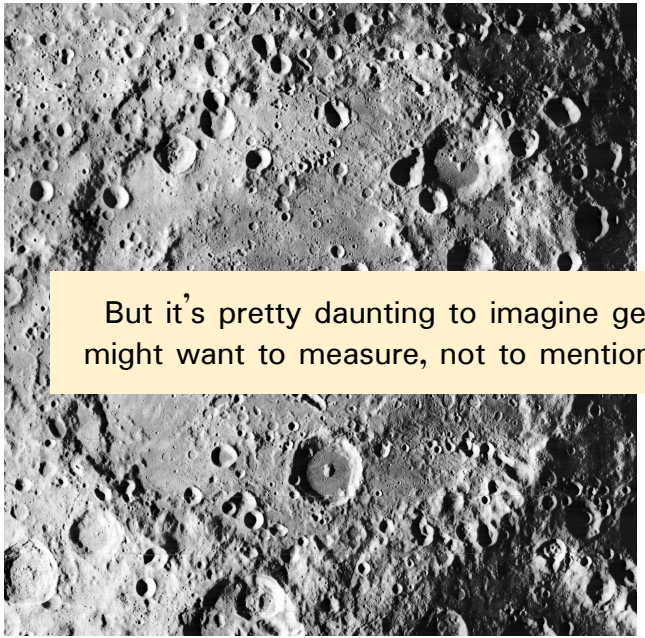
Using classifiers for unbinned unfolding

Mariel Pettee • June 11th, 2024

BERKELEY LAB
Bringing Science Solutions to the World

Does this image of a lunar crater confirm or contradict a given theory of crater formation?

Based on our understanding of astrophysics, we can make detailed simulations of craters...



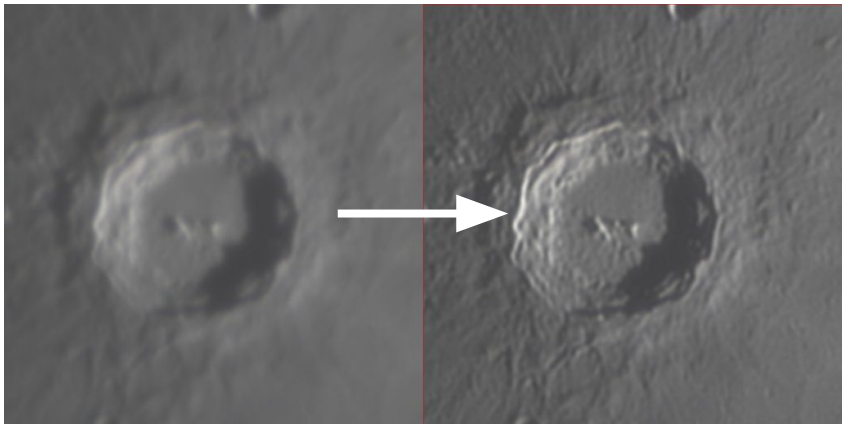
...and then we could make simulations to approximate the effects of our instrument.

But it's pretty daunting to imagine generating simulations for every kind of crater we might want to measure, not to mention accurately modeling each part of the telescope!

Alternatively, we can build software tools to “un-blur” (“unfold”) our data.

Given a smeared dataset, and some knowledge of our detector, we can try to estimate the true data.

Smeared image of a lunar crater **Unfolded image of a lunar crater**



“Unfolding” means removing unwanted detector effects from our experimental data.

This has the advantage of correcting a whole dataset on a statistical level, allowing us to combine data from multiple sources, and makes the data more flexible & useful for future analyses.

In contemporary data analysis for fundamental physics, we want more from our unfolding strategies.

Traditional unfolding approaches (binned & few-dimensional) have significant room for improvement.

Specifically:

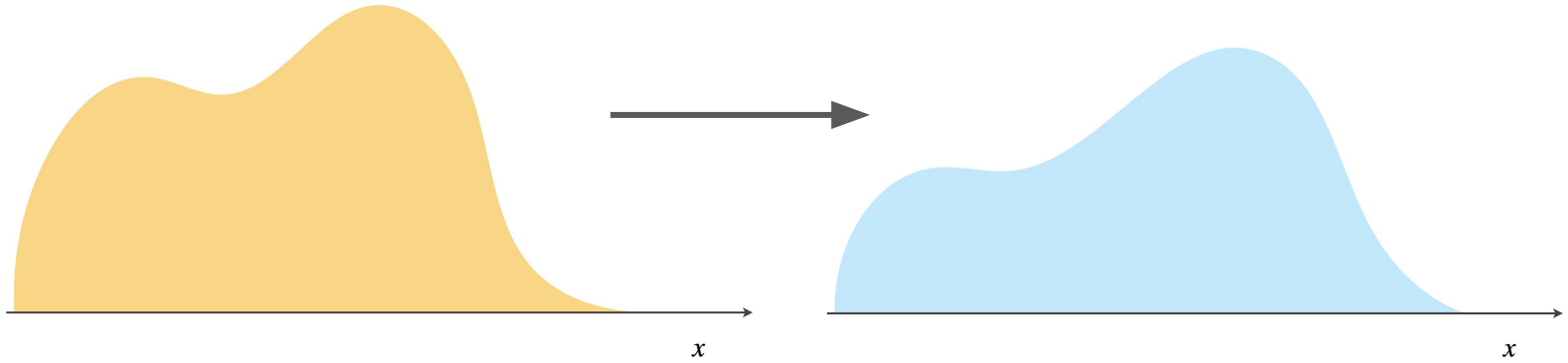
- Any future user of unfolded data would ideally want to choose their own bins for their measurement, but this is not possible with binned strategies in which bins are pre-determined.
- A future user of unfolded data might want to modify the phase space, but this is basically not possible with binned measurements.
- Measurements of properties that are a function of many observables are not possible with a few-dimensional unfolding.

