Unfolding of High Dimensional Observables: A Statistical Perspective

Erik A. Bensen

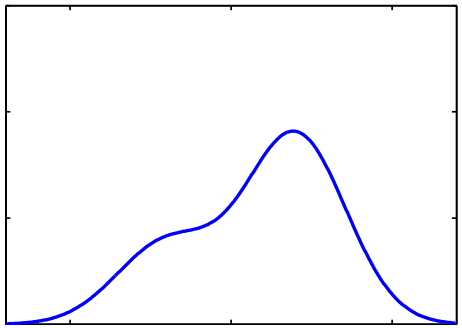
Department of Statistics and Data Science,
Statistical Methods for the Physical Sciences Research Center,
Carnegie Mellon University

Workshop on High Dimensional Problems for Statistical Methods in Fundamental Physics Data Analyses

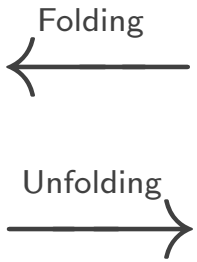
Banff, AB, Canada

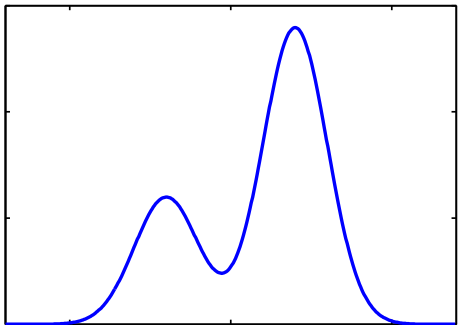
June 9, 2026

The unfolding problem



Smeared spectrum, $g(s)$
(reco-level)





True spectrum, $f(t)$
(part-level)

Mathematically, these are intensity functions of underlying Poisson point processes

$$g(s) = \int k(s,t) f(t)\, \mathrm{d}t,$$

where the smearing kernel k represents the response of the detector

$$k(s,t) = p(Y = s | X = t, X \text{ observed}) P(X \text{ observed} | X = t),$$

where X is a true event and Y the corresponding smeared event

Task: Infer the true spectrum f given smeared observations from g

Machine learning-based unfolding

Recent years have seen the introduction of various machine learning-based unfolding methods (see Huetsch et al. (2025) for a review)

Two main approaches:

1. OmniFold (Andreassen et al., 2020, 2021): iteratively reweight particle-level MC events using classifier-based density ratios
2. Generative unfolding (e.g., Bellagente et al., 2020; Backes et al., 2024): Train a generative ML model to sample from $p(X = t|Y = s)$; iterate to reduce dependence on $p^{\mathrm{MC}}(X = t)$

These provide several advantages:

- Unbinned event-level unfolding (no need to discretize the problem with histograms)
- Can handle (moderately) high-dimensional phase spaces
- No need to estimate the response matrix, K

Machine learning-based unfolding

Let us consider the unbinned unfolding problem:

$$g(s) = \int k(s,t) f(t)\,\mathrm{d}t = \int p(s|t) f(t)\,\mathrm{d}t,$$

where, for simplicity, we take f and g to be probability densities and assume $\int k(s,t)\,\mathrm{d}s = 1$ (i.e., ignore normalization and efficiency)

Our task is to find f

Let's assume, for now, that we get to observe g

Then we could find f by minimizing the KL divergence

$$\mathrm{KL}\left(g, \int p(\cdot|t) f(t) dt\right) = \int g(s) \log \frac{g(s)}{\int p(s|t) f(t) dt} ds,$$

which is equivalent to maximizing the (population) log-likelihood of f

$$\ell(f) = \int g(s) \log \left(\int p(s|t) f(t) dt\right) ds$$

Unbinned Expectation-Maximization iteration

The current ML-based unfolding methods can be understood as Expectation-Maximization (EM) iterations for maximizing the log-likelihood $\ell(f)$

The Q-function (expectation step) of the EM iteration is

$$Q(f, f^{(k)}) = \int g(s) \int p(t|s, f^{(k)}) \log[p(s|t)f(t)]dtds,$$

which can be rewritten as

$$Q(f, f^{(k)}) = \int g(s) \int \frac{p(s|t)f^{(k)}(t)}{\int p(s|t')f^{(k)}(t')dt'} \log[p(s|t)f(t)]dtds$$

Then the EM iteration proceeds by computing

$$f^{(k+1)} = \arg\max_f Q(f, f^{(k)})$$

subject to the constraint $\int f(t)dt = 1$ (maximization step)

Unbinned Expectation-Maximization iteration

The maximizer can be found in closed form and is given by

$$f^{(k+1)}(t) = f^{(k)}(t) \int \frac{p(s|t)g(s)}{\int p(s|t')f^{(k)}(t')dt'} ds \tag{1}$$

It is useful to compare this with D'Agostini iteration (with $\sum_{i=1}^{n} K_{ij} = 1$):

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} \sum_{i=1}^{n} \frac{K_{ij} y_i}{\sum_{l=1}^{p} K_{il} \lambda_l^{(k)}} \tag{2}$$

We see that (2) is a binned version of (1) with the population-level smeared histogram $g(s)$ replaced by the empirical bin counts y_i

Iteration (1) first appeared in Kondor (1983) and its theoretical properties were later studied in Mülthei and Schorr (1987a,b, 1989); Mülthei (1992)

OmniFold

OmniFold (Andreassen et al., 2020, 2021) rewrites the previous unbinned EM iteration using density ratios and learns these density ratios from data using an ML classifier

Specifically, assume access to MC samples $\{X_i', Y_i'\}_{i=1}^n$ from density $q(t,s) = p(s|t)q(t)$, where $p(s|t) = k(s,t)$ is the same smearing kernel as in the real observations $\{X_i, Y_i\}_{i=1}^n$

Then the unbinned EM update can be rewritten as

$$\frac{f^{(k+1)}(t)}{q(t)} = \frac{f^{(k)}(t)}{q(t)} \int \frac{p(s|t)g(s)}{\int p(s|t')q(t')\frac{f^{(k)}(t')}{q(t')}dt'} ds$$

Denoting the density ratio $\nu^{(k)}(t) = \frac{f^{(k)}(t)}{q(t)}$, this becomes

$$\nu^{(k+1)}(t) = \nu^{(k)}(t) \int \frac{p(s|t)g(s)}{\int \nu^{(k)}(t')q(t',s)dt'} ds$$

OmniFold

The OmniFold EM update

$$\nu^{(k+1)}(t) = \nu^{(k)}(t)\frac{1}{q(t)}\int \frac{g(s)}{\int \nu^{(k)}(t')q(t',s)dt'}p(s|t)q(t)ds$$

can be written as two density ratio estimation problems, leading to the following two-step algorithm:

