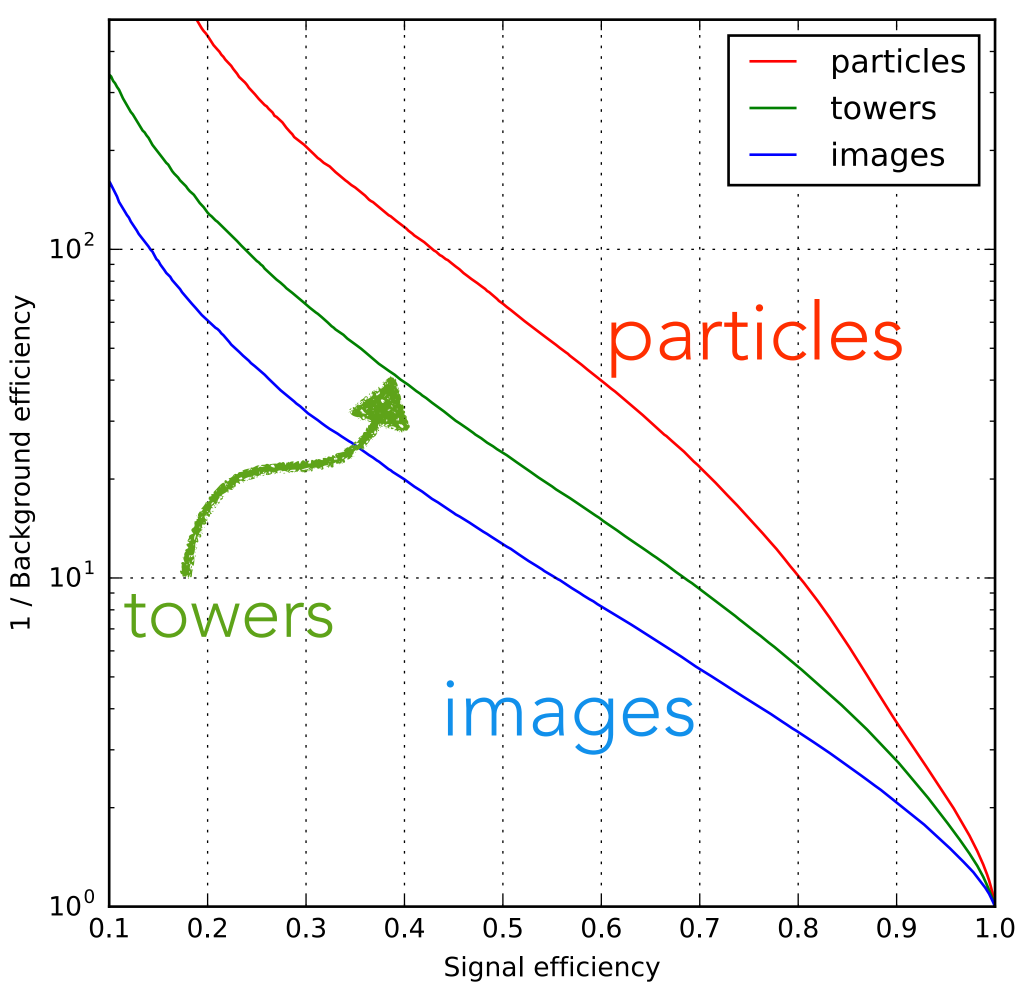
arXiv:1702.00748 & follow up work with Joan Bruna using graph conv nets