One important category of unfolding techniques is based on **classification.**

Other talks today will describe other approaches, e.g. **generative models.**

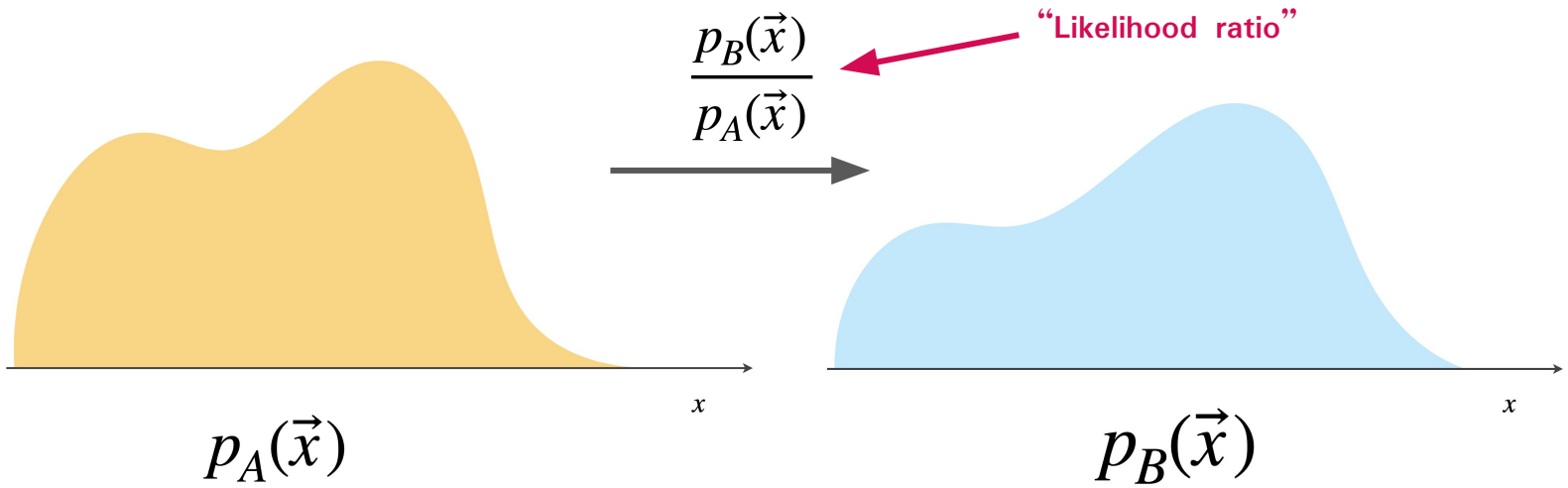
How are classification and reweighting connected?

Q: How can we adjust one distribution to look like another?



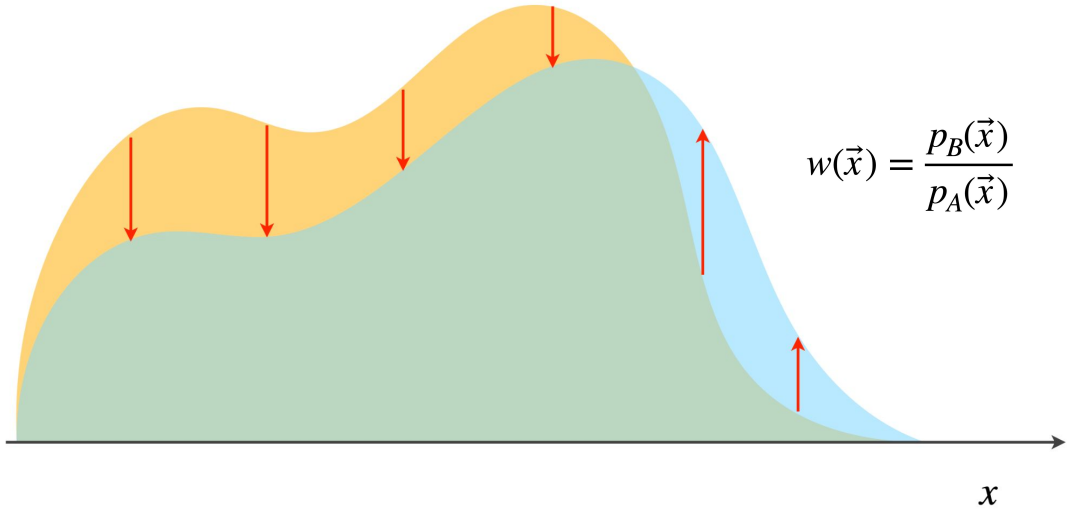
Q: How can we adjust one distribution to look like another?

A: Learn a reweighting function based on the ratio of their probability densities.



Q: How can we adjust one distribution to look like another?

A: Learn a reweighting function based on the ratio of their probability densities.



In practice, calculating individual densities $p_A(x)$ and $p_B(x)$ can be hard.

But machine learning strategies can help by **directly approximating the ratio of the likelihoods.**

Classifier functions can be re-used to directly approximate a likelihood ratio.

A vanilla NN classifying between two classes could be trained using **binary cross-entropy loss:**

$$L_{\mathrm{BCE}}[f] = -\int \mathrm{d}x \left(p_A(x) \log(f(x)) + p_B(x) \log(1-f(x)) \right)$$

where $f(x)$ is the output of a NN classifier, and our datasets are sampled from these two probability distributions $p_A(x)$ and $p_B(x)$.

Classifier functions can be re-used to directly approximate a likelihood ratio.

A vanilla NN classifying between two classes could be trained using **binary cross-entropy loss:**

$$L_{\mathrm{BCE}}[f] = -\int \mathrm{d}x \left(p_A(x) \log(f(x)) + p_B(x) \log(1-f(x)) \right)$$

To find where this is minimized, we need to find the extremum, i.e. differentiate with respect to $f(x)$ and set equal to 0:

$$\frac{\partial L}{\partial f} = -\frac{\partial}{\partial f}\left(p_A(x) \log(f(x)) + p_B(x) \log(1-f(x)) \right)$$

$$= -\frac{p_A(x)}{f(x)} + \frac{p_B(x)}{1-f(x)}$$

$$\frac{\partial L}{\partial f} = 0 \Rightarrow \frac{f(x)}{1-f(x)} = \frac{p_A(x)}{p_B(x)}$$

Rescaling of classifier output

Likelihood ratio

Classifier–based unfolding works well in practice and has several advantages.