1. Estimate detector-level (reco-level) density ratio:
   $r^{(k)}(s) = \frac{g(s)}{\tilde{q}^{(k)}(s)}$, where $\tilde{q}^{(k)}(s) = \int \nu^{(k)}(t)q(t,s)dt$
2. Estimate particle-level density ratio and update $\nu^{(k)}$:
   $\nu^{(k+1)}(t) = \nu^{(k)}(t)\frac{\tilde{q}^{(k)}(t)}{q(t)}$, where $\tilde{q}^{(k)}(t) = \int r^{(k)}(s)q(t,s)ds$

The sample-level implementation of this estimates the density ratios using classifiers (usually neural networks) and applies $r^{(k)}(s)$ and $\nu^{(k)}(t)$ iteratively as weights on the MC sample $\{X'_i, Y'_i\}_{i=1}^n$

After K iterations, the end result is a MC sample reweighted using $\nu^{(K)}(t)$

Aside: Classifiers and Density Ratios

Let:

$$\mathcal{C}_1: \quad \boldsymbol{z}_1^1, \boldsymbol{z}_2^1, \ldots, \boldsymbol{z}_n^1 \overset{\text{iid}}{\sim} p_1$$

$$\mathcal{C}_2: \quad \boldsymbol{z}_1^2, \boldsymbol{z}_2^2, \ldots, \boldsymbol{z}_n^2 \overset{\text{iid}}{\sim} p_2$$

Train a classifier $h(\boldsymbol{z})$ to separate $\mathcal{C}_1$ from $\mathcal{C}_2$ by minimizing the binary cross-entropy loss; then the classifier will asymptotically give:

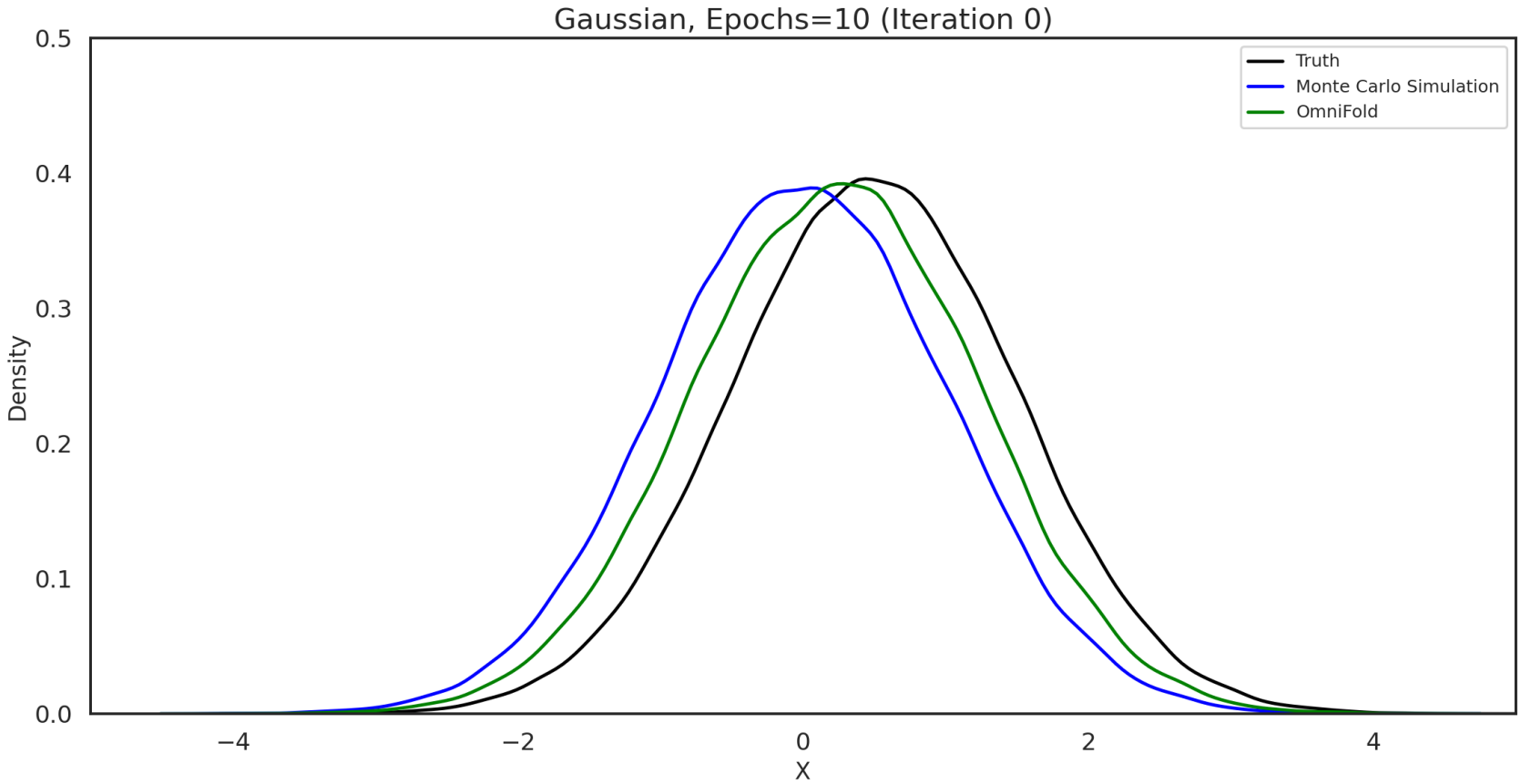
$$h(\boldsymbol{z}) = P(\mathcal{C}_1 \mid \boldsymbol{z}) = \frac{p(\boldsymbol{z} \mid \mathcal{C}_1)P(\mathcal{C}_1)}{p(\boldsymbol{z} \mid \mathcal{C}_1)P(\mathcal{C}_1) + p(\boldsymbol{z} \mid \mathcal{C}_2)P(\mathcal{C}_2)} = \frac{\frac{p(\boldsymbol{z}|\mathcal{C}_1)}{p(\boldsymbol{z}|\mathcal{C}_2)}}{\frac{p(\boldsymbol{z}|\mathcal{C}_1)}{p(\boldsymbol{z}|\mathcal{C}_2)} + 1} = \frac{\frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})}}{\frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})} + 1}$$

Solving this for $\psi(\boldsymbol{z}) = \frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})}$ gives $\psi(\boldsymbol{z}) = \frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})} = \frac{h(\boldsymbol{z})}{1-h(\boldsymbol{z})} = \text{odds}(h(\boldsymbol{z}))$

The classifier $h(\boldsymbol{z})$ is learning the density ratio $p_1(\boldsymbol{z})/p_2(\boldsymbol{z})$!