QCD-INSPIRED RECURSIVE NEURAL NETWORKS

k_t



anti-k_t

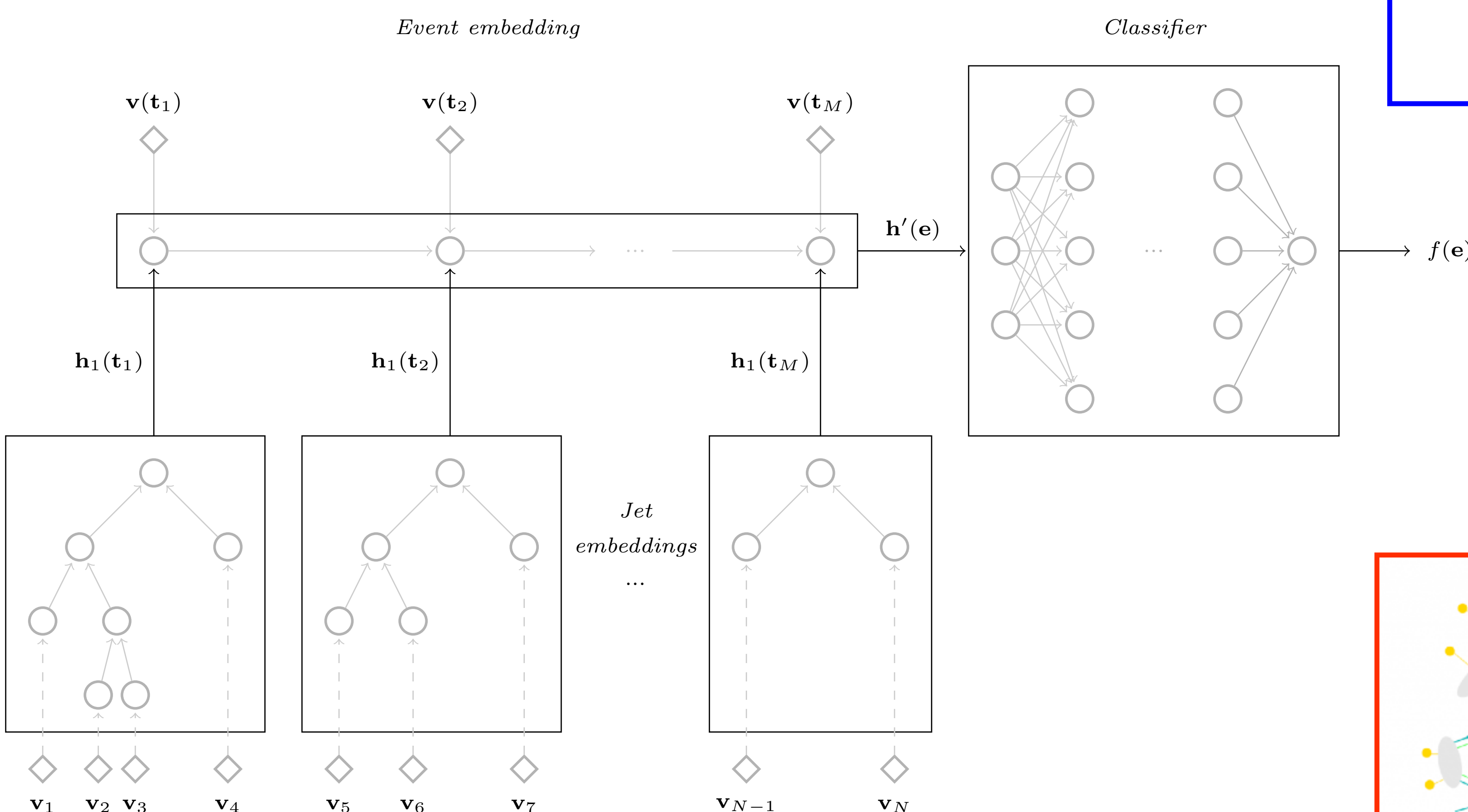


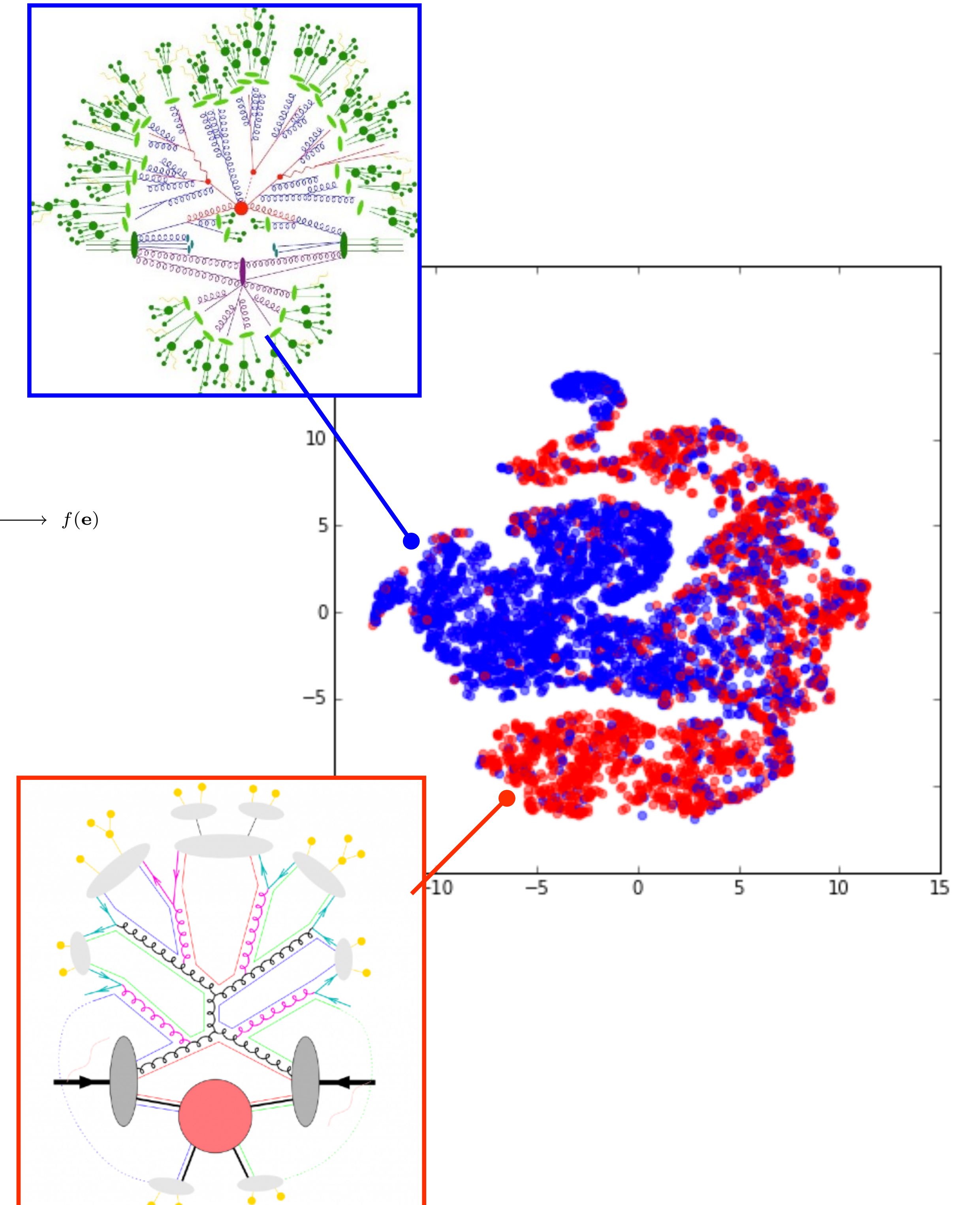
- W-jet tagging example using data from Dawe, et al arXiv:1609.00607
- down-sampling by projecting into images looses information
- RNN needs much less data to train!

HIERARCHICAL MODEL FOR THE ENTIRE EVENT

particle embedding → jet embedding → event embedding → classifier

It scales!