In particular, it learns small corrections to MC, meaning it starts from a good baseline solution.

Maximum–likelihood classifier–based unfolding is also prior–independent.

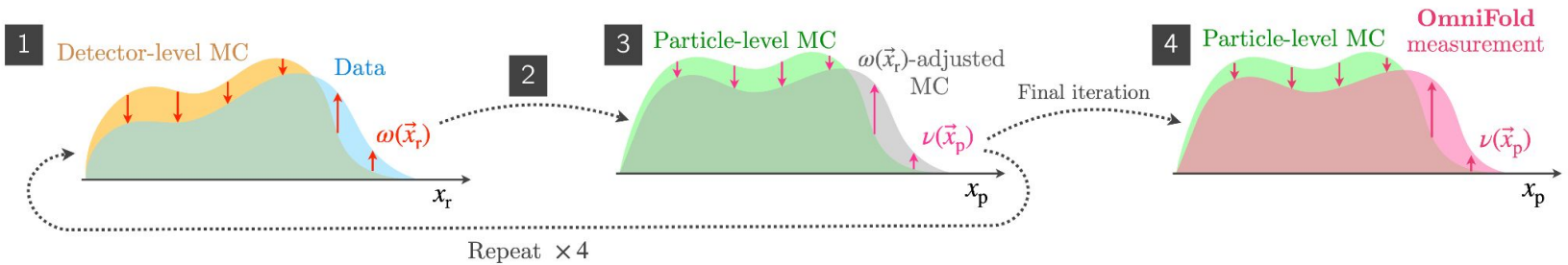
Putting this into practice: OmniFold

OmniFold overview

The OmniFold procedure requires two datasets* as inputs:
(*In practice, we use samples from different MC generators as well as systematically-shifted samples to determine uncertainties.)

- MC sample with events at both detector-level and particle-level
- Real data
  - (In fact, it's the only unfolding method of this kind that has been applied to real data)

In a multi-stage and iterative process, a series of neural networks are trained to learn a reweighting function that maps particle-level MC distributions to particle-level data distributions.



Note: Binned OmniFold reduces to Iterative Bayesian Unfolding (IBU)!

Each iteration of MultiFold consists of 2 reweighting stages.

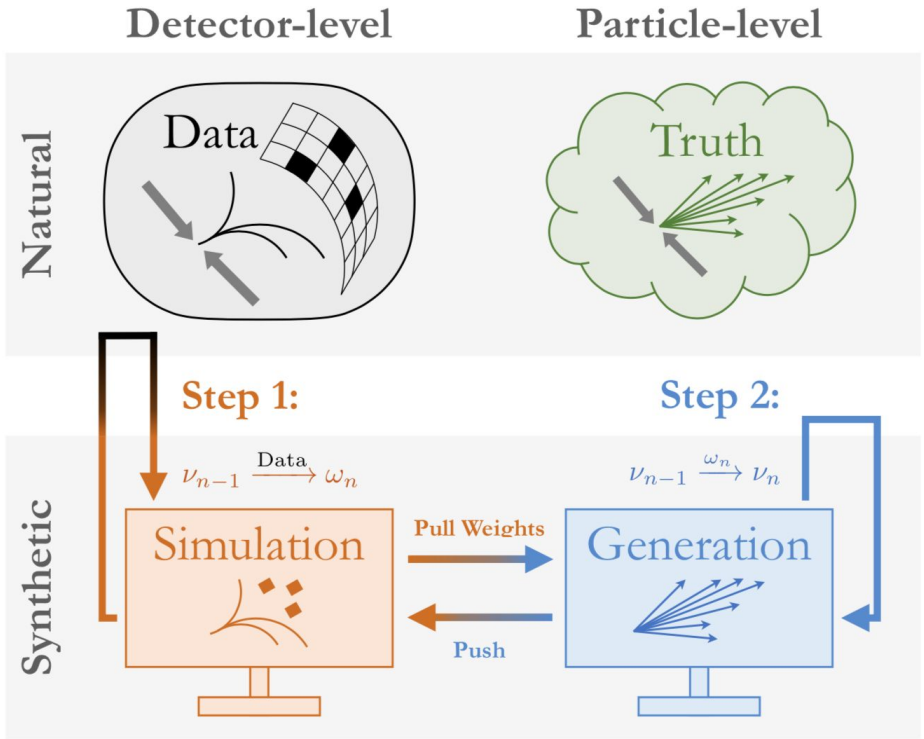
1. Reweight MC Reco to match Data Reco

2: Reweight MC Truth to match the reweighted MC Reco from Step 1

→ Ensures that if two identical particle-level events will be given the same weight, even if they are reconstructed differently