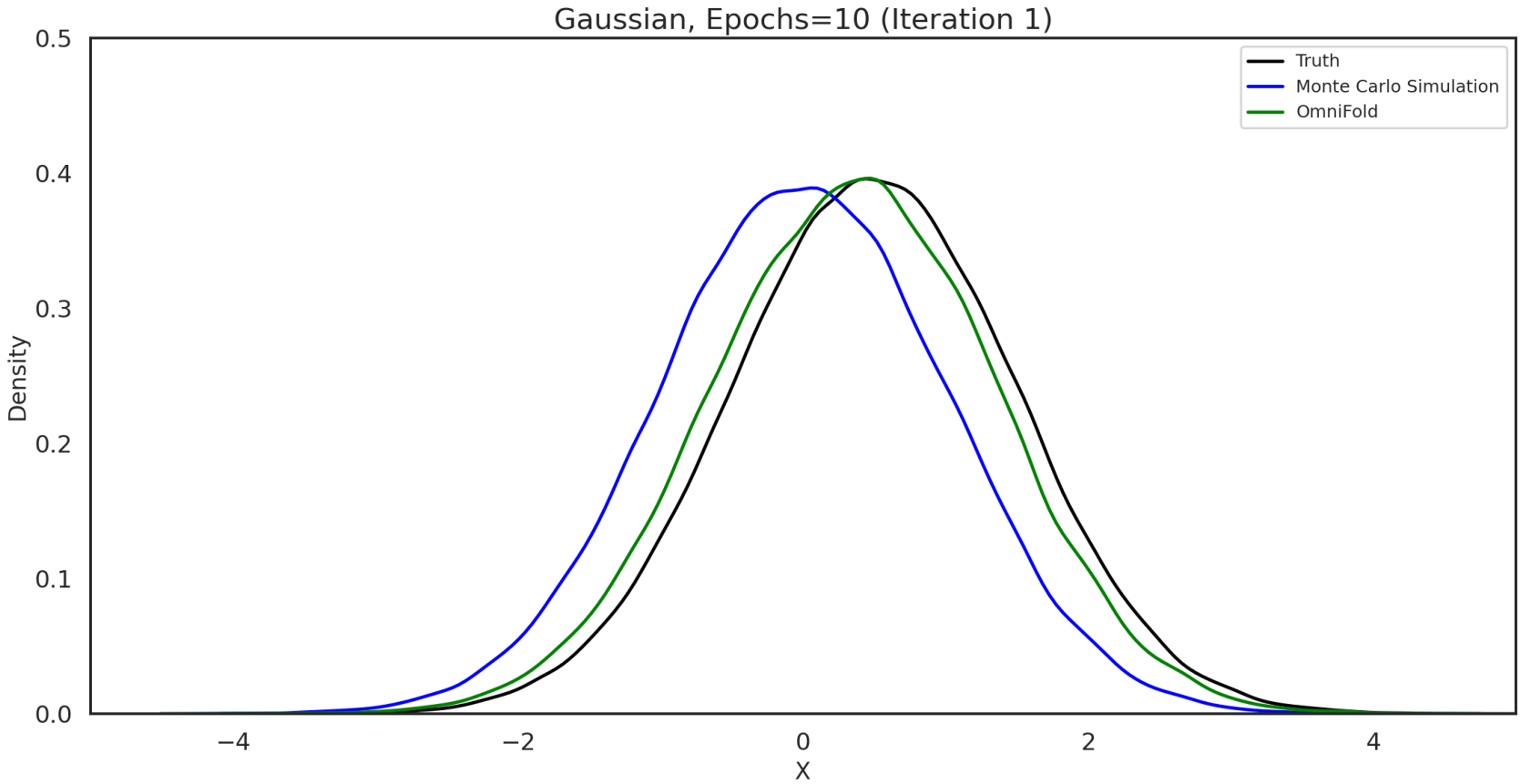
- When $h(\boldsymbol{z})$ is a (deep) neural network, this works amazingly well for high-dimensional $\boldsymbol{z}$
- This can be used to learn likelihood functions (Walchessen et al., 2024) and likelihood ratios (Cranmer et al., 2015), to perform two-sample testing (Chakravarti et al., 2023), for transfer learning (Manole et al., 2024) and many other tasks in high-dimensional spaces

OmniFold demo



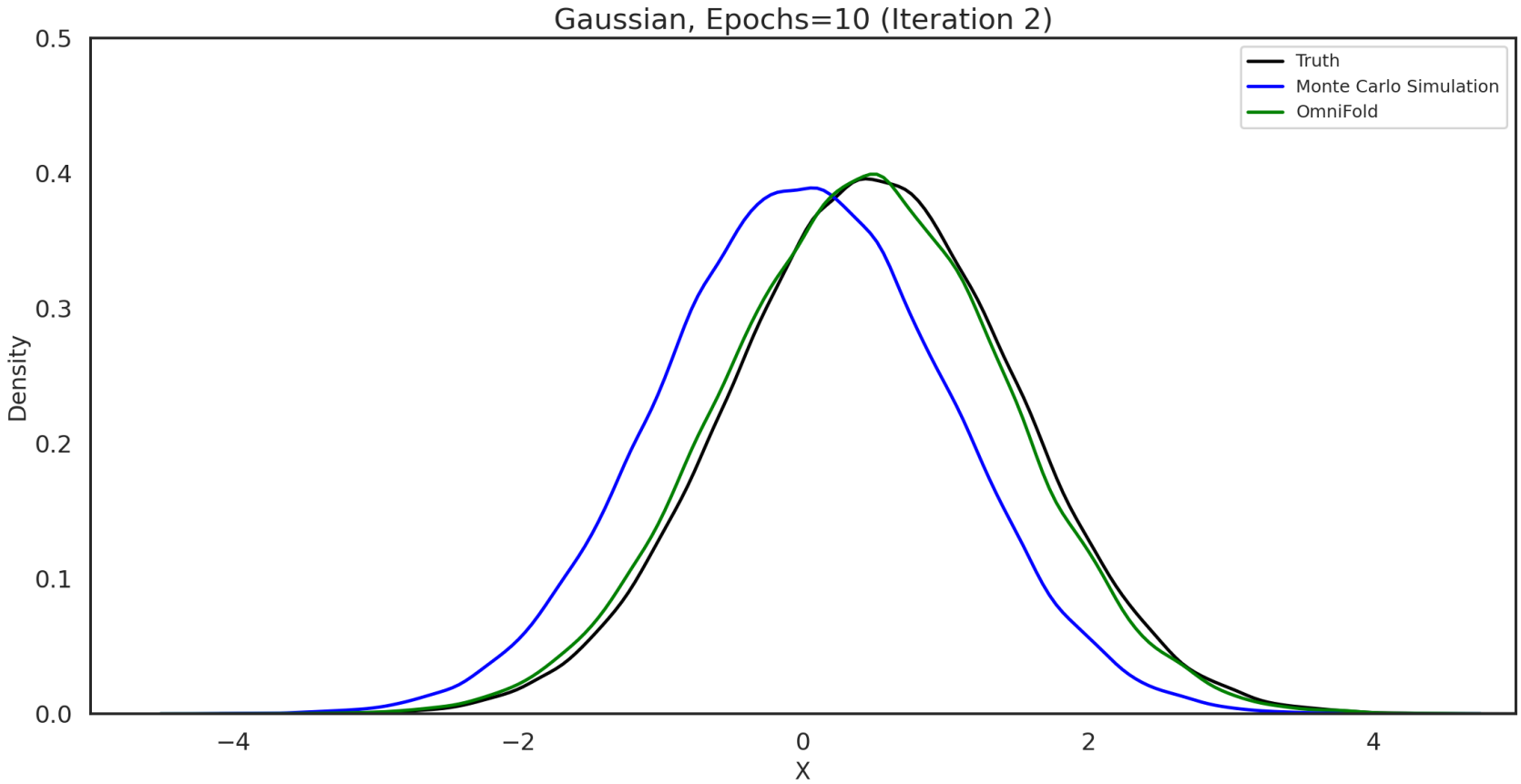
(Simulation by Richard Zhu)

