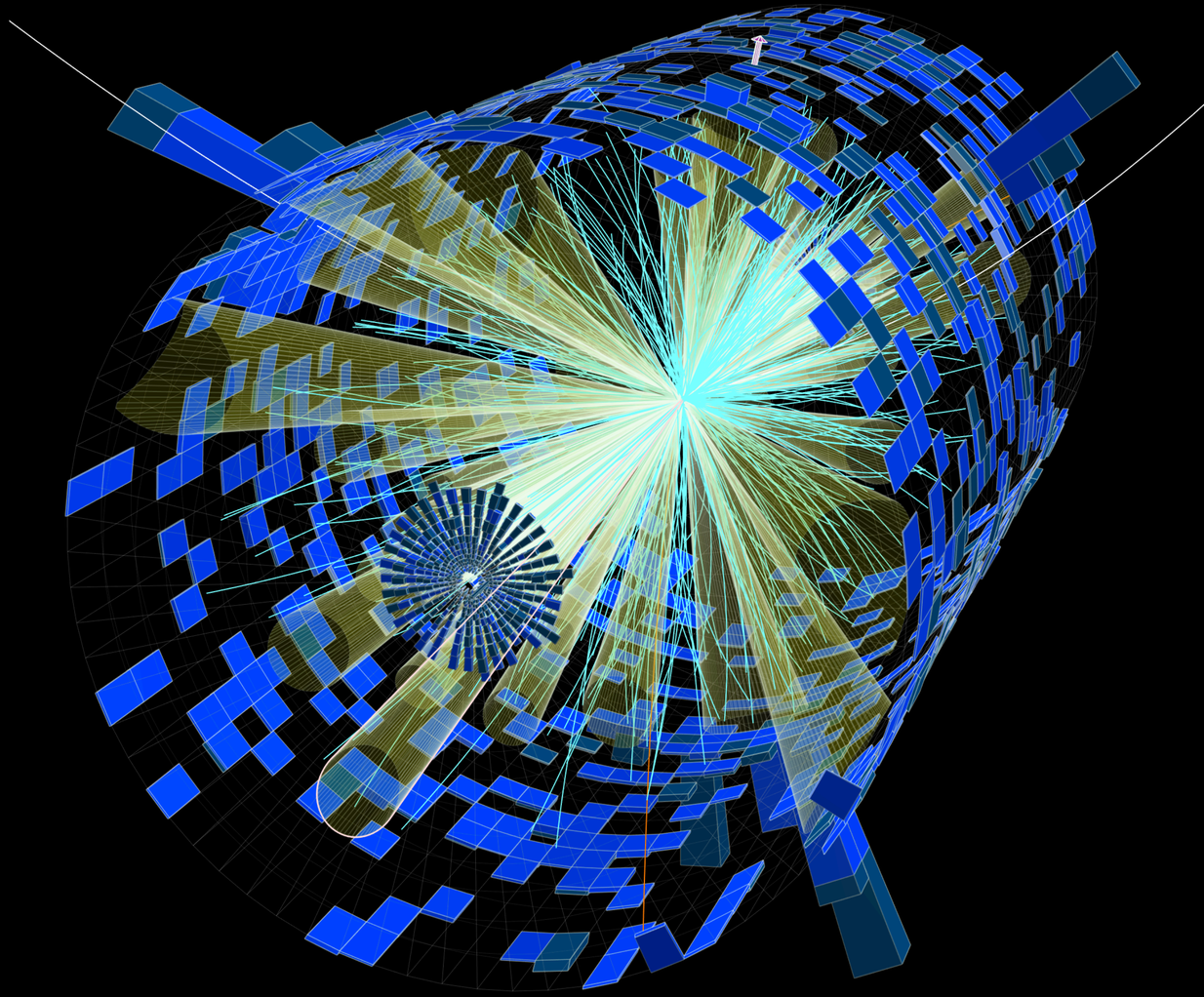
NYU Center for Data Science

Center for Cosmology and particle physics

~~THREE~~ FOUR APPROACHES TO SYSTEMATICS

WITH MACHINE LEARNING



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

Introduction

There is a lot of activity in utilizing machine learning in the analysis of particle physics data… duh.

I'll primarily focus on classification and regression tasks

- S**upervised learning**: function $f(x)$ to predict target y based on data x
- **Classification:** particle identification, signal vs. background discrimination
- **Regression:** estimate a particle's energy or momentum given detector readout

Classically physicists design some useful features / "observables" / summary statistics $h(x)$ motivated by domain knowledge and then design relatively simple functions $f(h) = f(h(x))$

- e.g. for classification "cuts" (a decision tree designed by hand)

Motivation

We want to take advantage of the power of machine learning, but we need to incorporate systematic uncertainties.

Two notions of "incorporate":

- **Don't be wrong:** view analysis chain as fixed and propagate systematic uncertainty through it.
  - e.g. control rate of type-I error in the presence of nuisance parameters
- **Try to be "optimal":** adjust the training of ML components so that the analysis is sensitive after accounting for systematics
  - e.g. minimize rate of type-II error / maximize power

Humans, heuristics, hubris, and humility

While ML can usually beat humans in classification tasks when neglecting systematic uncertainty, physicists are very good at designing features and "cuts" that are robust to systematic uncertainty.

- Physicists leverage various heuristics along the way to design "observables" / features / summary statistics that are robust or insensitive to the underlying sources of uncertainty
  - e.g. if you know some variable is poorly modeled, don't use it
  - e.g. if you know uncertainty leads to some (anti-)correlation, form the appropriate linear combination.

In the context of machine learning, we need to **formalize** what we mean by "incorporate systematic uncertainty" into an objective so that we can **operationalize** this

- e.g. a modified objective for optimization
- This is not so easy.
- Also, physicists usually are thinking about multiple downstream use cases

Formalism & notation

In classic supervised learning, the training data is $\{x_i, y_i\}_{i=1,\ldots,N}$

- e.g. binary classification where $y = 0$ for background, and $y = 1$ for signal
- For background: $y = 0 : x \sim p_{\text{bkg}}(x)$ or $p_{\text{bkg}}(x) = p(x \mid y = 0)$
- For signal: $y = 1 : x \sim p_{\text{sig}}(x)$ or $p_{\text{sig}}(x) = p(x \mid y = 1)$

Training data drawn from a joint distribution: $(x_i, y_i) \sim p(x, y) = p(x \mid y)p(y)$

- In many physics problems, physicists don't know the (prior) distribution $p(y)$ in data
  - often $p(y = 1)$ is the quantity of interest (is there a signal present or not)
- Physicists often try to generate balanced training data, even though in the real data the classes are typically very imbalanced.
- it can be useful to call $p(y)$ a "proposal" instead of a "prior" to avoid confusion with down-stream Bayesian inference and to remember physicists typically don't take prediction literally as $p(y \mid x)$

Formalism & notation

The primary notion of systematic uncertainty that physicists worry about is the lack of knowledge of the distribution $p(x \mid y)$ and how that uncertainty influences downstream inference

- e.g. training data often generated from a simulator $p_{\text{sim}}(x \mid y)$ and that simulator isn't perfect. Or training data comes from a control region in the data $p_{\text{control}}(x)$ that is assumed to be a good proxy for $p(x \mid y)$, but may not match for the region of the data being analyzed.

Typically we list what kinds of things might go wrong and parameterize their effect with nuisance parameters ν

- This gives us a generative model $p(x \mid y, \nu)$
- Uncertainty on the nuisance parameters is factorized from the effect they have, see Lukas Heinrich's talk. Frequentist: $p(a \mid \nu)$, Bayesian: $p(\nu \mid a) \propto p(a \mid \nu)p(\nu)$

Formalism & notation

Now we have a generative model $p(x \mid y, \nu)$ that can generate data x for each target y and systematic variation parameterized by ν

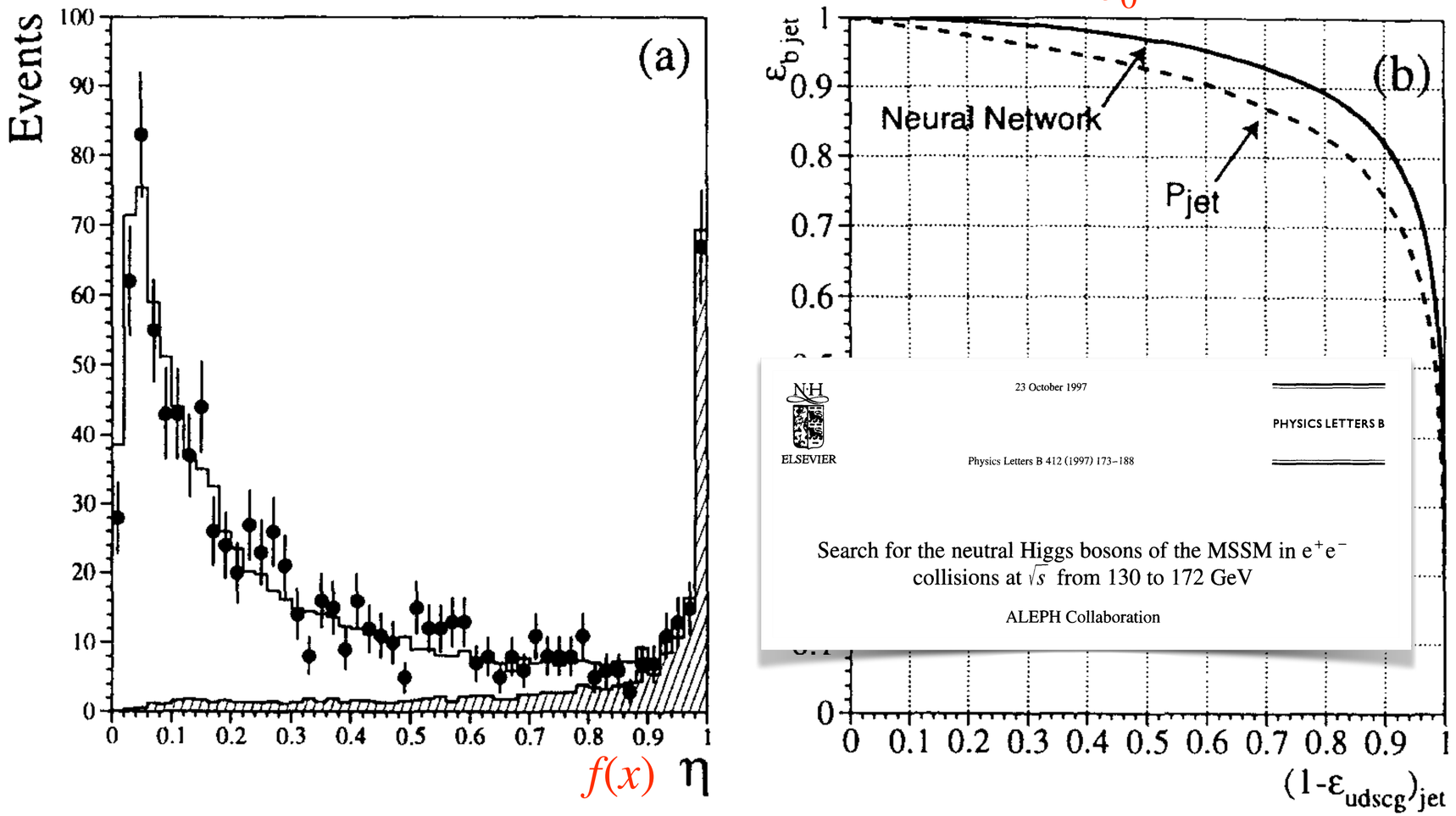
- e.g. y = "signal events" and ν = "jet reconstruction efficiency is off by 7%"
- e.g. y = "20 GeV electron" and ν = "energy calibration off by 5%"

Typically we have some best estimate or nominal settings ν_0

- Typical ML training is based on $(x_i, y_i) \sim p(x \mid y, \nu = \nu_0)p(y)$
- This leads to a trained model $f(x)$
- And then we think about the distribution of the output
  - For the nominal $p(f \mid y, \nu = \nu_0)$ and systematic variations $p(f \mid y, \nu)$

An example from the archives

$$\epsilon_{\text{b-jet}}(c) = \int_0^c p(f \mid y = 1)\mathrm{d}f$$



ELSEVIER

23 October 1997

PHYSICS LETTERS B

Physics Letters B 412 (1997) 173–188

Search for the neutral Higgs bosons of the MSSM in e^+e^- collisions at $\sqrt{s}$ from 130 to 172 GeV

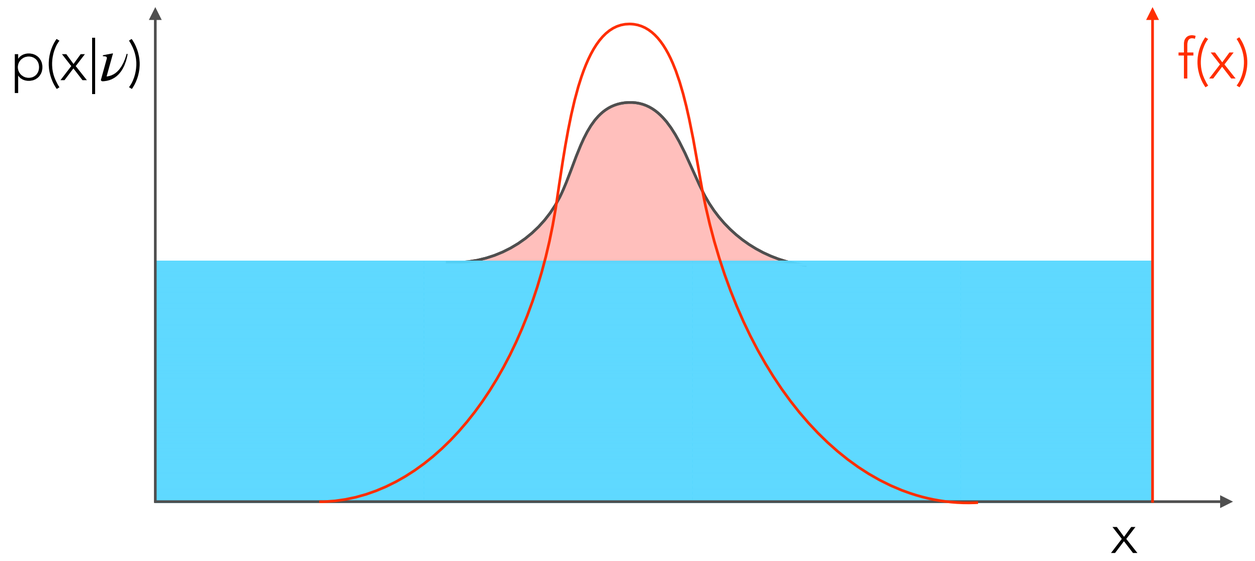
ALEPH Collaboration

Fig. 2. (a) The output η of the neural network b tag for radiative returns to the Z for 161 GeV $q\bar{q}$ Monte Carlo (histogram) compared to the data at 161 GeV (points). The shaded region shows the contribution from generated b-jets. (b) The performance of the neural network b tag (solid line) for Monte Carlo events, presented in terms of the efficiency for identifying b-jets versus the efficiency for rejecting light quark jets. The performance of the single most powerful b tagging input variable to the neural network is shown for comparison (dashed curve).