→ Can be thought of as an "averaging" step

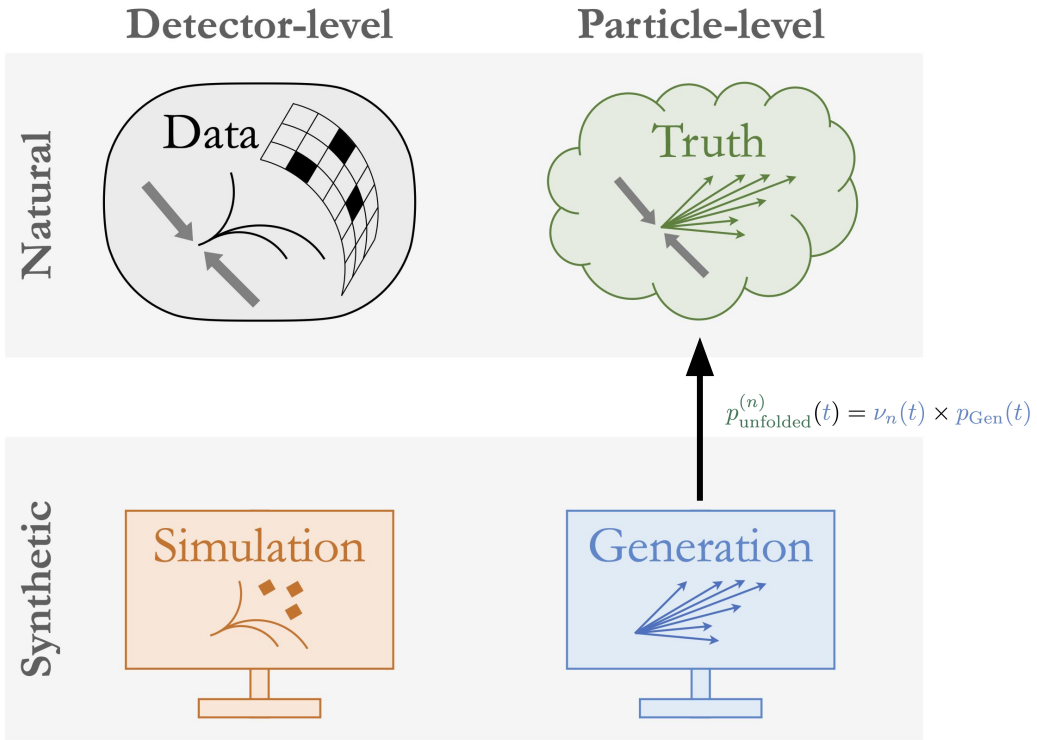
$$p^{(n)}_{\text{unfolded}}(t) = \nu_n(t) \times p_{\text{Gen}}(t)$$



Each iteration of MultiFold consists of 2 reweighting stages.

- Iterate this process multiple times to find a function that can **convert truth-level MC to unfolded data.**
- Neural networks are well-suited to this job: can process variable-length, high-dimensional inputs, and can learn a reweighting function by training a classifier & reinterpreting its outputs:

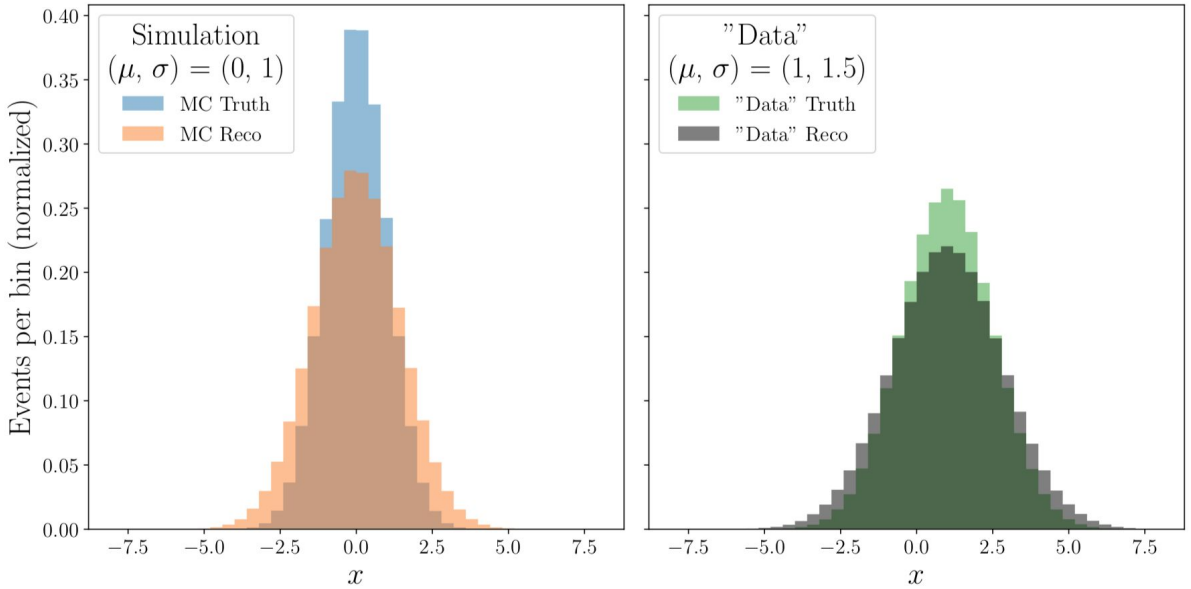
$$\frac{f^*(\vec{x})}{1-f^*(\vec{x})} \propto \frac{p(\vec{x}|\text{dataset 1})}{p(\vec{x}|\text{dataset 2})}$$



OmniFold reweights truth-level MC to estimate "truth-level" data.

Let's say we have two datasets: our MC simulation $\mathcal{N}(\mu=0, \sigma=1)$ and our data $\mathcal{N}(\mu=1, \sigma=1.5)$.