OmniFold demo



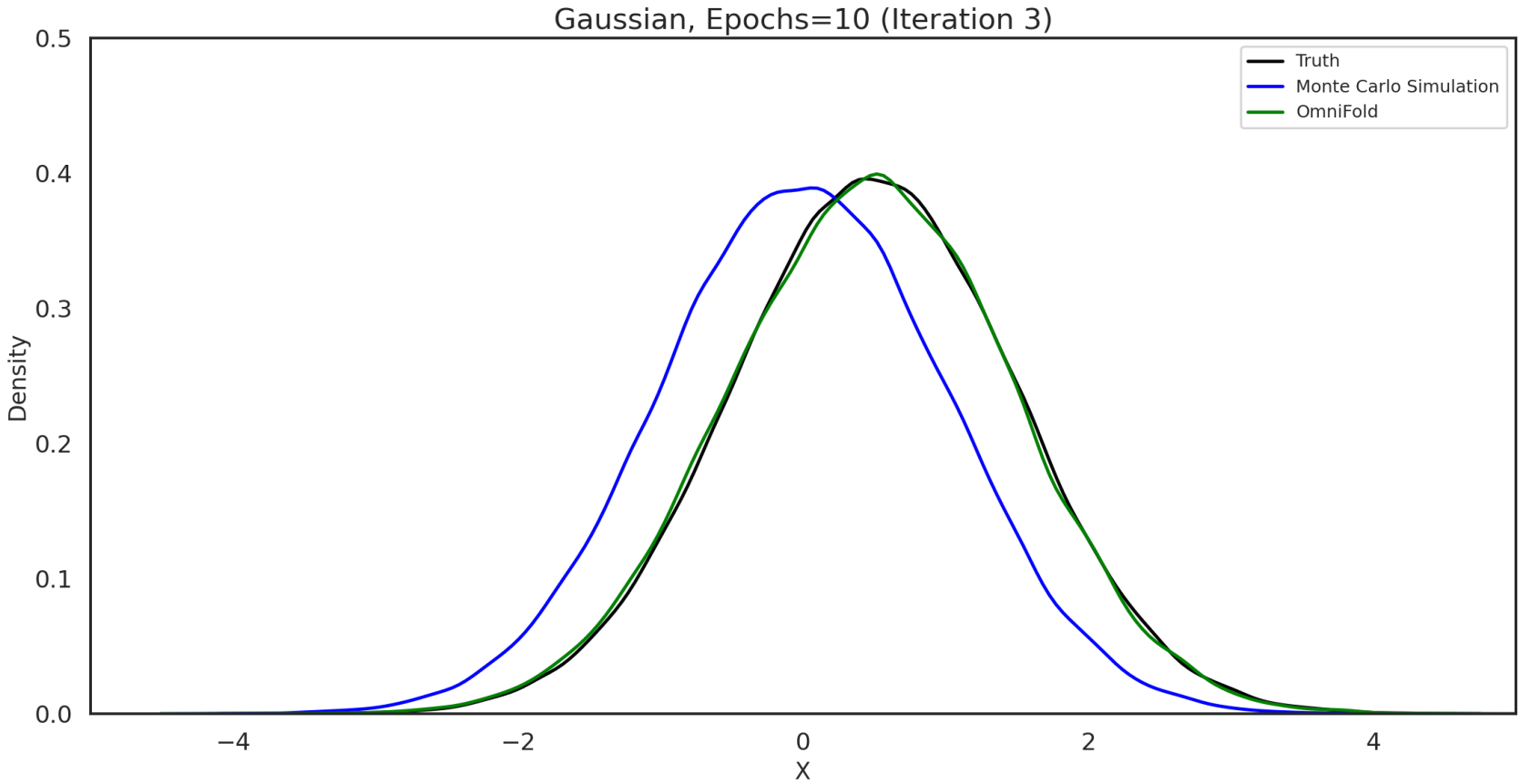
(Simulation by Richard Zhu)

OmniFold demo



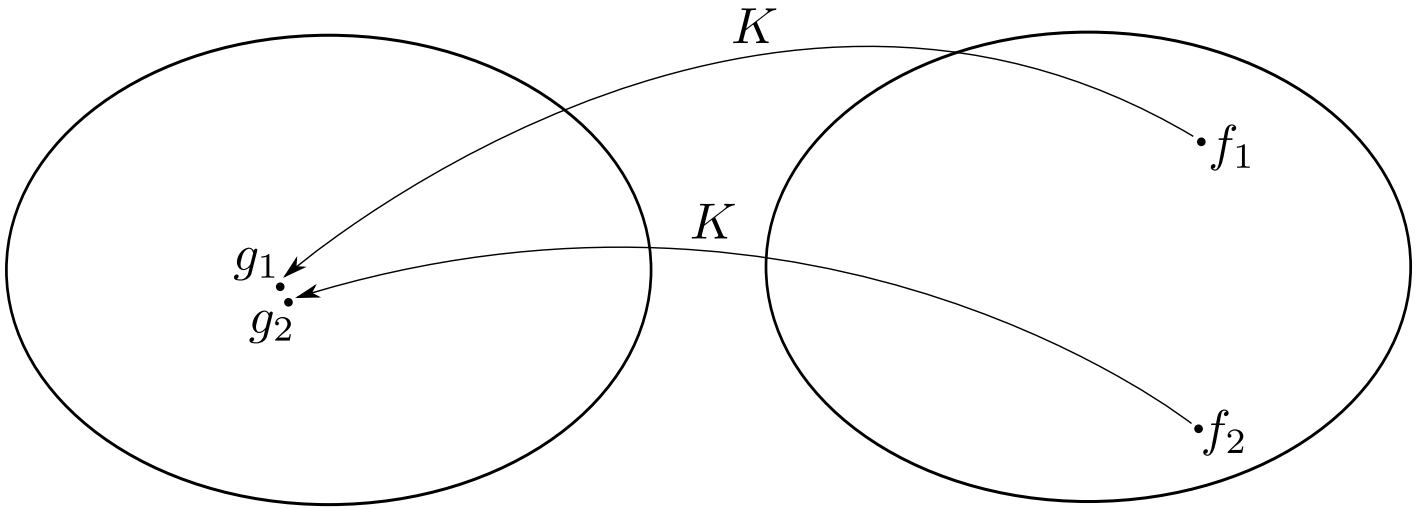
(Simulation by Richard Zhu)