Fixed classifier is not optimal

Imagine a simple example of bump on flat background

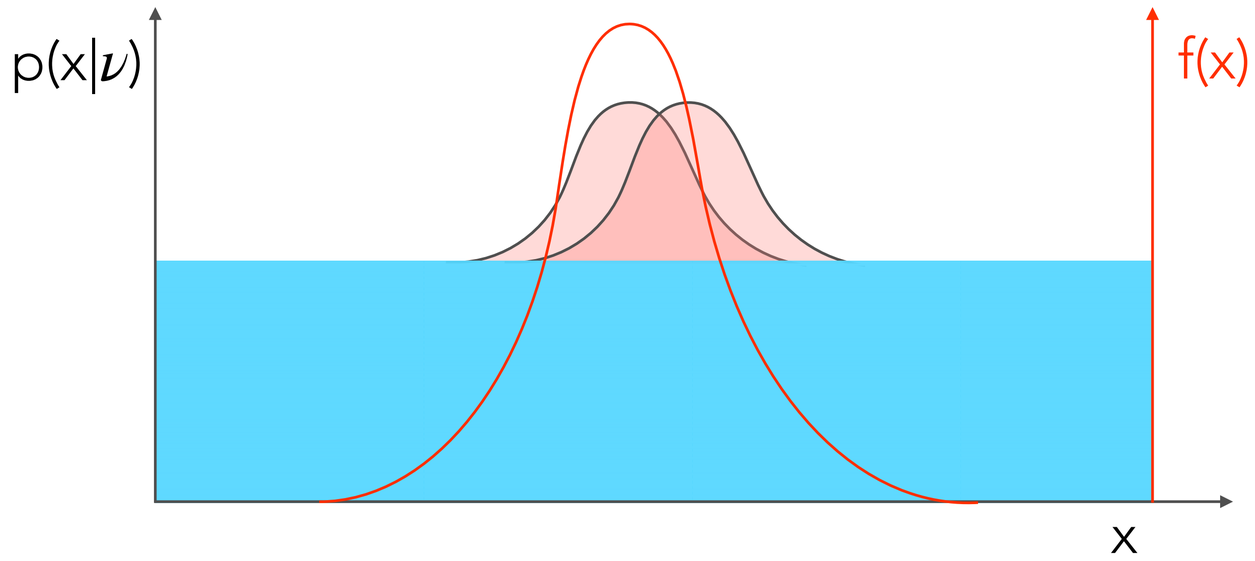
- train on samples with $\nu = \nu_0$ to obtain fixed classifier $f(x)$
- uncertainty in ν modifies location and width of peak
- the classifier not optimal for $\nu \neq \nu_0$, but we can propagate uncertainty



Fixed classifier is not optimal

Imagine a simple example of bump on flat background

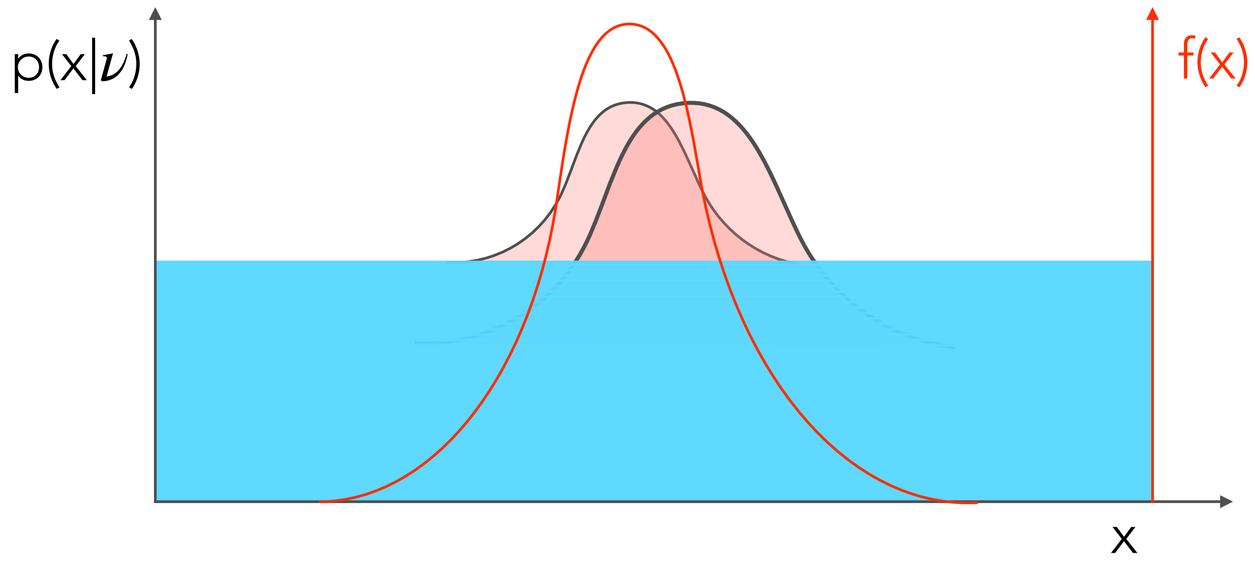
- train on samples with $\nu = \nu_0$ to obtain fixed classifier $f(x)$
- uncertainty in ν modifies location and width of peak
- the classifier not optimal for $\nu \neq \nu_0$, but we can propagate uncertainty



Fixed classifier is not optimal

Imagine a simple example of bump on flat background

- train on samples with $\nu = \nu_0$ to obtain fixed classifier $f(x)$
- uncertainty in ν modifies location and width of peak
- the classifier not optimal for $\nu \neq \nu_0$, but we can propagate uncertainty

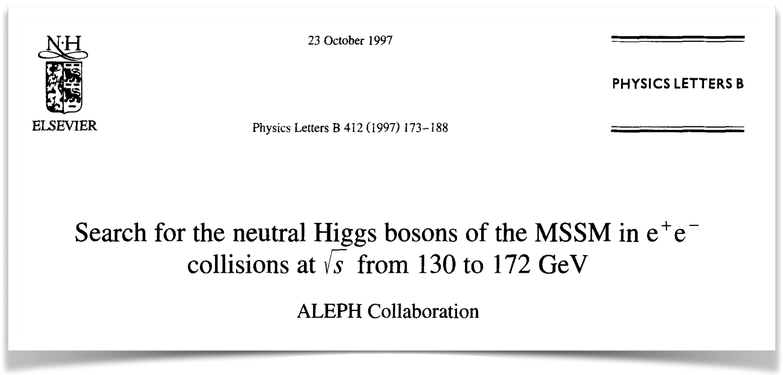


An example from the archives

Here is an example comparing the nominal efficiency (calculated from $p(f \mid y, \nu = \nu_0)$) to a variational sample $p(f \mid y, \nu)$ taken from data.

$$\epsilon_{\text{b-jet}}(c, \nu) = \int_0^c p(f \mid y = 1, \nu) \mathrm{d}f$$

$$\frac{\epsilon_{\text{b-jet}}(c, \nu) - \epsilon_{\text{b-jet}}(c, \nu = \nu_0)}{\epsilon_{\text{b-jet}}(c, \nu = \nu_9)}$$



NH
ELSEVIER

23 October 1997

PHYSICS LETTERS B

Physics Letters B 412 (1997) 173–188

Search for the neutral Higgs bosons of the MSSM in e^+e^- collisions at $\sqrt{s}$ from 130 to 172 GeV

ALEPH Collaboration

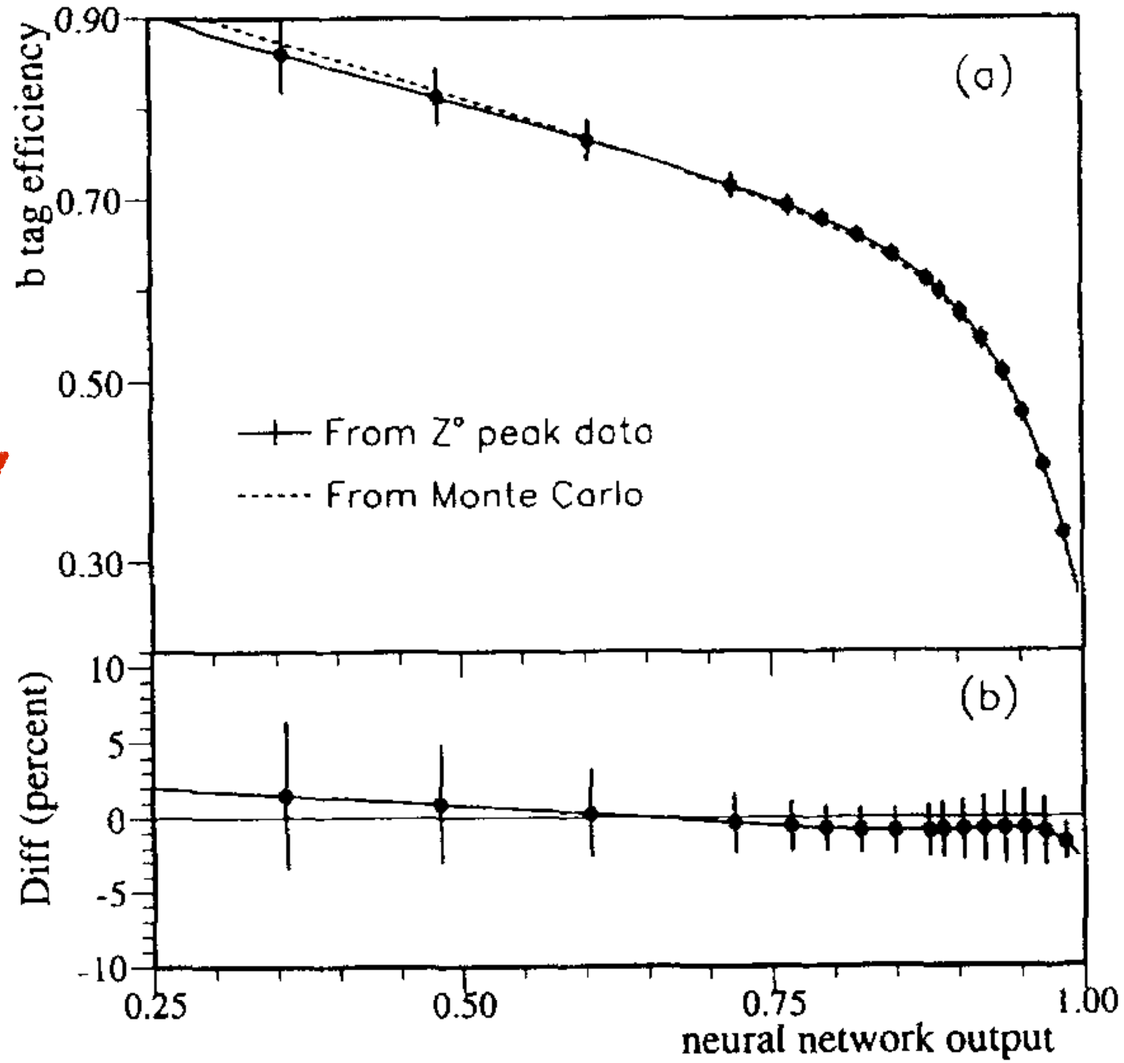


Fig. 3. Comparison of the neural network b tag efficiency for a single jet in Z peak data and Monte Carlo, after the smearing correction, as a function of the cut on the jet neural network output: (a) absolute tag efficiency; (b) difference between data and Monte Carlo.

Propagating uncertainty to downstream inference

After characterizing how the efficiency depends on the "working point" c and the nuisance parameters ν the experiments use this information in the downstream statistical analysis.