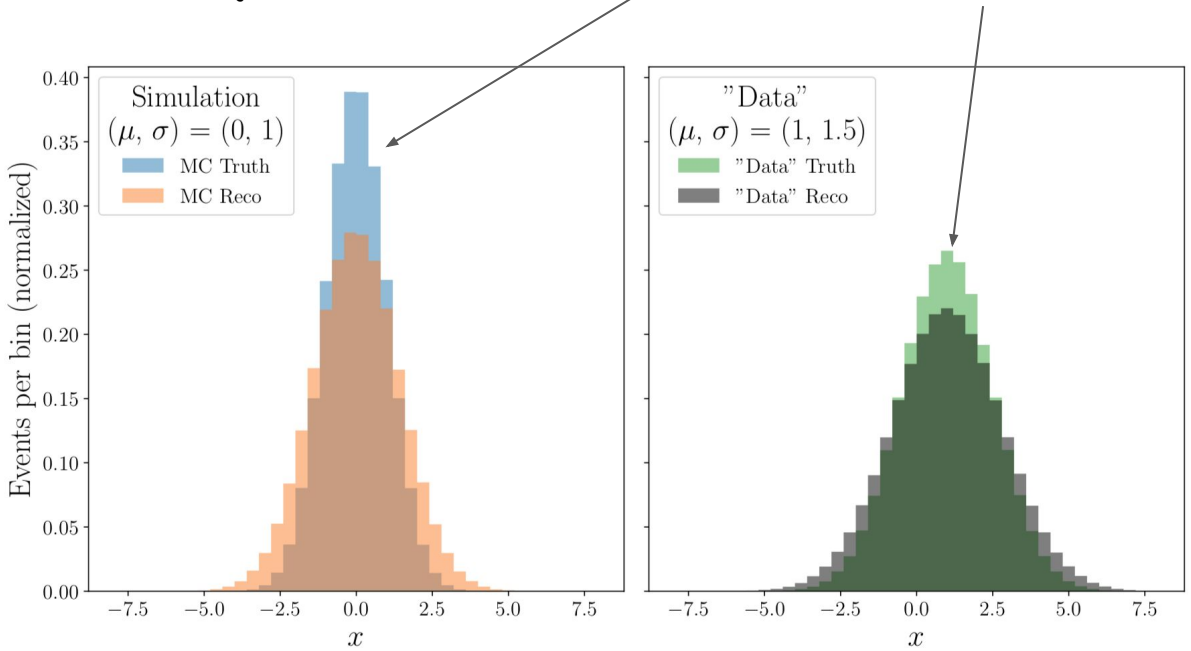
Due to detector effects, the distributions will look different at **truth-level** (sharper peaks) vs. at **reco-level.** (wider peaks)



OmniFold reweights truth-level MC to estimate "truth-level" data.

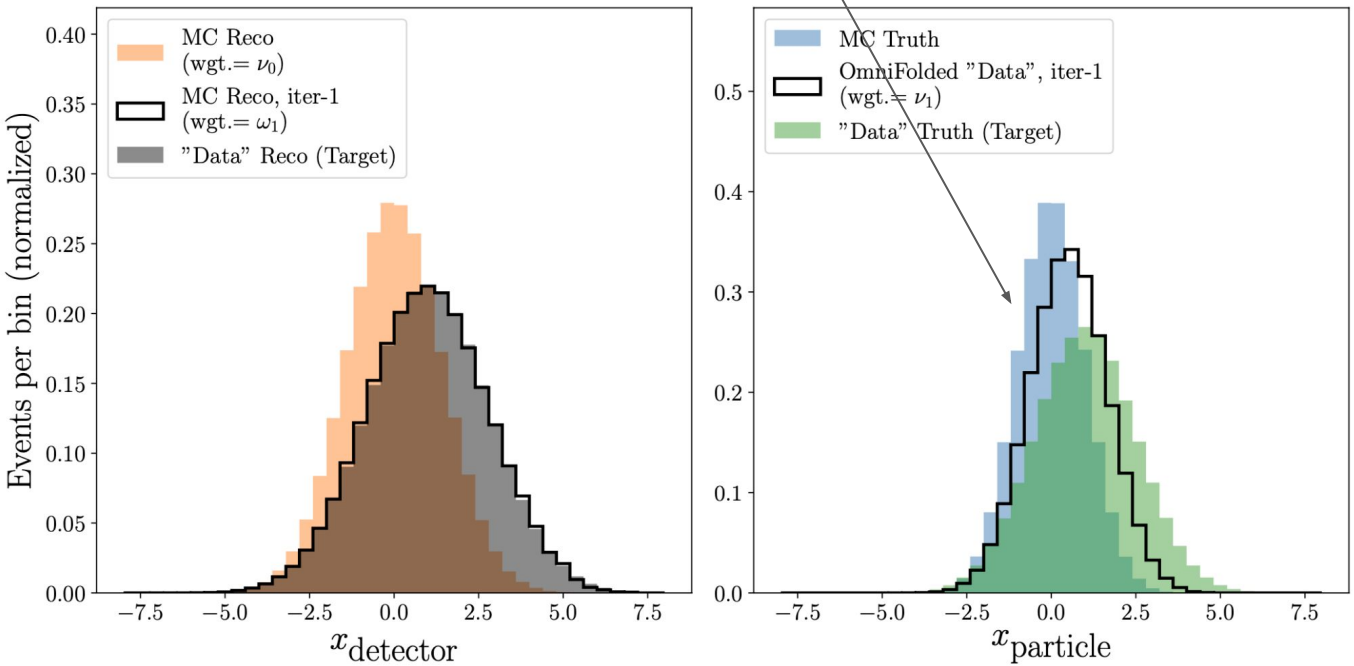
In reality, we'll never have access to the truth-level data – the best we get is the **reco-level data**.

Our goal, then, is to learn a way to transform our **truth-level MC** into **truth-level data**.