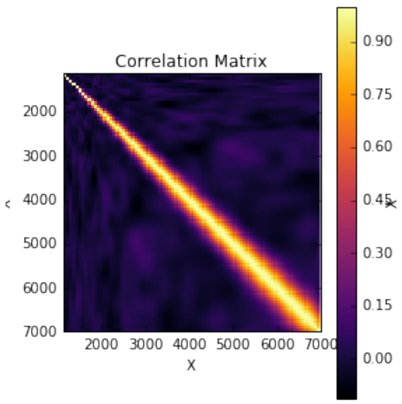


arXiv:1702.00748 & follow up work with Joan Bruna using graph conv nets

We can inject our knowledge of physics into the variational family

Physics-aware Gaussian Processes

arXiv:1709.05681

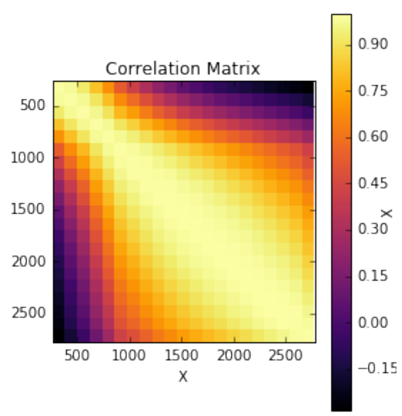


Final Kernel =

Poisson fluctuations

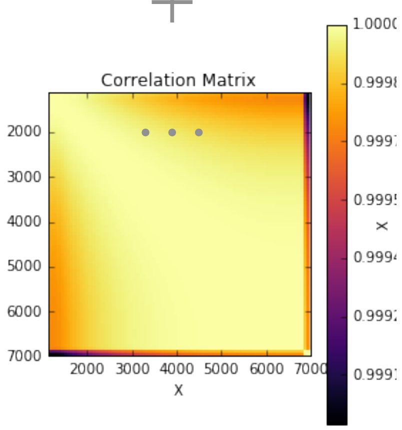
=

+ Mass Resolution



+ Parton Density Functions

+

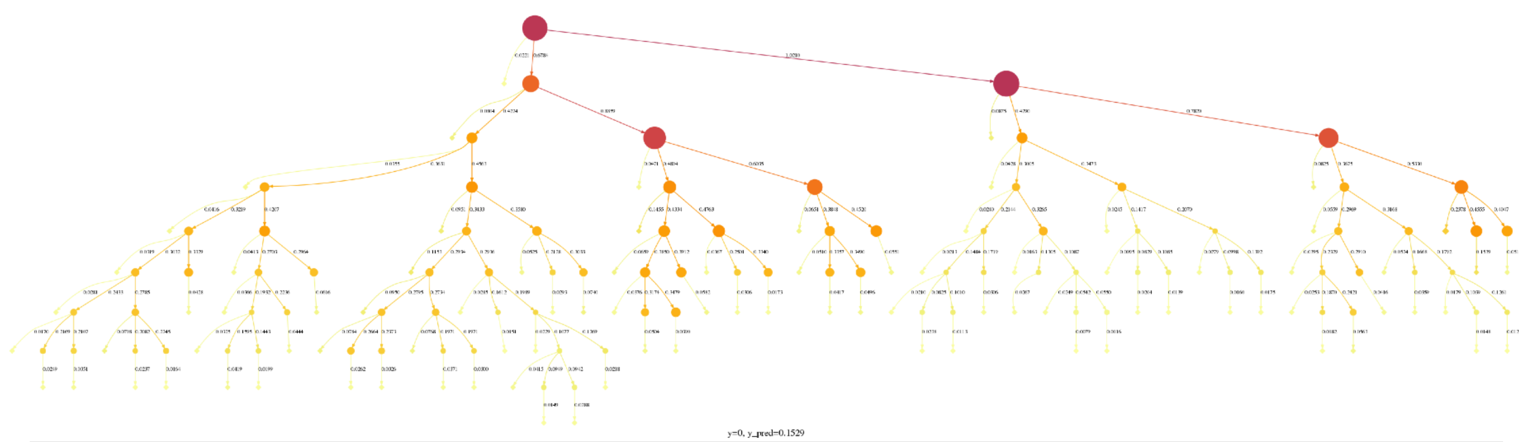


+ Jet Energy Scale

+

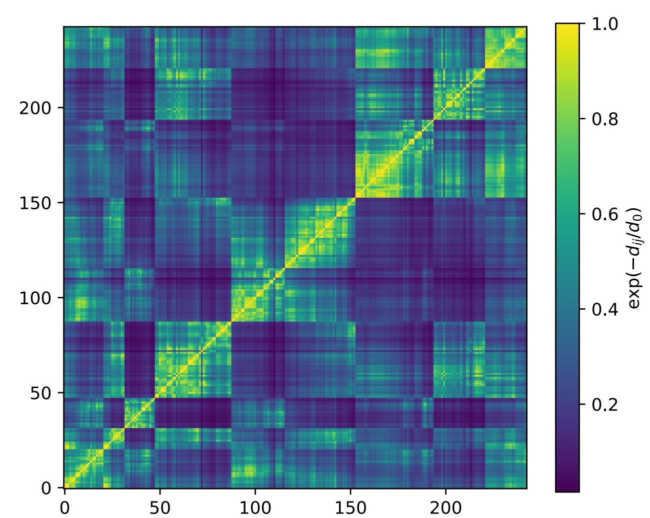
QCD-Aware recursive neural networks

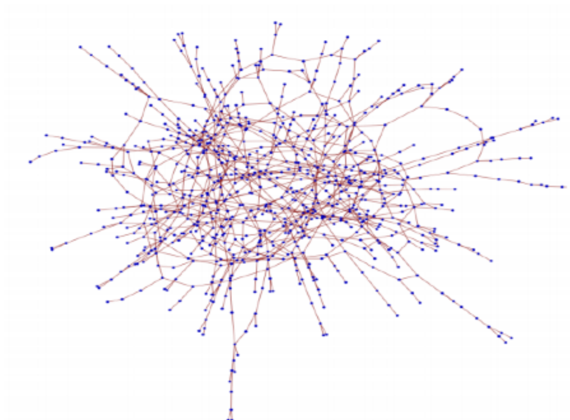
arXiv:1702.00748



QCD-Aware graph convolutional neural networks

NIPS2017 workshop





$$d^{\alpha}_{ii'} = \min(p^{2\alpha}_{ti}, p^{2\alpha}_{ti'}) \frac{\Delta R^2_{ii'}}{R^2}$$