For example: $\epsilon_{\text{sig}}(\nu) = \int_0^c p(f \,|\, y = 1, \nu) \mathrm{d}f$ $\epsilon_{\text{bkg}}(\nu) = \int_0^c p(f \,|\, y = 0, \nu) \mathrm{d}f$

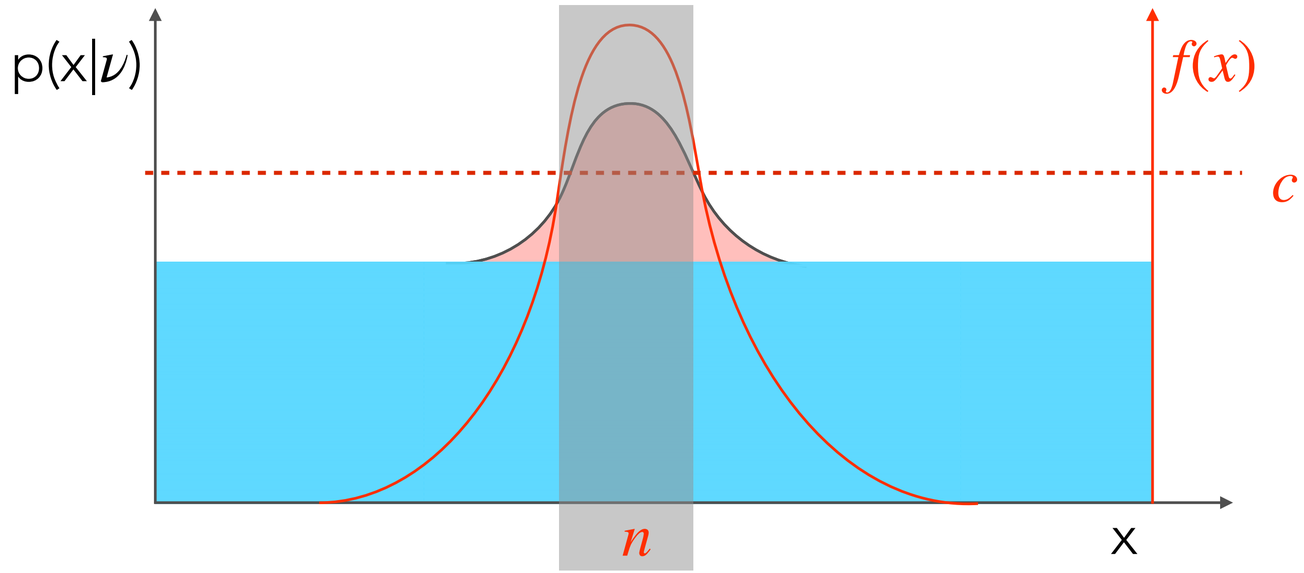
And later one might form a statistical model for the number of events n that have $f(x) > c$, where s, b are some nominal number of signal and background events that would be produced, μ is the parameter of interest, ν is the nuisance parameter, and $p(a \,|\, \nu)$ is the likelihood associated to some auxiliary measurement used to estimate ν $p(n, a \,|\, \mu, \nu) = \text{Pois}(n \,|\, \mu \epsilon_{\text{sig}}(\nu) s + \epsilon_{\text{bkg}}(\nu) b) p(a \,|\, \nu)$

A visual example

…and later one might form a statistical model for the number of events n that have $f(x) > c$

$$\epsilon_{\text{sig}}(\nu) = \int_0^c p(f \,|\, y = 1, \nu) \mathrm{d}f \qquad \epsilon_{\text{bkg}}(\nu) = \int_0^c p(f \,|\, y = 0, \nu) \mathrm{d}f$$

$$p(n, a \,|\, \mu, \nu) = \text{Pois}(n \,|\, \mu \epsilon_{\text{sig}}(\nu) s + \epsilon_{\text{bkg}}(\nu) b) p(a \,|\, \nu)$$

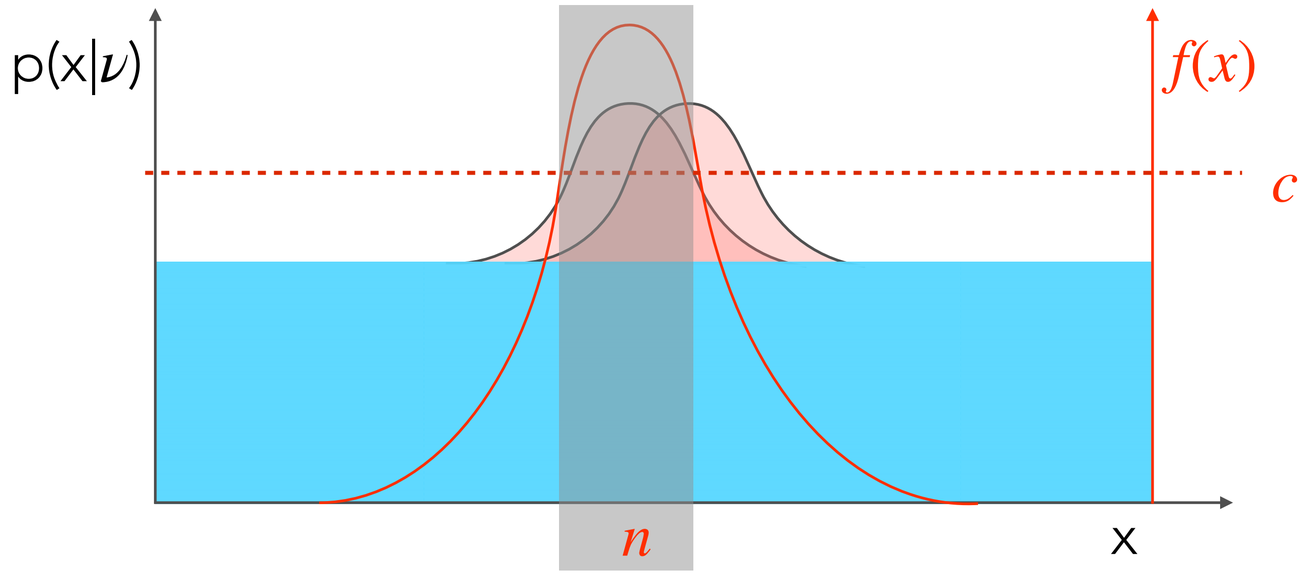


A visual example

…and later one might form a statistical model for the number of events n that have $f(x) > c$

$$\epsilon_{\text{sig}}(\nu) = \int_0^c p(f \,|\, y = 1, \nu) \mathrm{d}f \qquad \epsilon_{\text{bkg}}(\nu) = \int_0^c p(f \,|\, y = 0, \nu) \mathrm{d}f$$

$$p(n, a \,|\, \mu, \nu) = \text{Pois}(n \,|\, \mu \epsilon_{\text{sig}}(\nu) s + \epsilon_{\text{bkg}}(\nu) b) p(a \,|\, \nu)$$

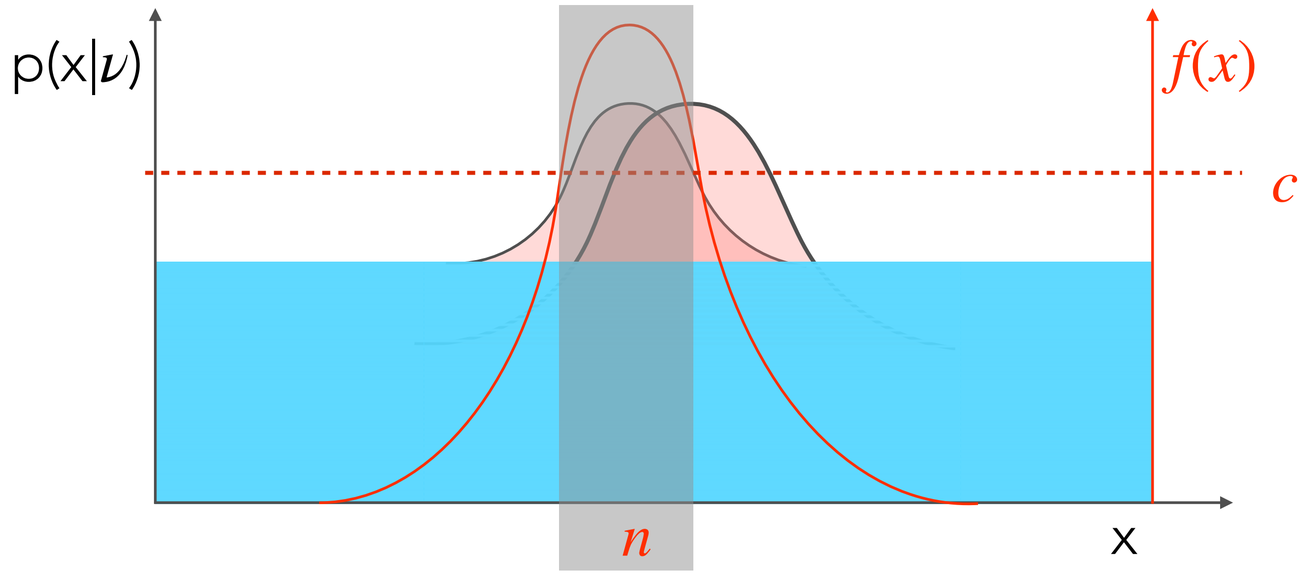


A visual example

…and later one might form a statistical model for the number of events n that have $f(x) > c$

$$\epsilon_{\text{sig}}(\nu) = \int_0^c p(f \,|\, y = 1, \nu) \mathrm{d}f \qquad \epsilon_{\text{bkg}}(\nu) = \int_0^c p(f \,|\, y = 0, \nu) \mathrm{d}f$$

$$p(n, a \,|\, \mu, \nu) = \text{Pois}(n \,|\, \mu \epsilon_{\text{sig}}(\nu) s + \epsilon_{\text{bkg}}(\nu) b) p(a \,|\, \nu)$$



Comments

The propagation of uncertainty approach is meant to ensure that downstream inference on parameter of interest μ is not wrong… (e.g. coverage)

$$p(n, a \,|\, \mu, \nu) = \text{Pois}(n \,|\, \mu \epsilon_{\text{sig}}(\nu) s + \epsilon_{\text{bkg}}(\nu) b) p(a \,|\, \nu)$$

Totally factorized from training of the ML model $f(x)$

- Therefore no reason to think that $f(x)$ is optimal from the point of view of power or sensitivity on μ even if $f(x)$ was optimal for the supervised learning task with data generated from the nominal scenario ν_0

Ok, this is all standard stuff and background. Where do we go from here?

Where do we go from here?

41. Machine Learning

Revised August 2019 by K. Cranmer (NYU), U. Seljak (UC Berkeley; LBNL) and K. Terao (SLAC).

~~Three~~ Four approaches to Systematics

propagation of errors: one works with a model $f(x)$ and simply characterizes how uncertainty in the data distribution propagate through the function to the down-stream task irrespective of how it was trained.

data augmentation: one trains a model $f(x)$ in the usual way using training data from multiple domains by sampling from some distribution over ν.

domain adaptation: one incorporates knowledge of the distribution for domains (or the parameterized family of distributions $p(x \mid y, \nu)$) into the training procedure so that the performance of $f(x)$ for the down-stream task is robust or insensitive to the uncertainty in ν.

parameterized models: instead of learning a single function of the data $f(x)$, one learns a family of functions $f(x; \nu)$ that is explicitly parameterized in terms of nuisance parameters and then accounts for the dependence on the nuisance parameters in the down-stream task.