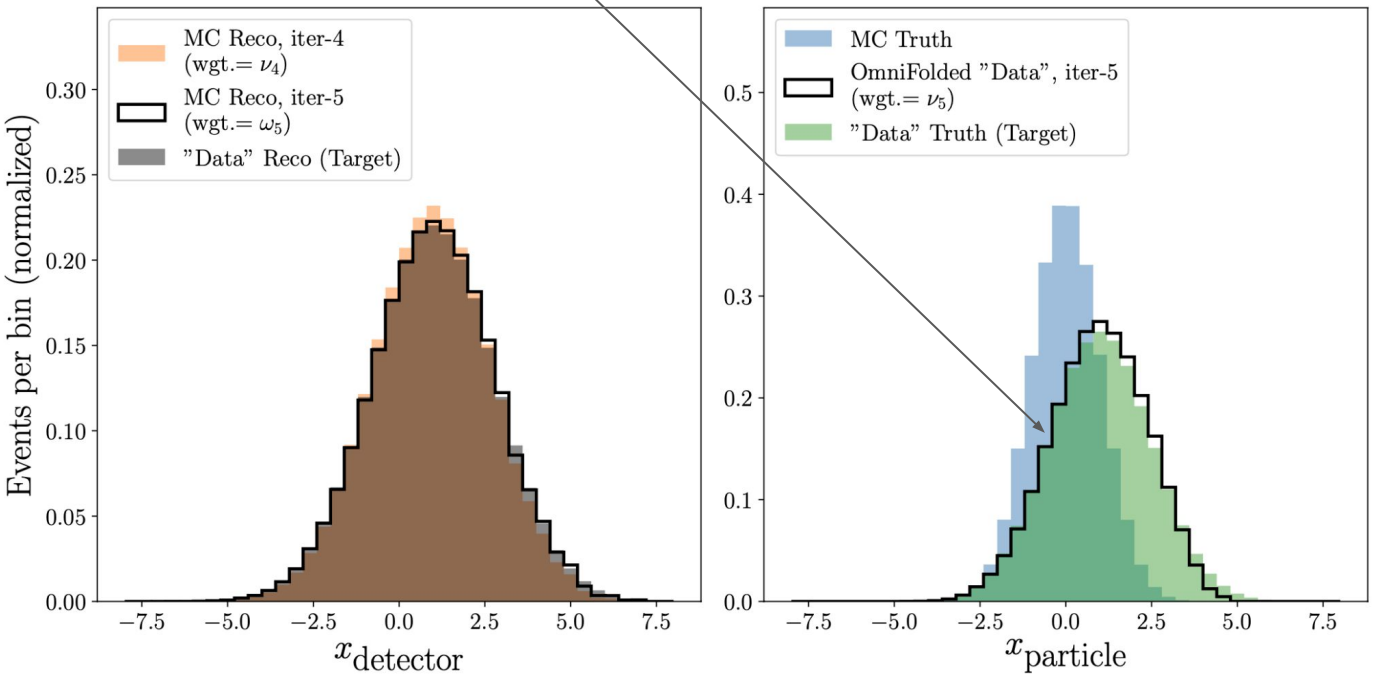
OmniFold reweights truth-level MC to estimate "truth-level" data.

After one iteration of our method, the truth-level MC is **reweighted** to look more like truth-level data.



OmniFold reweights truth-level MC to estimate "truth-level" data.

After five iterations of our method, the **reweighted** truth-level MC is closely aligned with truth-level data.



Choosing a better classifier

Mostly summarizing the results of Rizvi, Pettee, & Nachman (*JHEP*, 2023) [arXiv:2305.10500],
but please also look at our citations for many other related studies we are building upon!

The “likelihood ratio trick” is not limited to the binary cross-entropy loss function.

In fact, there is an entire class of loss functionals that have this property.

$$L[f] = -\int dx \left(p(x|\theta_A)\, A(f(x)) + p(x|\theta_B)\, B(f(x)) \right)$$

Probability density A

Rescaling function

Probability density B

Rescaling function

$$L[f] = -\int dx \left(p(x|\theta_A)\, A(f(x)) + p(x|\theta_B)\, B(f(x)) \right)$$

Probability density A

Rescaling function

Probability density B

Rescaling function

$$\frac{\delta L}{\delta f} = -\frac{\partial}{\partial f}\Big(p(x \mid \theta_0) A(f(x)) + p(x \mid \theta_1) B(f(x)) \Big) = 0 \iff -\frac{B'(f(x))}{A'(f(x))} = \frac{p(x \mid \theta_0)}{p(x \mid \theta_1)} = \mathcal{L}(x)$$

Must be a monotonic rescaling of *f*

$$L[f] = -\int dx \left(p(x|\theta_A)\, A(f(x)) + p(x|\theta_B)\; B(f(x)) \right)$$

Loss Name	$A(f)$	$B(f)$
Binary Cross-Entropy	$\ln(f)$	$\ln(1-f)$
Mean Squared Error	$-(1-f)^2$	$-f^2$
Maximum Likelihood Classifier	$\ln(f)$	$1-f$
Square Root	$-\frac{1}{\sqrt{f}}$	$-\sqrt{f}$

We can even go beyond just rescaling the classifier function by modifying the neural network activation functions.

We can even go beyond just rescaling the classifier function by modifying the neural network activation functions.

$$\sigma(z)$$

Instead of sigmoid activation... choose another function with range between $(0,1)$!

$$\frac{1}{\pi}\left(\arctan z + \frac{\pi}{2}\right) \qquad \Phi(z) = \int_0^z \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\,\mathrm{d}x$$

$$\mathrm{ReLU}(z)$$

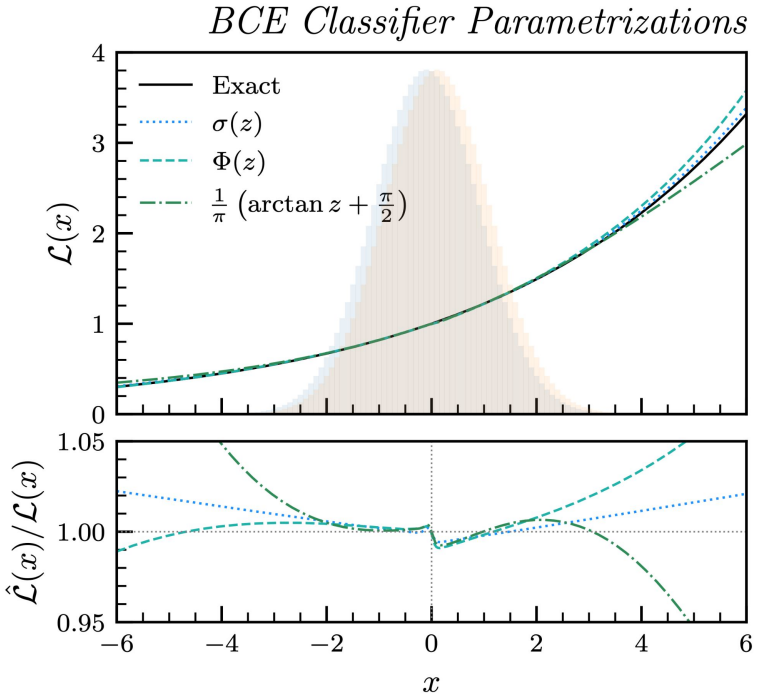
Instead of ReLU activation... choose another function with range between $(0,\infty)$!