OmniFold demo



(Simulation by Richard Zhu)

Unfolding is an ill-posed inverse problem



The forward folding mapping

$$g = K(f) = \int k(\cdot, t) f(t)\, \mathrm{d}t$$

is such that it can take inputs f_1 and f_2 that look very different into outputs g_1 and g_2 that are very similar

This means that, without further information, data collected in the g-space can only mildly constrain the solution in the f-space

Regularization in unfolding

Regularization complements the likelihood function with additional information about physically plausible solutions.
Classically, this often biases the solution towards a MC ansatz, $\boldsymbol{\lambda}^{\mathrm{MC}}$

- Main idea: bias $\uparrow$, variance $\downarrow$ $\Rightarrow$ MSE $\downarrow$

Tikhonov Regularization

- SVD and TUnfold
  Höcker and Kartvelishvili (1996);
  Schmitt (2012)

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}^p} (\boldsymbol{y}-\boldsymbol{K}\boldsymbol{\lambda})^{\mathrm{T}}\boldsymbol{C}^{-1}(\boldsymbol{y}-\boldsymbol{K}\boldsymbol{\lambda})+\delta P(\boldsymbol{\lambda})$$

$$P_{\mathrm{SVD}}(\boldsymbol{\lambda}) = \|\boldsymbol{L}(\boldsymbol{\lambda} \div \boldsymbol{\lambda}^{\mathrm{MC}})\|^2$$

$$P_{\mathrm{TUnfold}}(\boldsymbol{\lambda}) = \|\boldsymbol{L}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{\mathrm{MC}})\|^2$$

$P(\boldsymbol{\lambda})$ explicitly penalizes deviations from $\boldsymbol{\lambda}^{\mathrm{MC}}$

Iterative Methods

- D'Agostini (1995) iteration

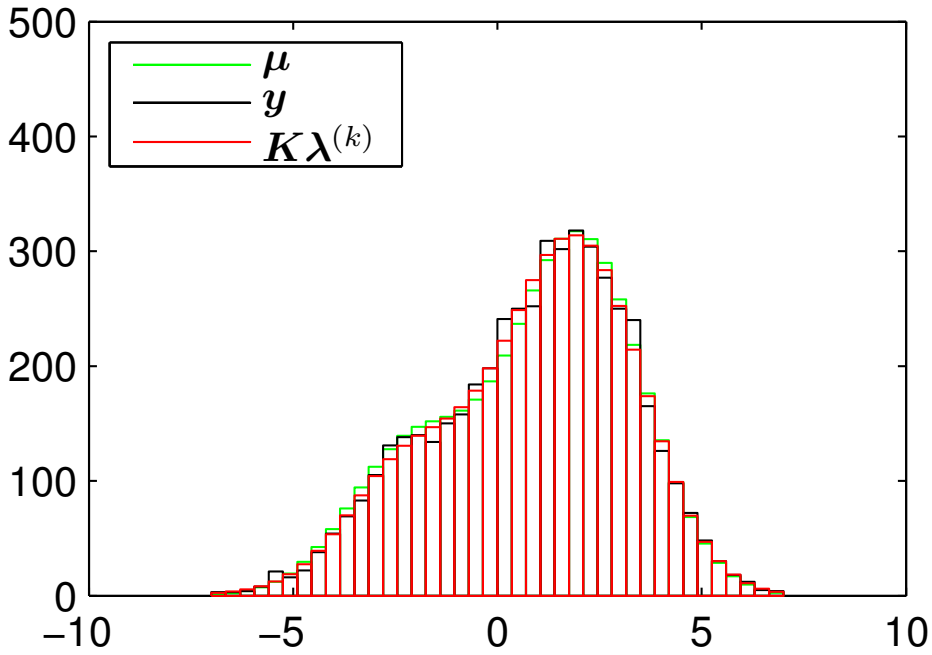
$$\lambda_j^{(t+1)} = \frac{\lambda_j^{(t)}}{\sum_{i=1}^n K_{i,j}} \sum_{i=1}^n \frac{K_{i,j} y_i}{\sum_{k=1}^p K_{i,k}\lambda_k^{(t)}}$$

  $\boldsymbol{\lambda}^{(0)} = \boldsymbol{\lambda}^{\mathrm{MC}}$ with early stopping prevents large deviations from $\boldsymbol{\lambda}^{\mathrm{MC}}$

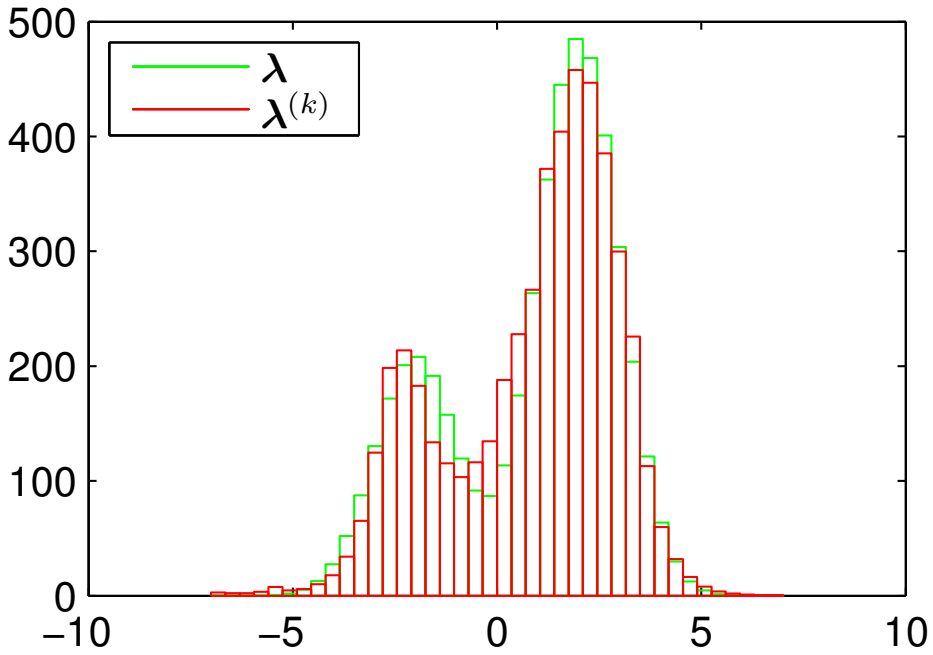
- OmniFold, uses early stopping, but is also *implicitly* regularized by model training and architecture