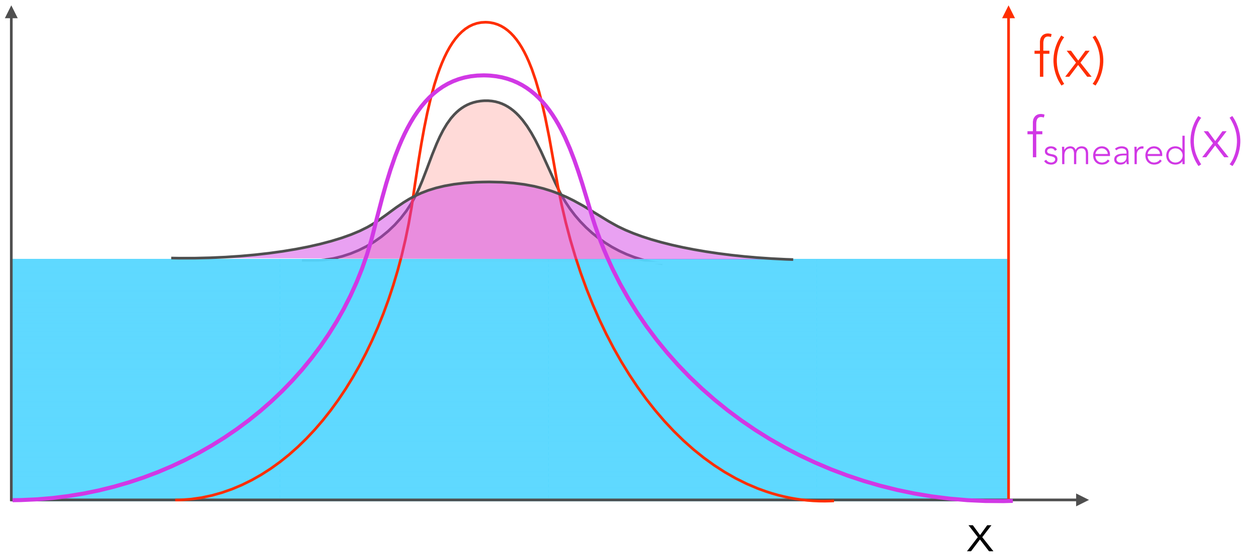
Data augmentation

An intuitive approach to incorporate systematics into training is to train on "smeared data", or data generated from a marginal model

$$x_i, y_i \sim p(x, y) = \int d\nu p(x, y|\nu) p(\nu)$$

- Note: this requires a prior / proposal distribution $p(\nu)$

$$p(x) = \int p(x\,|\,\nu) p(\nu) d\nu$$

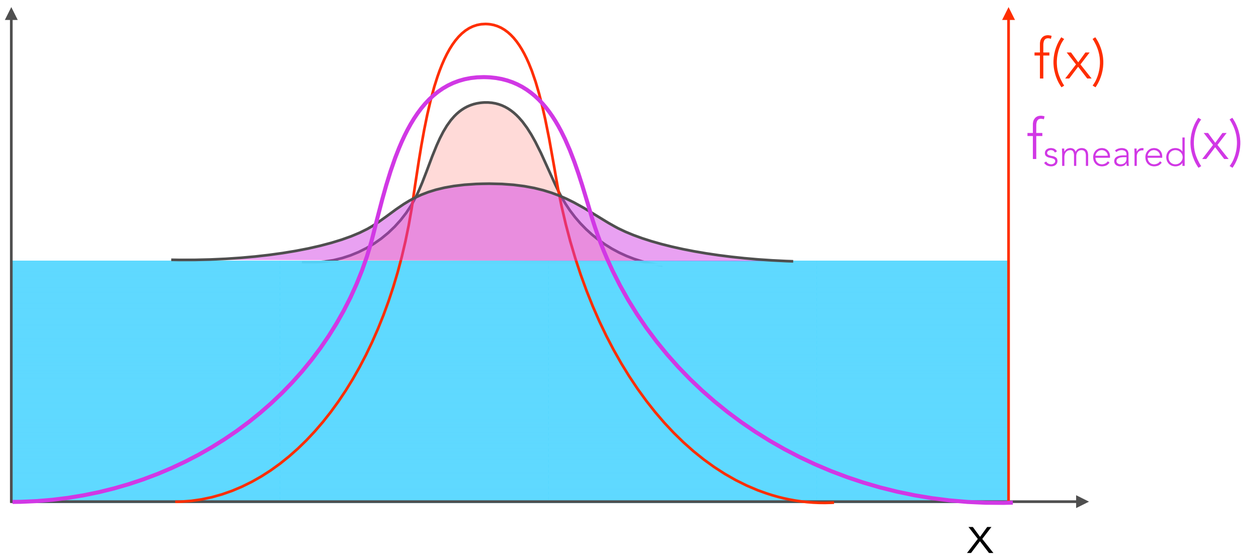


Fixed classifier is not optimal

Imagine a simple example of bump on flat background

- train on **smeared** samples with $\nu \sim p(\nu)$ to obtain fixed classifier $f_{\text{smeared}}(x)$
- we can propagate the fixed learner, but classifier not optimal for any ν

$$p(x) = \int p(x \mid \nu) p(\nu) d\nu$$

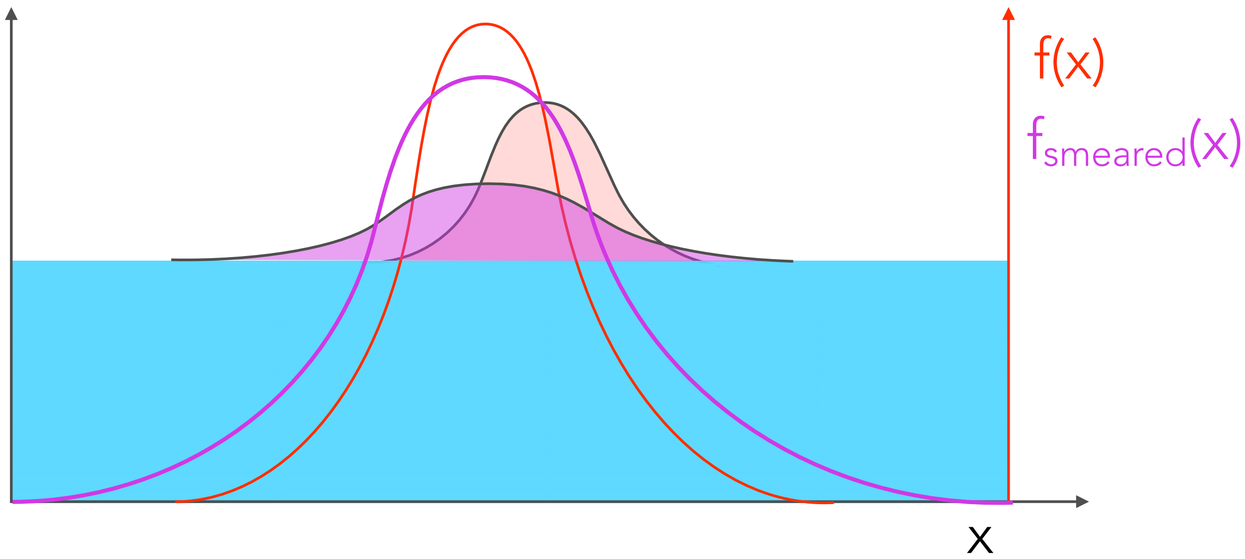


Fixed classifier is not optimal

Imagine a simple example of bump on flat background

- train on **smeared** samples with $\nu \sim p(\nu)$ to obtain fixed classifier $f_{\text{smeared}}(x)$
- we can propagate the fixed learner, but classifier not optimal for any ν

$$p(x) = \int p(x \mid \nu) p(\nu) d\nu$$

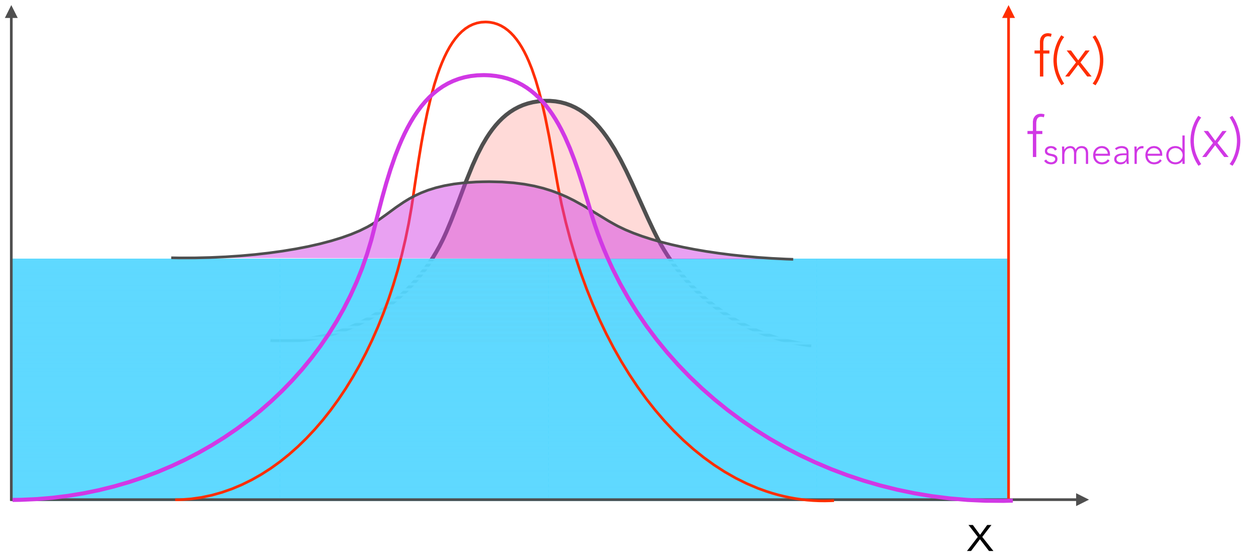


Fixed classifier is not optimal

Imagine a simple example of bump on flat background

- train on **smeared** samples with $\nu \sim p(\nu)$ to obtain fixed classifier $f_{\text{smeared}}(x)$
- we can propagate the fixed learner, but classifier not optimal for any ν

$$p(x) = \int p(x \mid \nu) p(\nu) d\nu$$

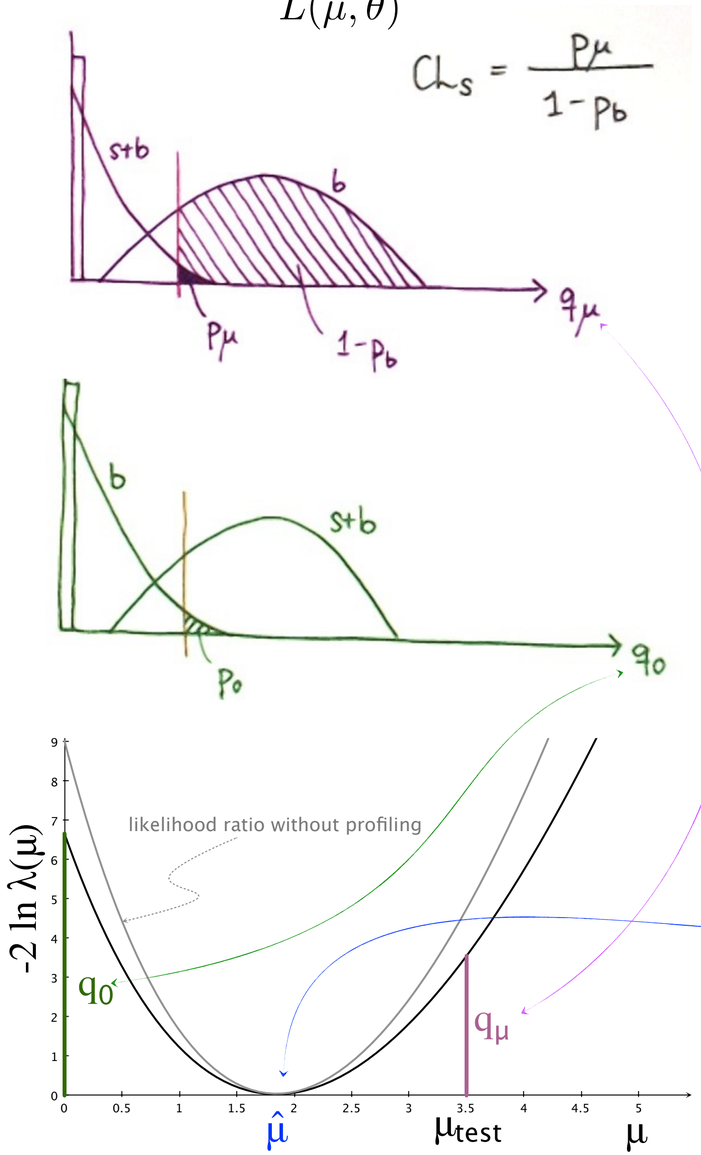


Reminder of standard statistical procedures in HEP

THUMBNAIL OF THE STATISTICAL PROCEDURE

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

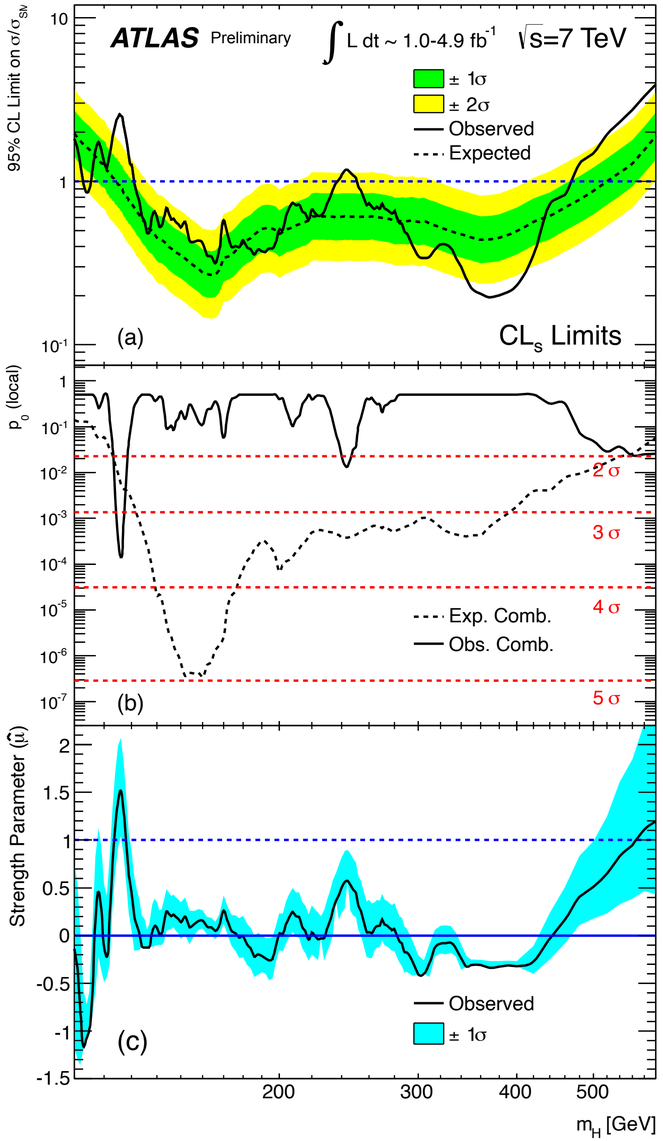
Follow LHC-HCG Combination Procedures



CL_s to test signal hypothesis

p_0 to test background hypothesis

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



The Profile Likelihood Ratio

Consider statistical model with parameters of interest μ and nuisance (here called θ)

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)
- $\mathcal{D}$ denotes observed data and $\mathcal{G}$ denotes "global observables" (central values for nuisance parameters)

Wilks' Theorem: 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

- distribution is known and it is independent of θ !
- $\Rightarrow$ robust to uncertainty, a quantity like this is called a "**pivot**"

Propagating uncertainty with a pivotal classifier

If we have a pivotal classifier, then efficiencies are independent of ν

$$\epsilon_{\text{sig}}(\nu) = \int_0^c p(f \,|\, y = 1, \nu) \mathrm{d}f \qquad \epsilon_{\text{bkg}}(\nu) = \int_0^c p(f \,|\, y = 0, \nu) \mathrm{d}f$$

Thus one won't "pay more" when accounting for systematics in the downstream analysis

$$p(n, a \,|\, \mu, \nu) = \text{Pois}(n \,|\, \mu\epsilon_{\text{sig}}(\nu)s + \epsilon_{\text{bkg}}(\nu)b)p(a \,|\, \nu)$$

$$\Rightarrow \text{Pois}(n \,|\, \mu\epsilon_{\text{sig}}s + \epsilon_{\text{bkg}}b)$$

… but that still doesn't mean that $f(x)$ is optimizing power / sensitivity.