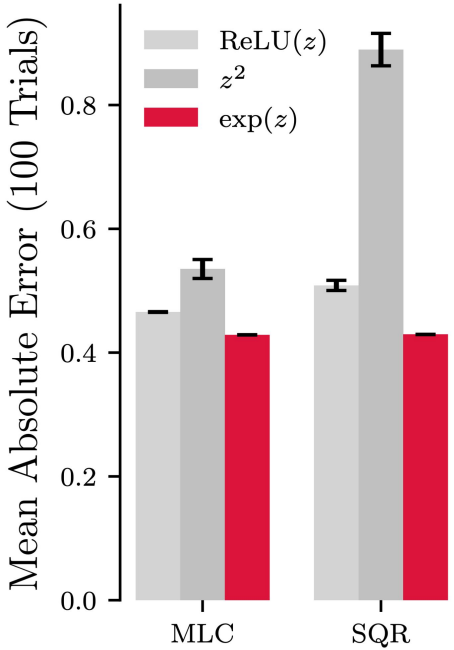
$$z^2 \qquad e^z$$

We find that:

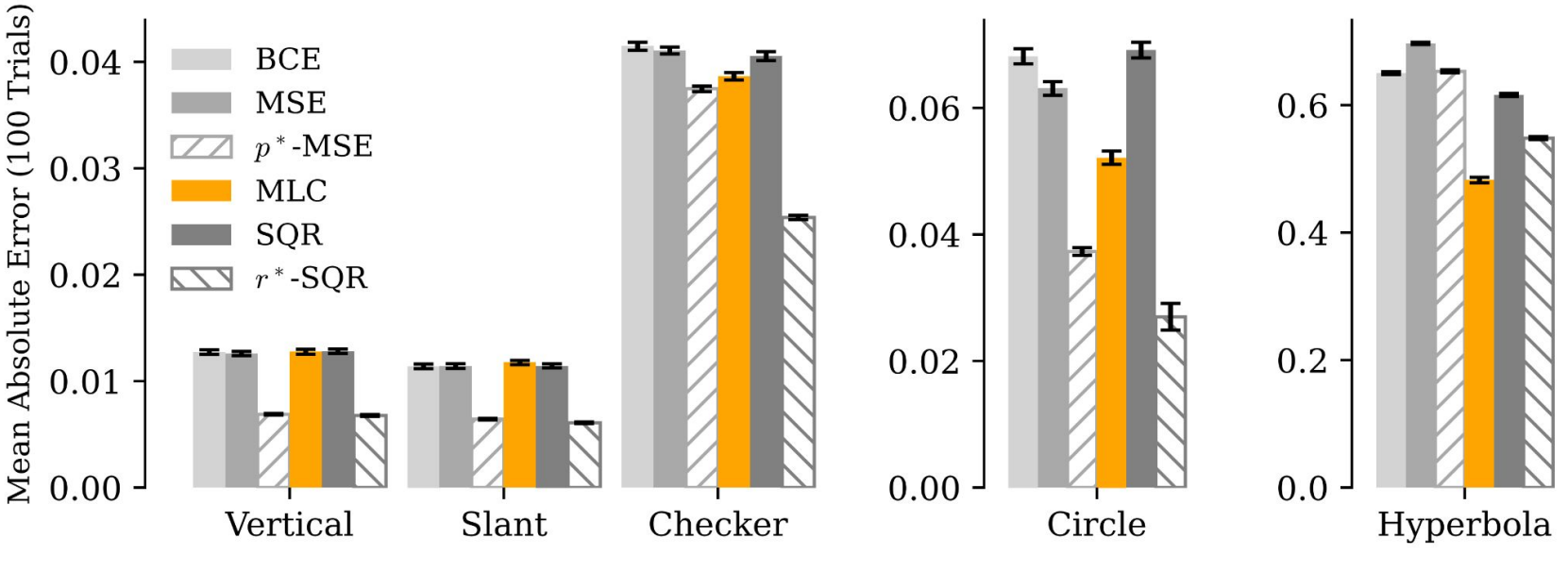
1. The choice of activation function is often crucial
2. Binary cross-entropy is not always the best choice of loss function
3. The best choice can often be found by scanning over a family of losses