Question: How does ML implicit regularization affect solutions?
Not well understood $\rightarrow$ Unreliable uncertainty estimates

D'Agostini demo, $k = 100$

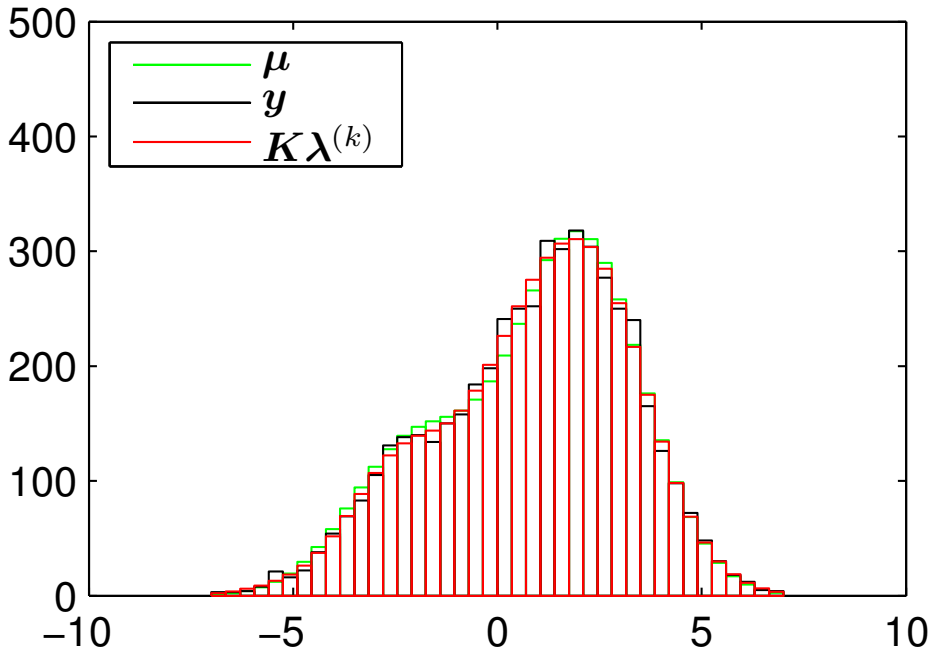


Smeared histogram

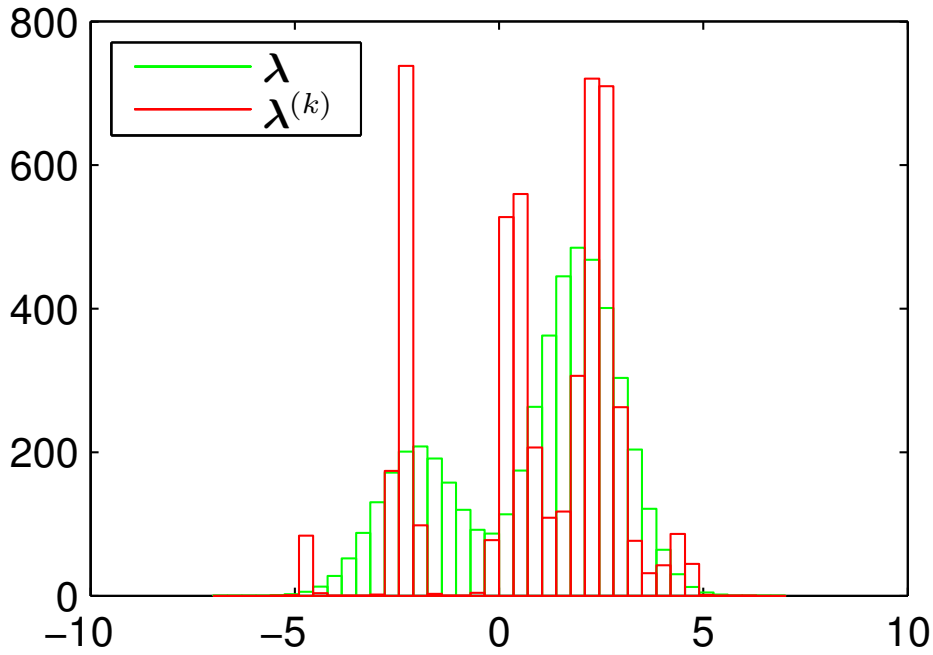


True histogram

D'Agostini demo, $k = 10000$



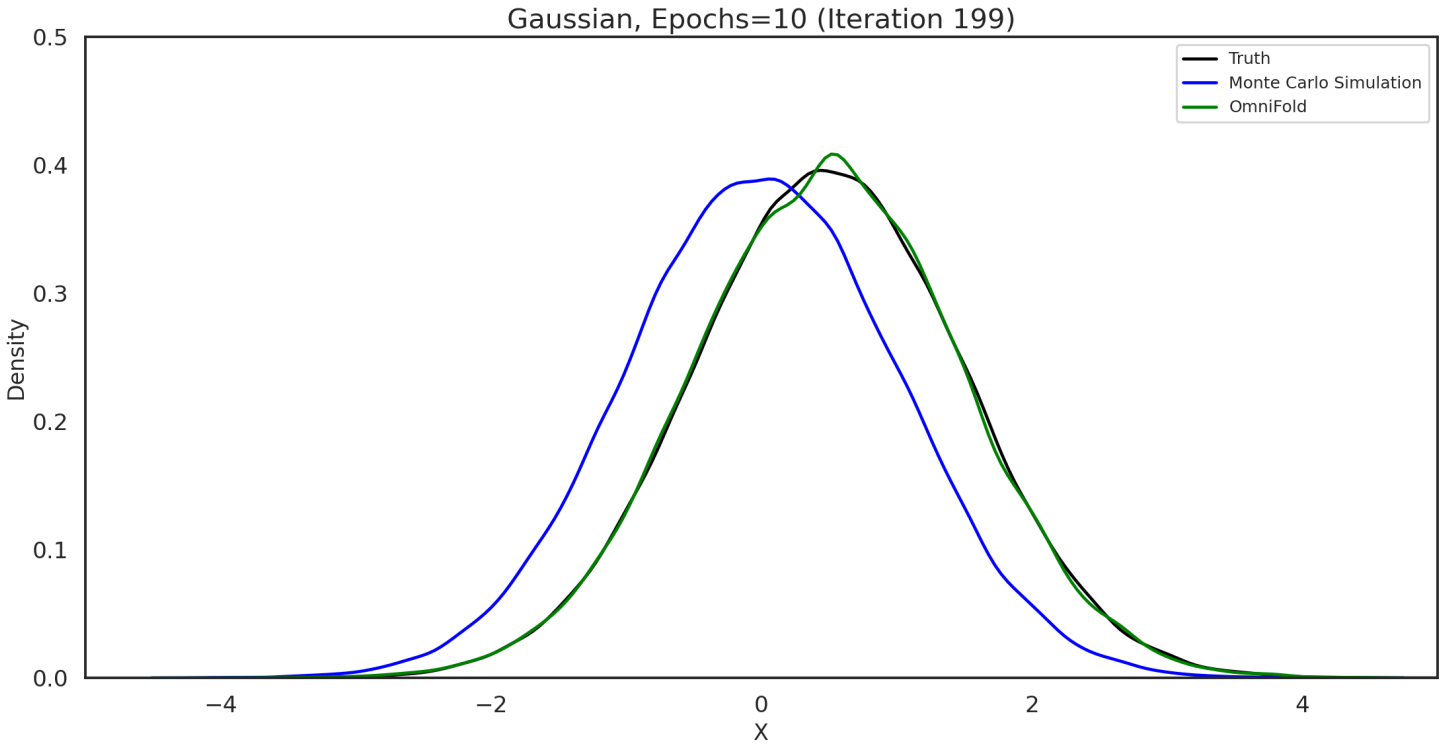
Smeared histogram



True histogram

Regularization in ML-based unfolding