How do we obtain a pivotal classifier?

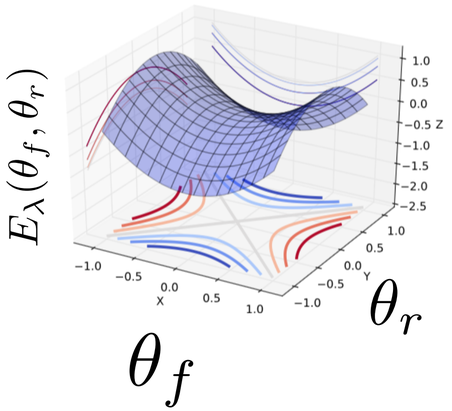
Learning to pivot with adversarial networks

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

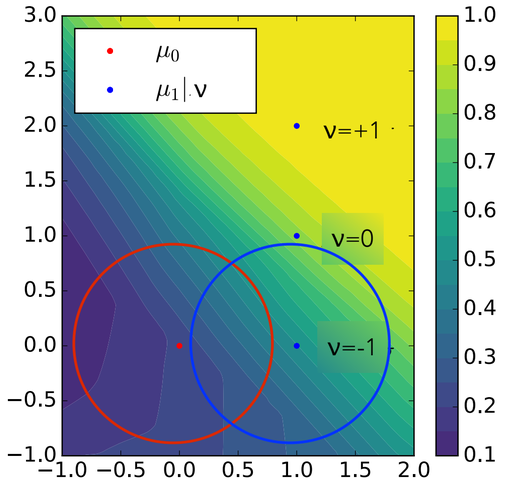
- want classifier output to be insensitive to systematics (nuisance parameter $\boldsymbol{\nu}$)
- introduce an **adversary r** that tries to predict ν based on f.
- setup as a minimax game:

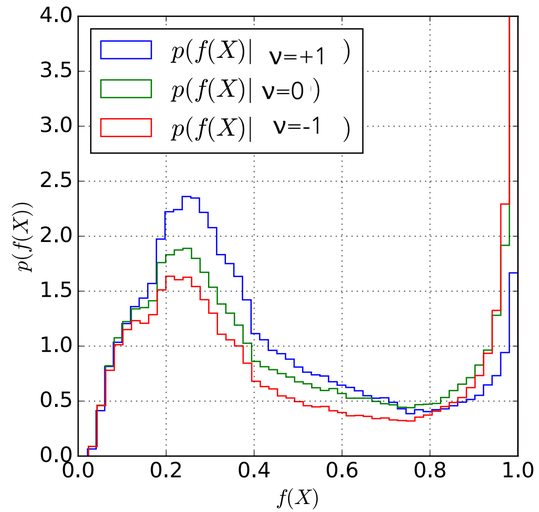
$$\hat{\theta}_f, \hat{\theta}_r = \arg\min_{\theta_f} \max_{\theta_r} E(\theta_f, \theta_r).$$

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

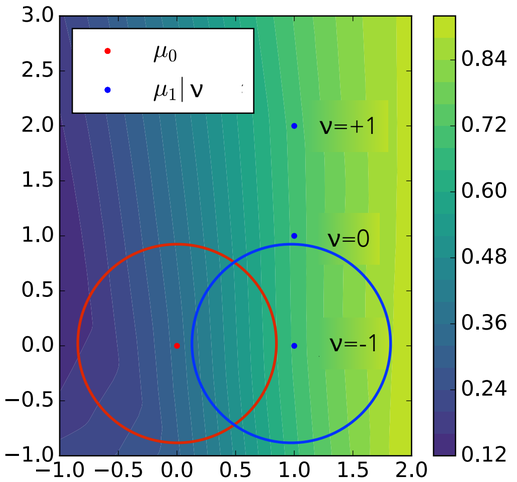


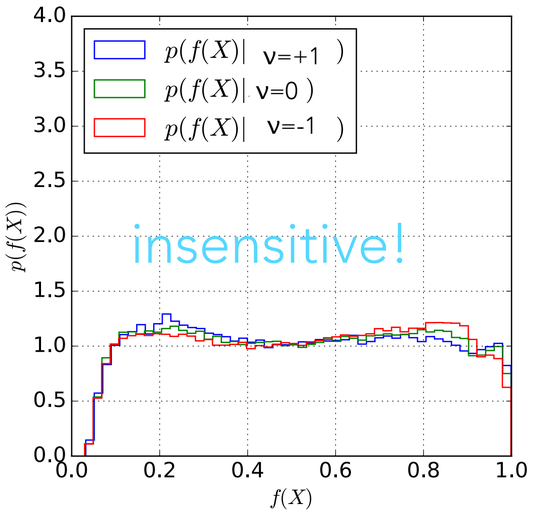
normal training





adversarial training

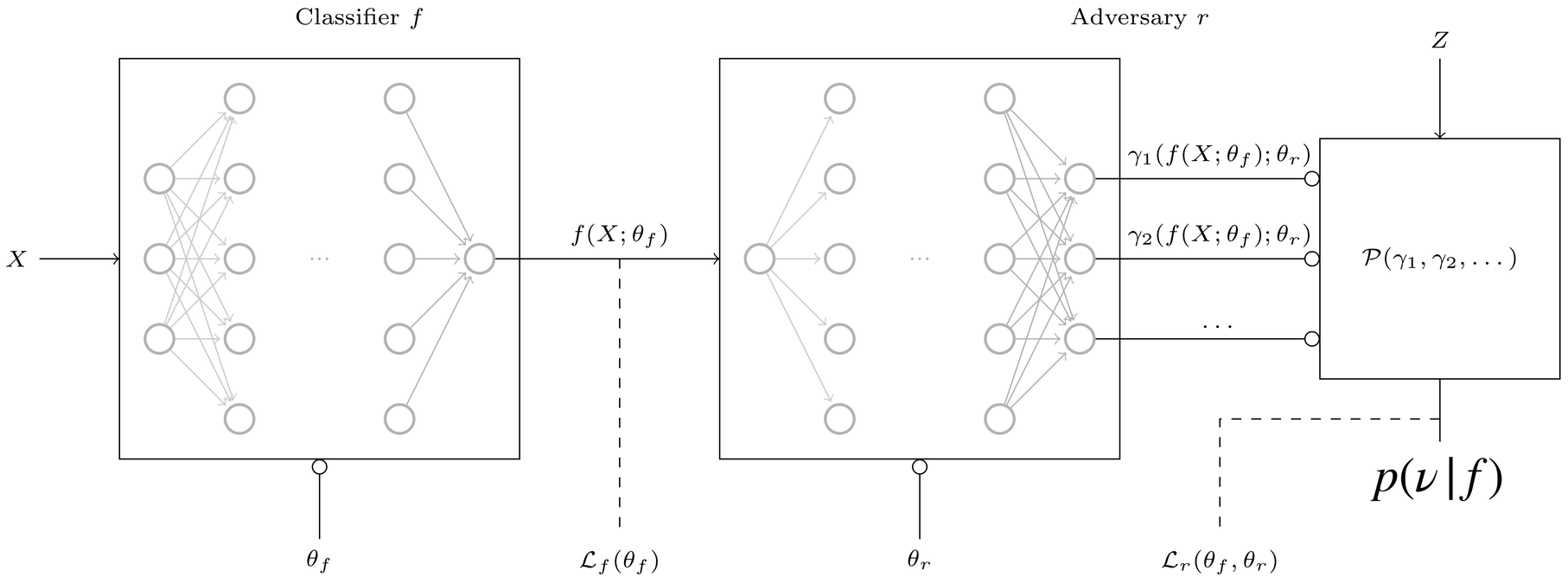




G. Louppe, M. Kagan, K. Cranmer, arXiv:1611.01046

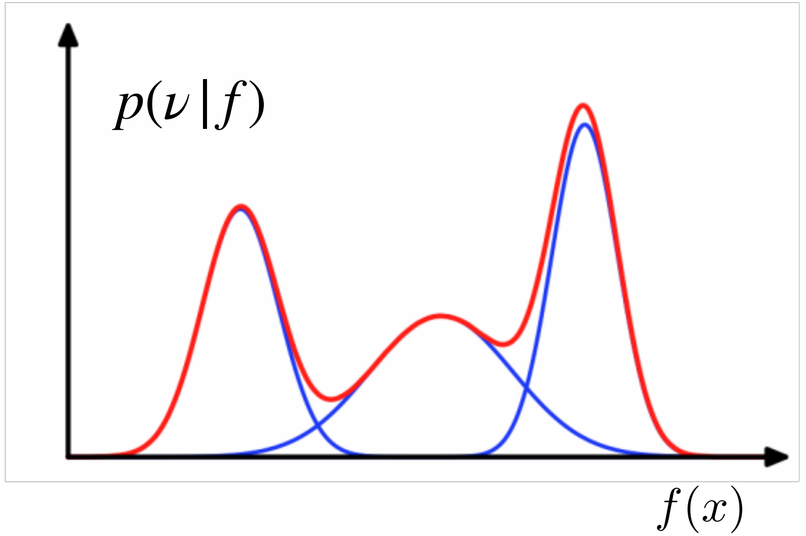
The Adversarial model

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



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

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



An example of learning to pivot

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

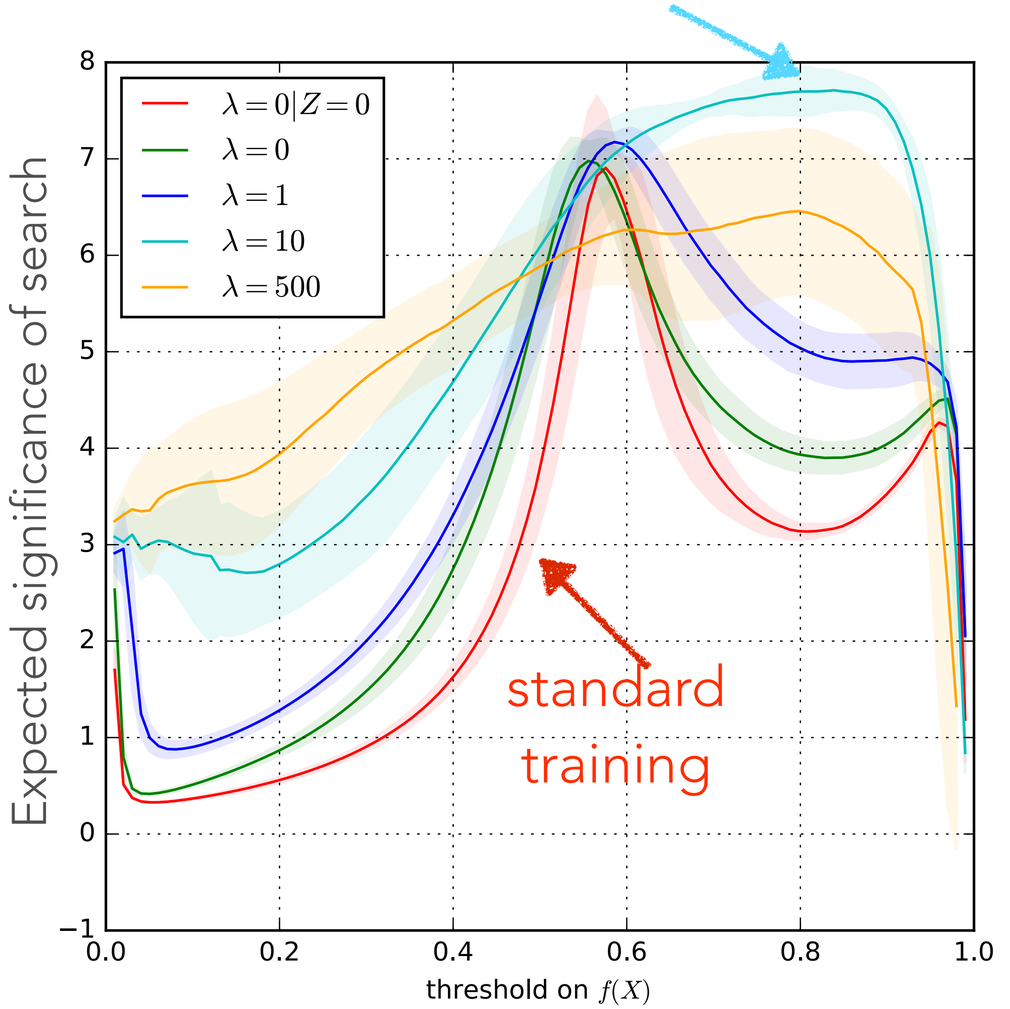
signal: 100 boosted W's

Train W vs. QCD classifier

Pileup as source of uncertainty

Simple cut-and-count analysis with background uncertainty.