1. The choice of activation function is often crucial



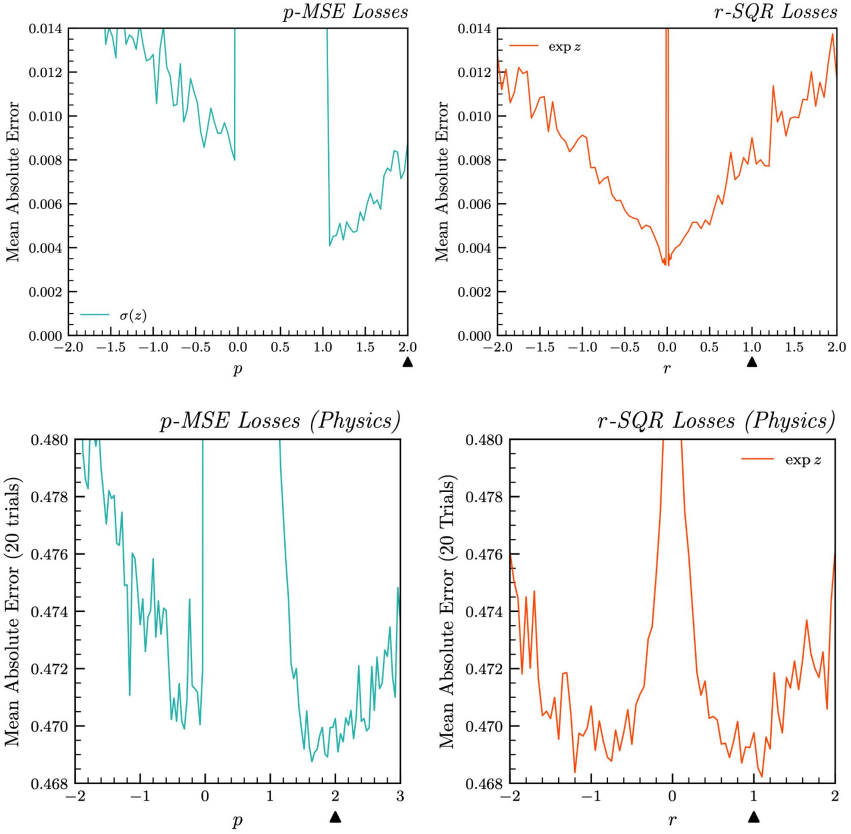


2. Binary cross–entropy is not always the best choice of loss function

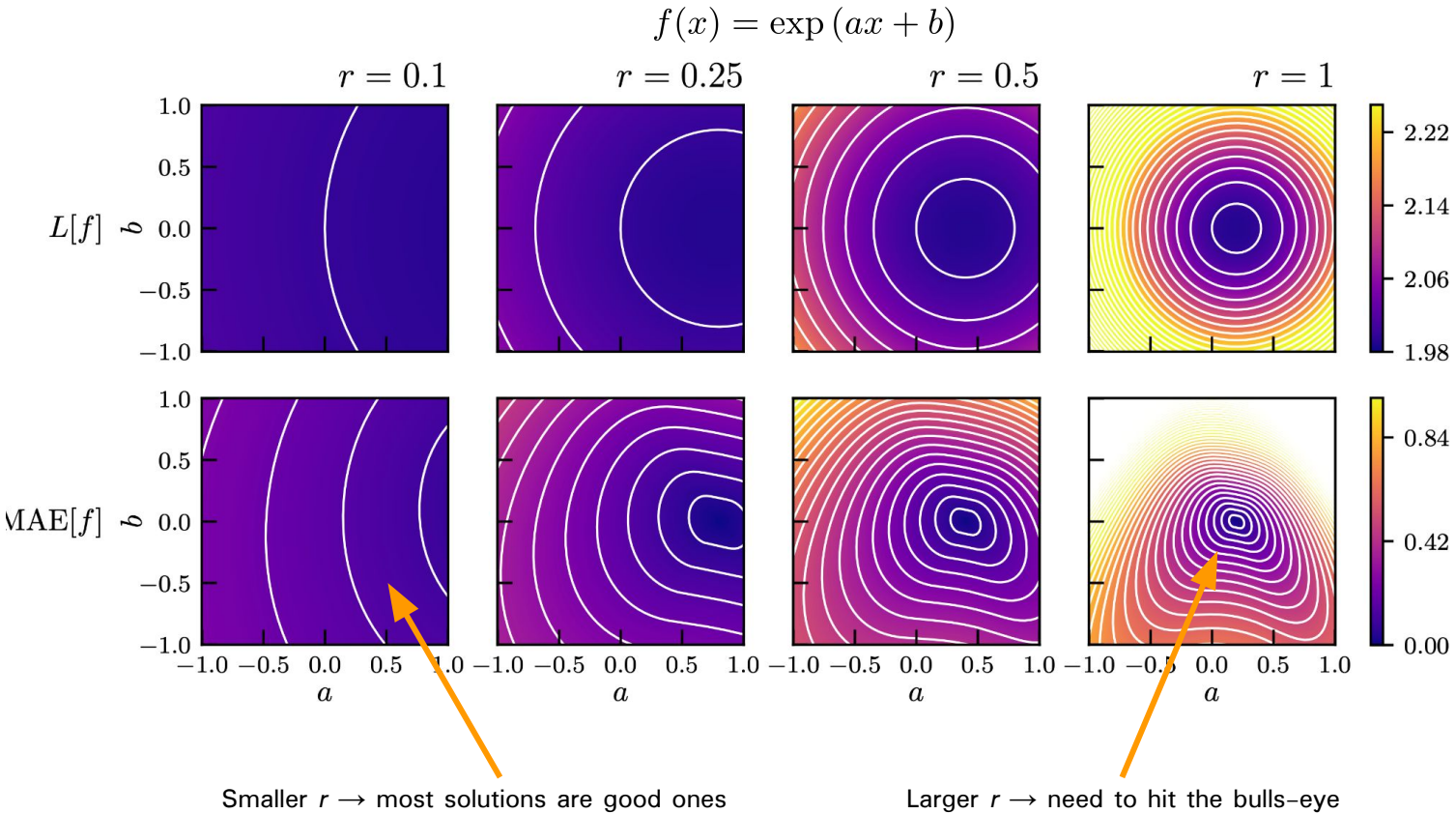


3. The best choice can often be found by scanning over a family of losses

	$A(f)$	$B(f)$
p-MSE	$-(1-f)^p$	$-f^p$
r-SQR	$-f^{-\frac{r}{2}}$	$-f^{\frac{r}{2}}$



3. The best choice can often be found by scanning over a family of losses



Summary

Summary

In short, classifiers are lightweight & effective tools for removing detector effects from data.

Classification is an **easier task** for neural networks than density estimation, and classifiers have already enabled **unbinned, high-dimensional measurements of real physics datasets.**

This is encouraging for imagining more flexible and useful unfolding measurements going forward:

- Choosing your own bins & modifying the phase space, thanks to unbinned measurements
- Capturing complex and even unforeseen properties of events, thanks to high-dimensional unfolding

(See Fernando's talk/tutorial, and my talk on Thursday for new results of 24-dimensional OmniFold on the ATLAS Run 2 dataset!)

Thanks!