In ML-based unfolding, regularization is not only controlled by the number of iterations, but also by the ML model training and architecture

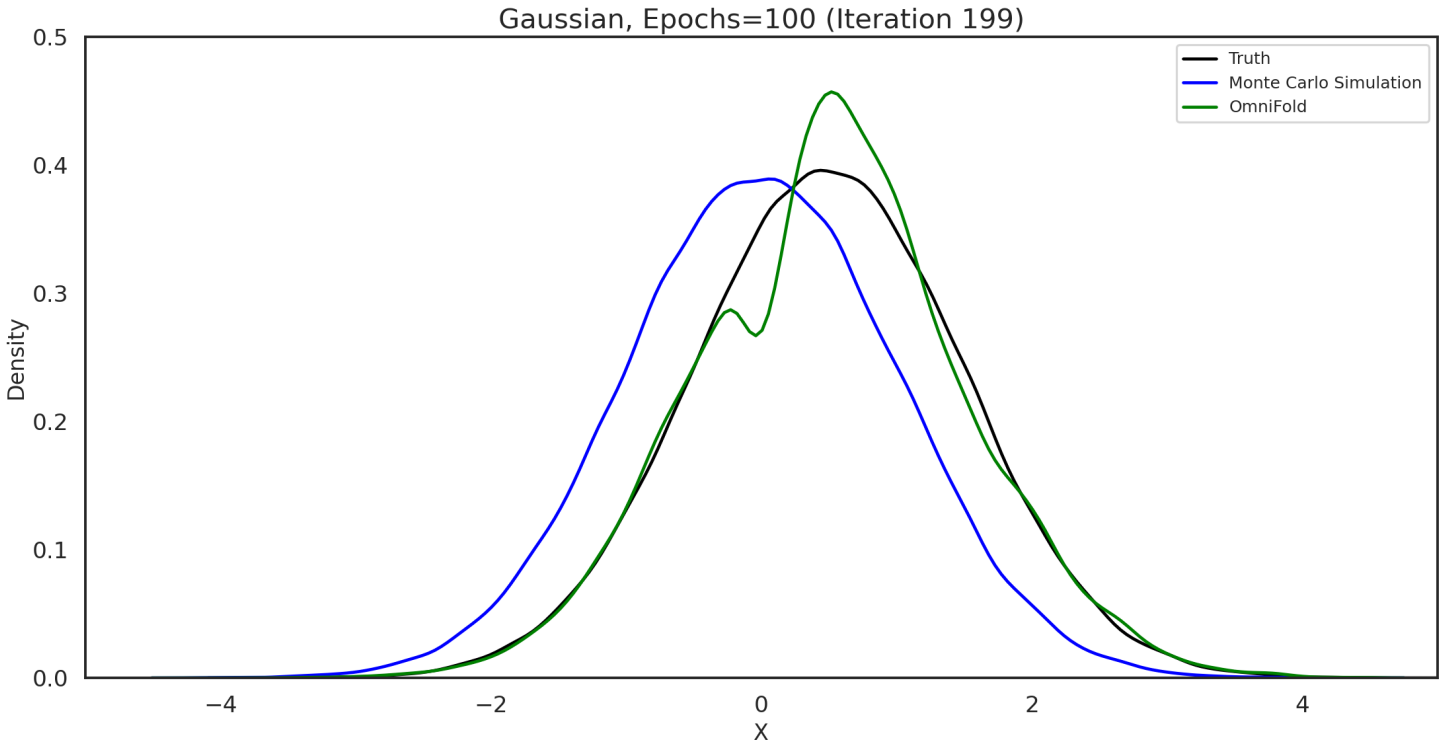


200 iterations of OmniFold with the neural nets trained for 10 epochs

(Simulation by Richard Zhu)

Regularization in ML-based unfolding

In ML-based unfolding, regularization is not only controlled by the number of iterations, but also by the ML model training and architecture