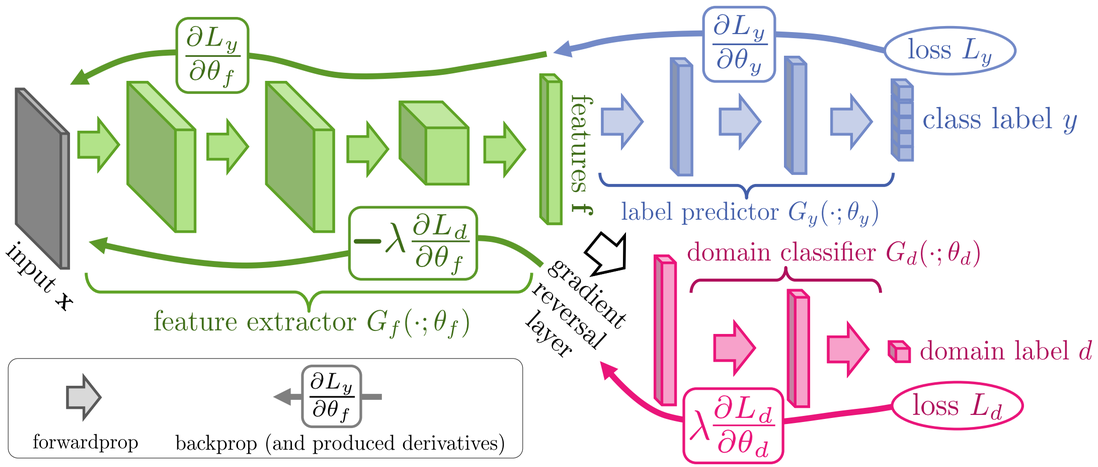
optimal tradeoff of classification vs. & robustness



Domain adaptation

https://arxiv.org/pdf/1505.07818.pdf

Ganin, Ustinova, Ajakan, Germain, Larochelle, Laviolette, Marchand and Lempitsky



In machine learning literature, the setting where training data doesn't match real world data is referred to as "domain shift" and techniques to mitigate the loss in performance are called "domain adaptation"

A similar adversarial technique was introduced in arxiv:1505.07818 where adversary tries to get distribution of hidden state features to be invariant. This works for discrete domains, but doesn't generalize well to continuous nuisance parameters.

- adversary works on some low-level features (not just the class prediction)

Learned adversary → explicit regularization

Phys.Rev.Lett. 125 (2020) 12, 122001

One way of interpreting the mini-max game $\hat{\theta}_f, \hat{\theta}_r = \arg\min_{\theta_f} \max_{\theta_r} E(\theta_f, \theta_r).$ is to minimize a **regularized** loss term $\tilde{L}(\theta_f) = \arg\max_{\theta_r} E_\lambda(\theta_f, \theta_r)$ where the optimization with respect to θ_r is not exposed

This motivates another approach in which the regularization is not achieved through a learned adverary, but some other measure of discrepancy

DisCo Fever: Robust Networks Through Distance Correlation

Gregor Kasieczka[1,*] and David Shih[2,3,4,†]

$$L = L_{classifier}(\vec{y}, \vec{y}_{true}) + \lambda \, \mathrm{dCorr}^2_{y_{true}=0}(\vec{m}, \vec{y})$$

$$\begin{aligned}\mathrm{dCov}^2(X,Y) = &\langle |X - X'||Y - Y'| \rangle \\ &+ \langle |X - X'| \rangle \langle |Y - Y'| \rangle \\ &- 2\langle |X - X'||Y - Y''| \rangle\end{aligned}$$

Is there a better way?

The Profile Likelihood Ratio

Consider statistical model with parameters of interest μ and nuisance (**here called** θ)

Define **profile likelihood ratio**

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

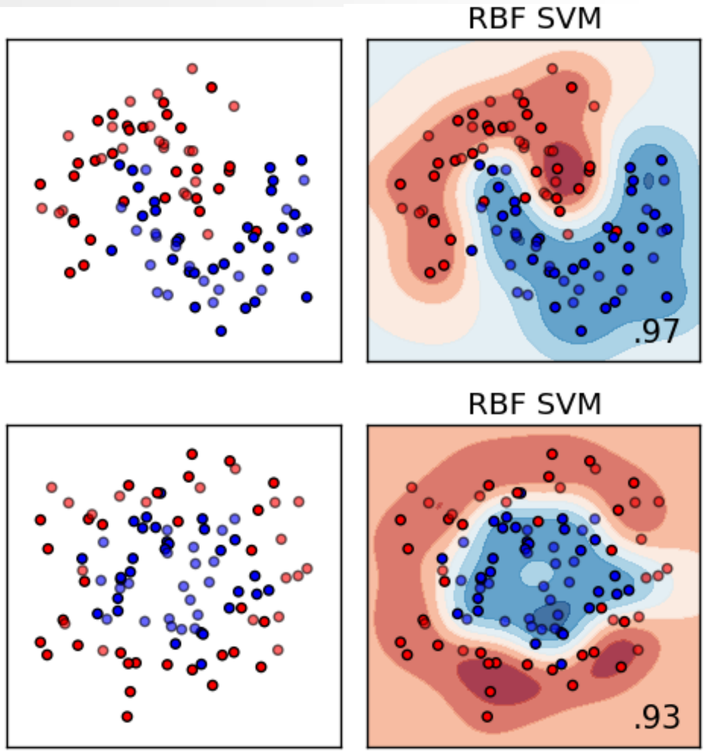
- where $\hat{\hat{\theta}}(\mu; \mathcal{D}, \mathcal{G})$ is best fit with μ fixed (the constrained maximum likelihood estimator, depends on data)
- and $\hat{\theta}$ and $\hat{\mu}$ are best fit with both left floating (unconstrained)
- $\mathcal{D}$ denotes observed data and $\mathcal{G}$ denotes "global observables" (central values for nuisance parameters)

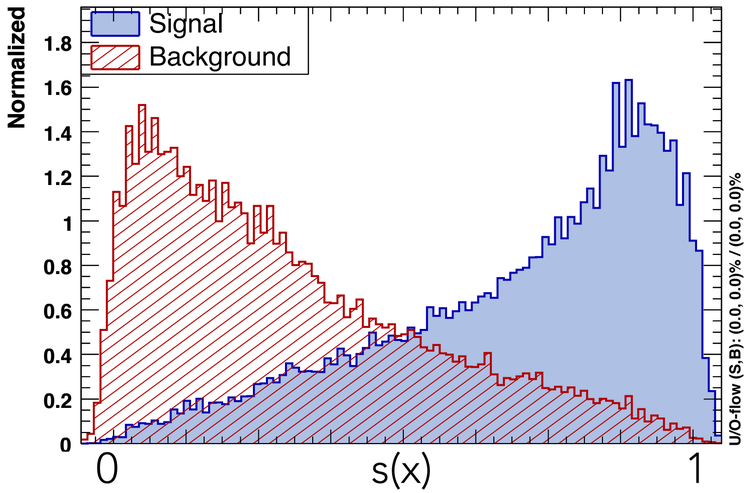
The data $\mathcal{D} = \{x_1, \ldots, x_N\}$ are iid, so the likelihood is just a product over events.

- Profiling introduces a coupling across events: $\hat{\hat{\theta}}(\mu; \mathcal{D}, \mathcal{G})$
- But we can postpone profiling to the final inference stage and frame optimal ML model (eg. classifier) at the event-level by targeting the likelihood ratio

Likelihood Ratio Trick

[Hastie et al., 2001; Sugiyama et al., 2012; Cranmer et. al., 2015]





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

$$L[s] = \mathbb{E}_{p(x|H_1)}[-\log s(x)] + \mathbb{E}_{p(x|H_0)}[-\log(1 - s(x))]$$

- i.e. approximate the optimal classifier

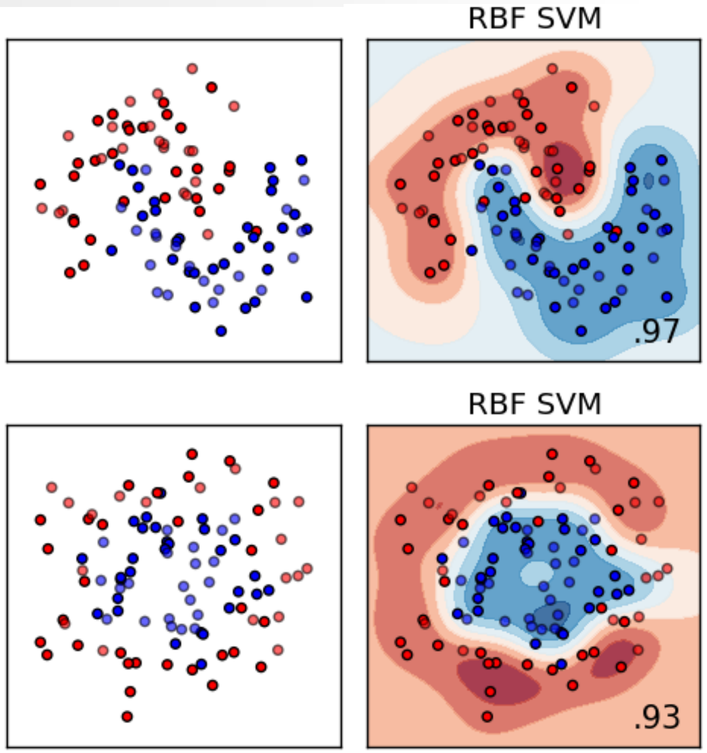
$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

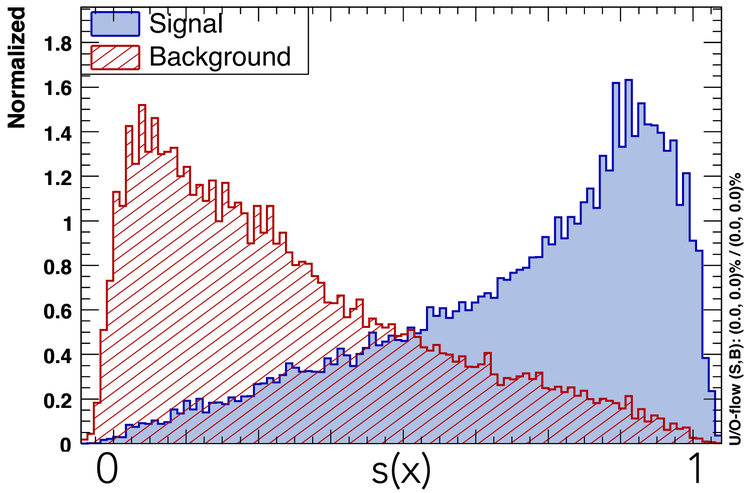
- which is 1-to-1 with the likelihood ratio

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

Likelihood Ratio Trick

[Hastie et al., 2001; Sugiyama et al., 2012; Cranmer et. al., 2015]





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

$$L[s] = \mathbb{E}_{p(x|H_1)}[-\log s(x)] + \mathbb{E}_{p(x|H_0)}[-\log(1-s(x))]$$

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

- 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

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

Parametrizing the Likelihood Ratio Trick

Can do the same thing for any two points θ_0 & θ_1 in parameter space Θ.

$$r(x;\theta_0,\theta_1) = \frac{p(x \mid \theta_0)}{p(x \mid \theta_1)} = 1 - \frac{1}{s(x;\theta_0,\theta_1)}$$

Or train to classify data from $p(x|\theta)$ versus some fixed reference $p_{\text{ref}}(x)$

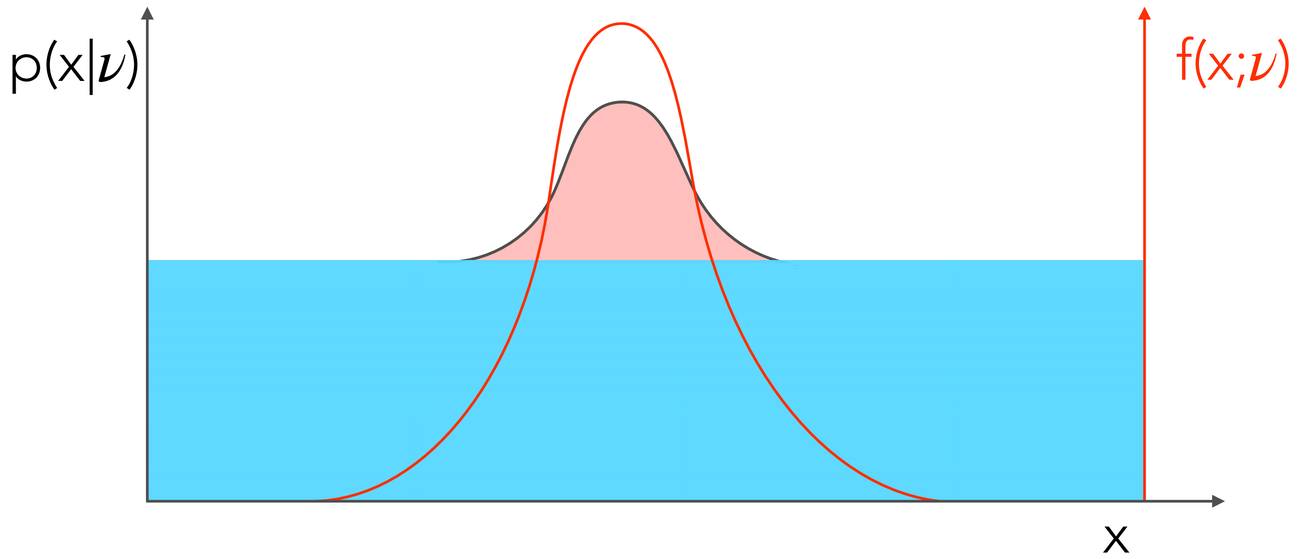
$$r(x;\theta) = \frac{p(x|\theta)}{p_{\text{ref}}(x)} = 1 - \frac{1}{s(x;\theta)}$$

I call this a **parametrized classifier.**

Visualizing the parameterized classier

We want a learner parametrized by ν

- augment training data (x,y) → (x,ν,y) to obtain f(x;ν)

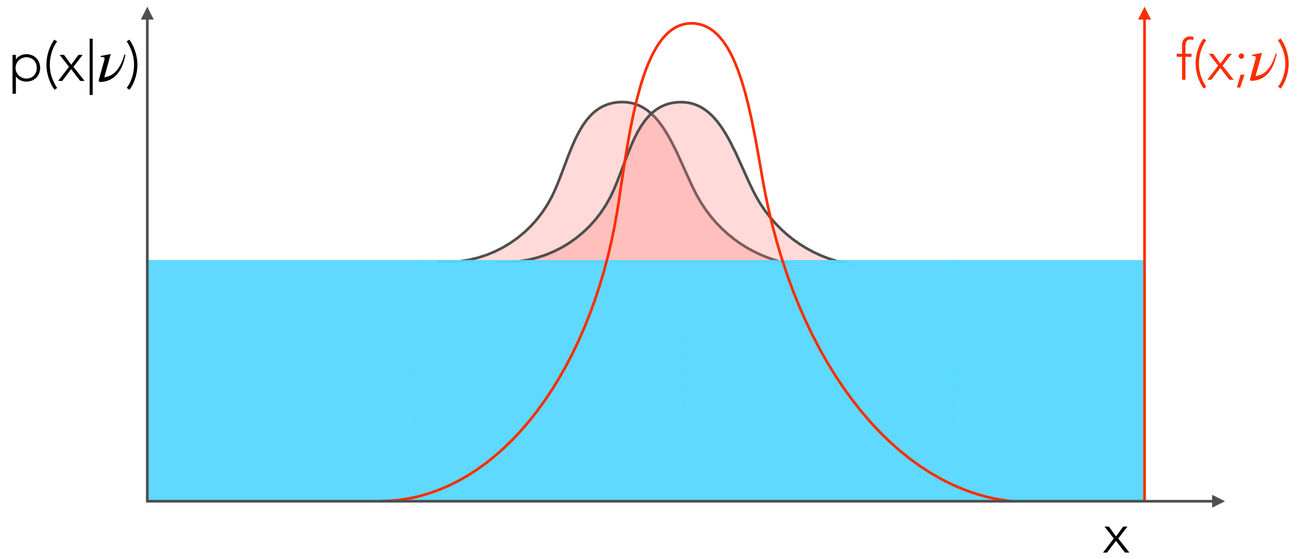


- **problem**: how do we evaluate on testing data when ν is unknown?

Visualizing the parameterized classier

We want a learner parametrized by ν

- augment training data (x,y) → (x,ν,y) to obtain f(x;ν)

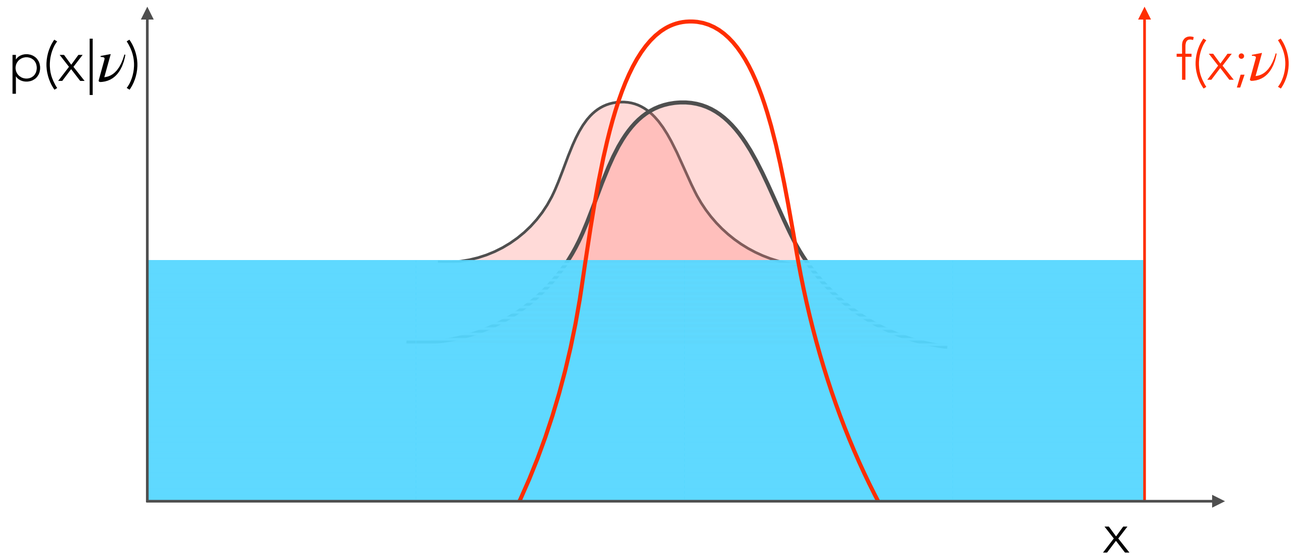


- **problem**: how do we evaluate on testing data when ν is unknown?

Visualizing the parameterized classier

We want a learner parametrized by ν

- augment training data (x,y) → (x,ν,y) to obtain f(x;ν)

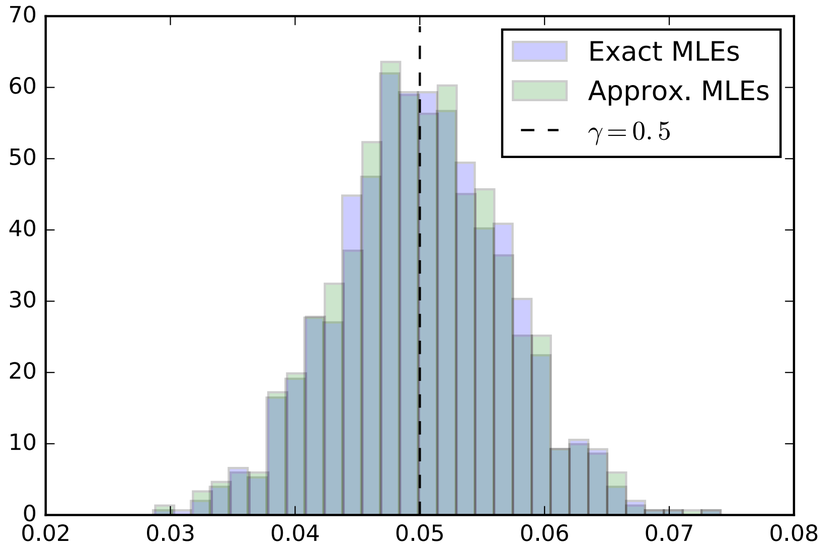


- **problem**: how do we evaluate on testing data when ν is unknown?

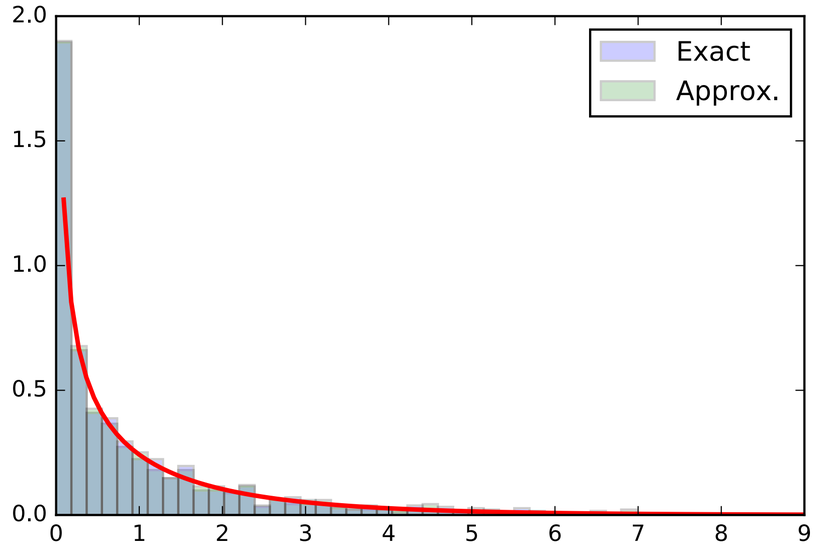
Amortized likelihood ratio

Once we've learned the likelihood ratio $r(x;\theta)$, we can apply it to any data x.

- It is amortized, we pay biggest computational costs up front
- Great for calibrated frequentist confidence intervals with guaranteed coverage
- Here we repeat inference thousands of times & check asymptotic statistical theory



(a) Exact vs. approximated MLEs.



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

Calibrating the likelihood-ratio trick

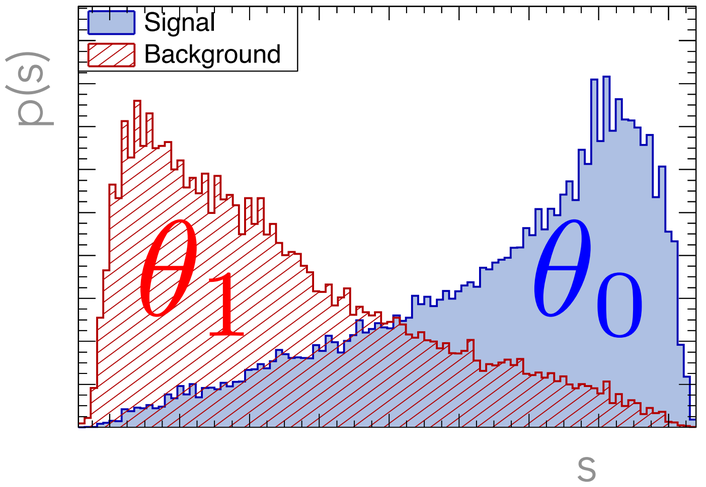
We can weaken the requirements for the likelihood ratio trick in case the classifier

If the scalar map s: X → ℝ has the same level sets as the likelihood ratio

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

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

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



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

Cranmer, Louppe, Pavez: Approximating Likelihood Ratios with Calibrated Discriminative Classifiers [arXiv:1506.02169]
[Dalmasso, Izbicki, Lee, ICML2020 arXiv:2002.10399]

Some parameterized classifier history

2015 NeurIPS ML & Physics workshop:

- http://yandexdataschool.github.io/aleph2015/
- https://indico.cern.ch/event/465572/

ALEPH Workshop @ NIPS 2015

Applying (machine) Learning to Experimental Physics (ALEPH) and «Flavours of Physics» challenge

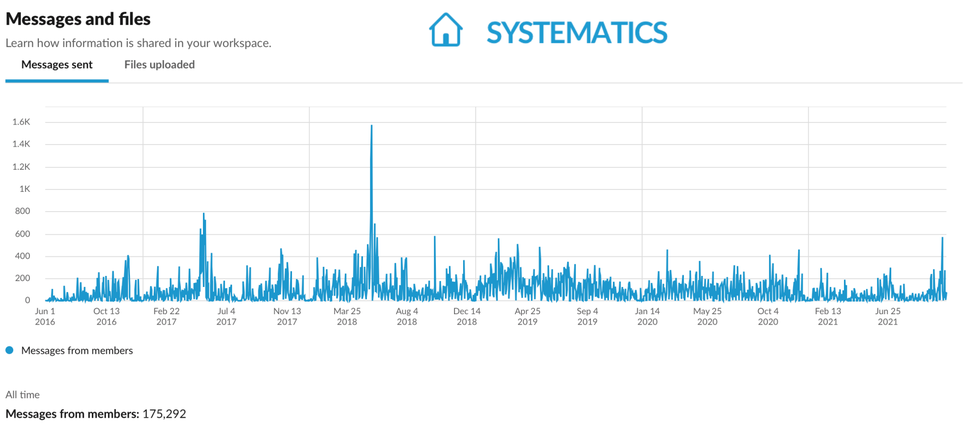
When: **11th of December 2015, 8:30 - 18:30**
Where: **room 515 bc**, NIPS, Montreal, Canada

2016 ATLAS Statistics & ML workshop:

- https://indico.cern.ch/event/577209/

Thinking about systematics in ML is what inspired my work in simulation-based inference

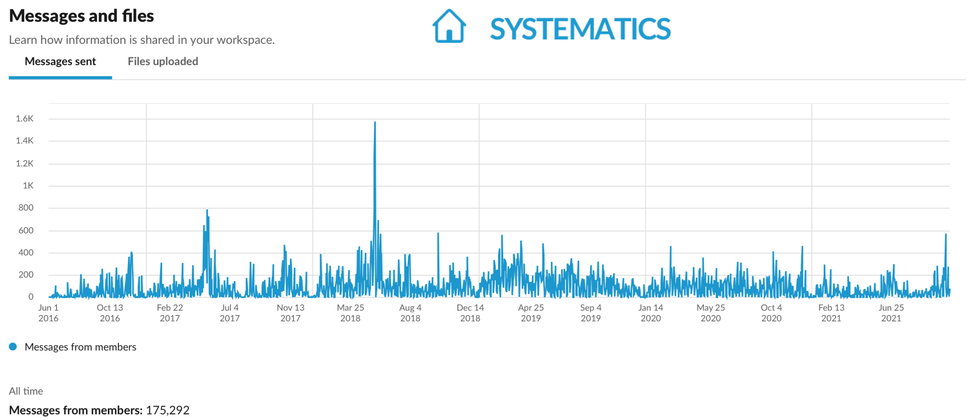
- Slack channel I use for ML work is called "**systematics**" and has 115 members and >175,000 messages!



Some parameterized classifier history

Thinking about systematics in ML is what inspired my work in simulation-based inference

- Slack channel I use for ML work is called "**systematics**" started in 2016, has 115 members and >175,000 messages!



2015 NeurIPS ML & Physics workshop:

- http://yandexdataschool.github.io/aleph2015/
- https://indico.cern.ch/event/465572/

2016 ATLAS Statistics & ML workshop:

- https://indico.cern.ch/event/577209/

ALEPH Workshop @ NIPS 2015

Applying (machine) Learning to Experimental Physics (ALEPH) and «Flavours of Physics» challenge

When: **11th of December 2015, 8:30 - 18:30**
Where: **room 515 bc**, NIPS, Montreal, Canada

A review with other approaches

Dealing with Nuisance Parameters using Machine Learning in High Energy Physics: a Review

T. Dorigo and P. de Castro Manzano

Istituto Nazionale di Fisica Nucleare - Sezione di Padova,
Via Marzolo 8, 35131 Padova - Italy,
tommaso.dorigo@cern.ch pablo.de.castro@cern.ch*

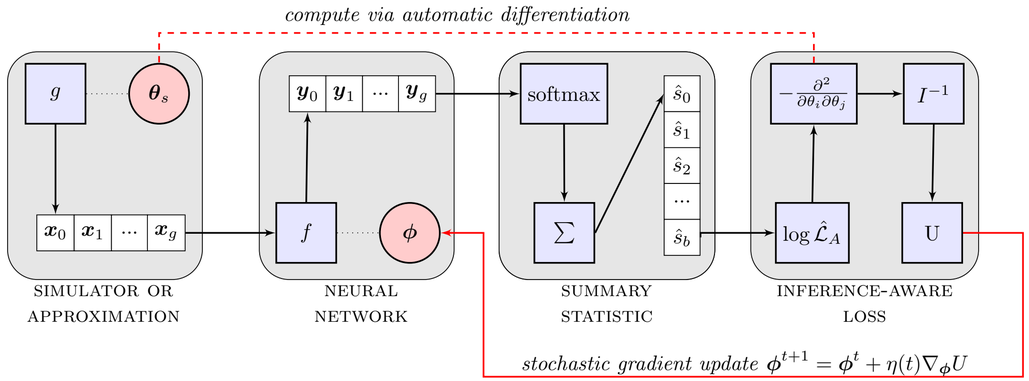
In this work we discuss the impact of nuisance parameters on the effectiveness of machine learning in high-energy physics problems, and provide a review of techniques that allow to include their effect and reduce their impact in the search for optimal selection criteria and variable transformations. The introduction of nuisance parameters complicates the supervised learning task and its correspondence with the data analysis goal, due to their contribution degrading the model performances in real data, and the necessary addition of uncertainties in the resulting statistical inference. The approaches discussed include nuisance-parameterized models, modified or adversary losses, semi-supervised learning approaches, and inference-aware techniques.

To appear in "Artificial Intelligence for Particle Physics", World Scientific Publishing Co
https://inspirehep.net/literature/1807719

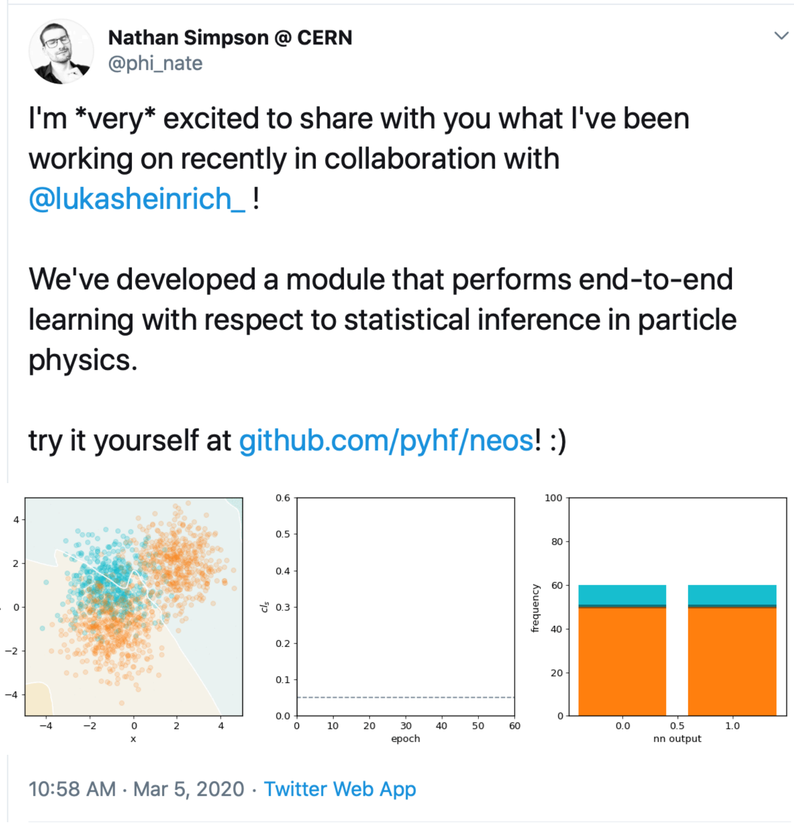
Tradition meets differentiable programming

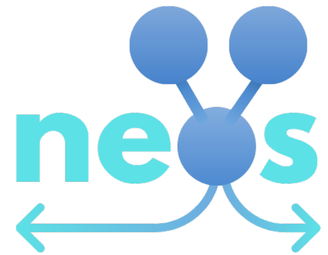
Recent efforts in particle physics to maintain traditional approaches to likelihood estimation with summaries, but optimize summary statistics with automatic differentiation

- Connects to differentiable programming paradigm
- Optimization objective is power of full statistical analysis, which involves backproping through statistical procedure
- Does not exploit i.i.d. property, optimization is "global"



INFERNO: de Castro & Dorigo, [arXiv:1806.04743]



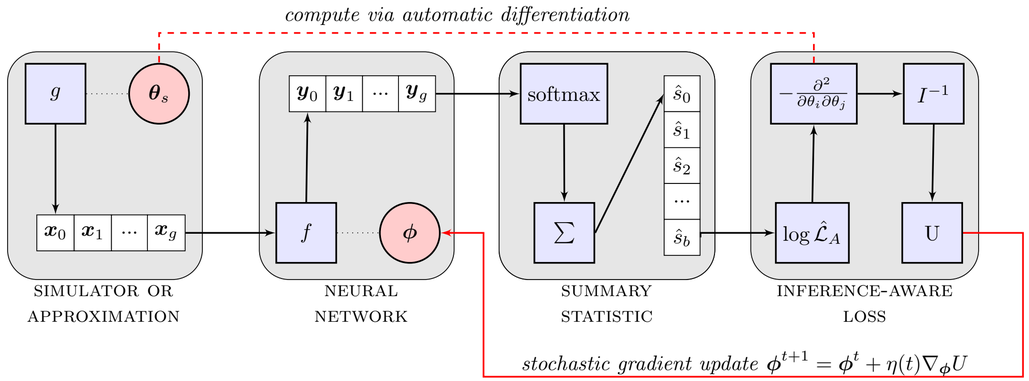


https://github.com/pyhf/neos

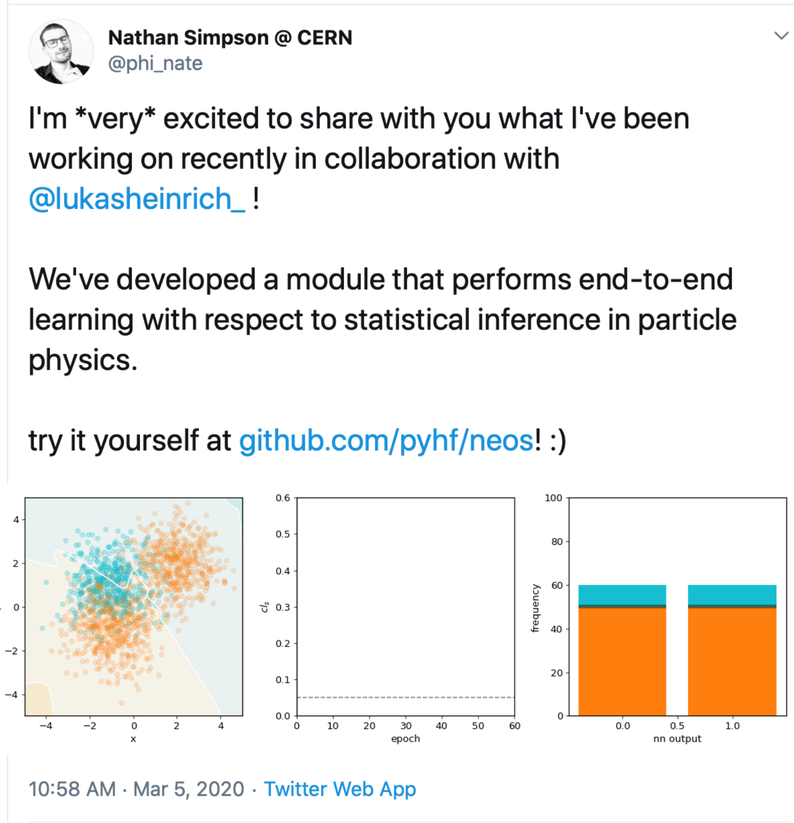
Tradition meets differentiable programming

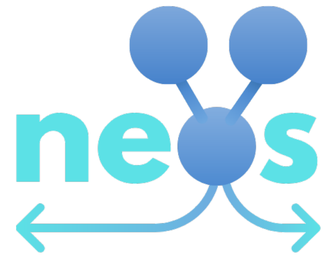
Recent efforts in particle physics to maintain traditional approaches to likelihood estimation with summaries, but optimize summary statistics with automatic differentiation

- Connects to differentiable programming paradigm
- Optimization objective is power of full statistical analysis, which involves backproping through statistical procedure
- Does not exploit i.i.d. property, optimization is "global"



INFERNO: de Castro & Dorigo, [arXiv:1806.04743]





https://github.com/pyhf/neos

Conclusion

Traditional (non-ML) approaches used by physicists design analyses to be robust to systematic uncertainties through heuristics

- While not formalized, physicists are quite good at this

First wave of ML in physics optimized for a nominal scenario and then propagate uncertainty through the ML components in downstream inference

- “Not optimal, but not wrong”

In order to optimize sensitivity in an ML context, we need to formalize what we want (or some heuristic) so that we can operationalize it

- e.g. a pivotal classifier. Tradeoff between dependence on nuisance parameter and classification/regression performance
- a parameterized classifier: more moving parts, but closest to the ideal single-step likelihood ratio test in high dimensions

A comment on Aleatoric and Epistemic Uncertainty

“roughly speaking, aleatoric (aka statistical) uncertainty refers to the notion of randomness, that is, the variability in the outcome of an experiment which is due to inherently random effects”, while “epistemic (aka systematic) uncertainty refers to uncertainty caused by a lack of knowledge (about the best model)”.

In the literature on Uncertainty Quantification (UQ), which is more closely connected to physics given the role of computer simulations, the terminology is more fine grained and less ambiguous.

That community uses the terms:

- parameter uncertainty (i.e. nuisance parameters),
- structural uncertainty (i.e. mismodelling),
- algorithmic uncertainty (i.e. numerical uncertainty),
- experimental uncertainty (i.e. uncertainty from experimental resolution and statistical fluctuations), and
- interpolation uncertainty (i.e. uncertainty due to interpolating between different parameter values due to lack of computational resources).

Perhaps a more important distinction between the perspective of physicists and machine learning researchers has to do with the use of the term “model” and what exactly is uncertain. In physics, the systematic and epistemic uncertainty is typically associated to our understanding of the underlying physics and “the model” usually refers to the physics model, detector model encapsulated in a simulation. In contrast, for machine learning research, “the model” usually refers to the trained model $f \in F$ used as described in Section 41.2.1 (or the class of functions F itself). This makes sense if we recall that in the bulk of machine learning research, one has little insight into the process that generated the data (e.g. images of cats and dogs, natural language, etc.). In that sense, the epistemic uncertainty in machine learning is usually associated to uncertainty in the model parameters φ after training, which would be reduced if one could collect more training data (see Ref. [328] for this point of view).