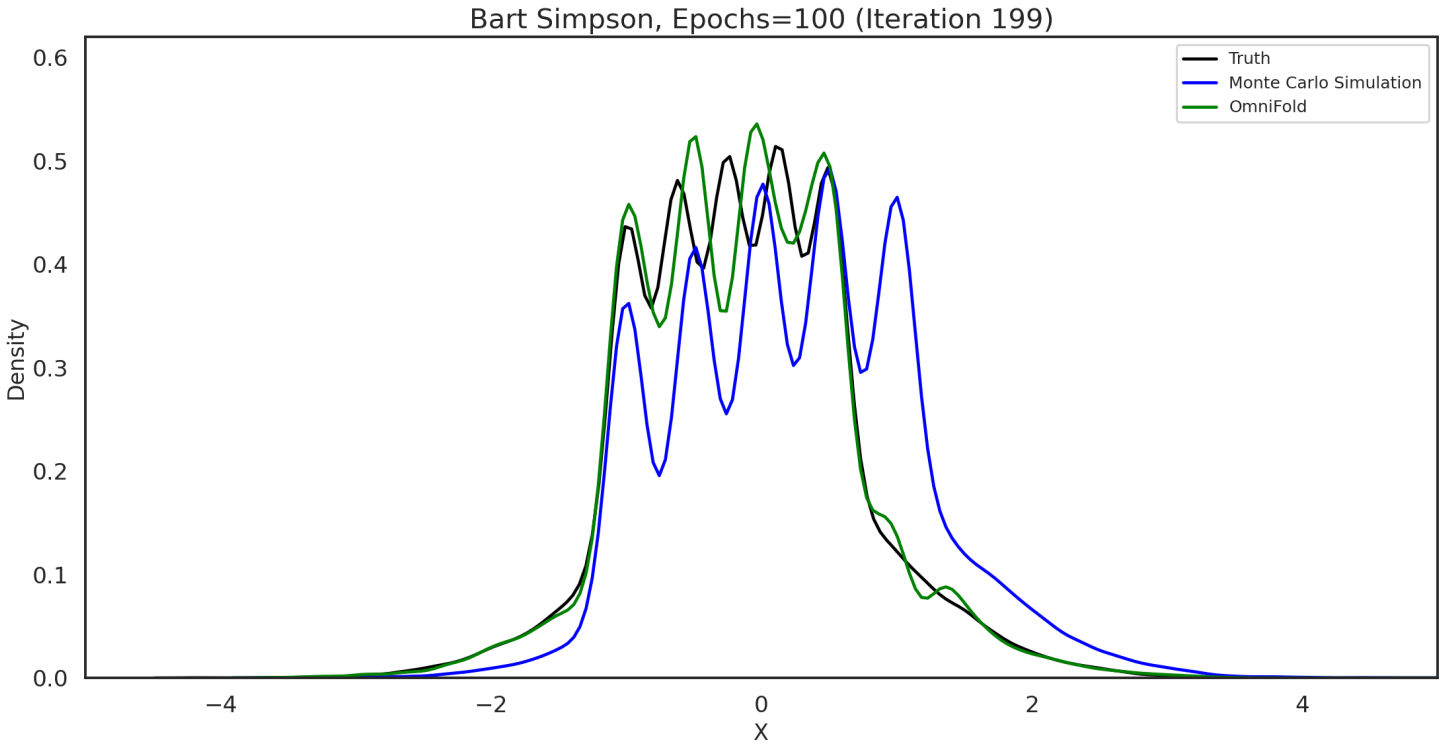


200 iterations of OmniFold with the neural nets trained for 100 epochs

(Simulation by Richard Zhu)

Regularization in ML-based unfolding

In ML-based unfolding, regularization is not only controlled by the number of iterations, but also by the ML model training and architecture



This particular combination of neural net architecture and training scheme struggles with features that are comparable to the smearing kernel size

(Simulation by Richard Zhu)

Recent Developments: AUSSIE (Ore and Plehn, 2026)

Main Idea: Adversary-free Unfolding SanS Iteration or Emulation (AUSSIE) changes the optimization step to directly target $f(t)$

OmniFold

Step 1: estimate observed ratio

$$r^{(k)}(s) = \frac{g(s)}{\int p(s|t)q(t)\nu^{(k)}(t)dt}$$

Step 2: fix t and adjust weights by integrating over smoothed observations, s

$$\nu^{(k+1)}(t) = \nu^{(k)}(t) \cdot \frac{\int p(s|t)r^{(k)}(s)ds}{q(t)}$$

Solution: learn ratios using classifier trick, converges by nature of EM algorithm

AUSSIE

Step 1: estimate observed ratio

$$r(s) = \frac{g(s)}{\int p(s|t)q(t)dt}$$

Step 2: fix s and directly solve the integral equation over the true observations t

$$r(s) = \int p(t|s)\nu(t)dt$$

Solution: can show $\nu(t)$ is a fixed point of the MLC loss. Train neural model to target the fixed point by minimizing $\|\nabla \mathrm{MLC}\|^2$

Discussion and conclusions

- ML-based unfolding enables high-dimensional unbinned unfolding
- Unfolding is an ill-posed inverse problem
  - need additional information to get any sensible solution
- Regularization supplements the likelihood by adding additional information about physically plausible solutions
  - Explicitly penalize deviations from MC estimate (Tikhonov Style)
  - Prevent solution from "moving too far" with early stopping (D'Agostini)
- ML training and architecture *implicitly* regularizes the solution
  - This is poorly understood
- How to obtain rigorous uncertainty quantification for ML-based unfolding? How trustworthy are the current uncertainty estimates?

References I

A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, and J. Thaler. Omnifold: A method to simultaneously unfold all observables. *Physics Review Letters*, 124: 182001, 2020. doi: 10.1103/PhysRevLett.124.182001.

A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, A. Suresh, and J. Thaler. Scaffolding simulations with deep learning for high-dimensional deconvolution. In *9th International Conference on Learning Representations*, 2021.

M. Backes, A. Butter, M. Dunford, and B. Malaescu. An unfolding method based on conditional invertible neural networks (cINN) using iterative training. *SciPost Physiscs Core*, 7:007, 2024. doi: 10.21468/SciPostPhysCore.7.1.007.

M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone, and U. Köthe. Invertible networks or partons to detector and back again. *SciPost Physics*, 9:074, 2020. doi: 10.21468/SciPostPhys.9.5.074.

P. Chakravarti, M. Kuusela, J. Lei, and L. Wasserman. Model-independent detection of new physics signals using interpretable semi-supervised classifier tests. *The Annals of Applied Statistics*, 17(4):2759–2795, 2023.

K. Cranmer, J. Pavez, and G. Louppe, Approximating likelihood ratios with calibrated discriminative classifiers, 2015, preprint arXiv:1506.02169 [stat.AP].

References II

G. D'Agostini. A multidimensional unfolding method based on Bayes' theorem. *Nuclear Instruments and Methods A*, 362:487–498, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

A. Höcker and V. Kartvelishvili. SVD approach to data unfolding. *Nuclear Instruments and Methods in Physics Research A*, 372:469–481, 1996.

N. Huetsch, J. M. Villadamigo, A. Shmakov, S. Diefenbacher, V. Mikuni, T. Heimel, M. Fenton, K. Greif, B. Nachman, D. Whiteson, A. Butter, and T. Plehn. The landscape of unfolding with machine learning. *SciPost Physics*, 18:070, 2025. doi: 10.21468/SciPostPhys.18.2.070.

A. Kondor. Method of convergent weights – An iterative procedure for solving Fredholm's integral equations of the first kind. *Nuclear Instruments and Methods*, 216:177–181, 1983.

T. Manole, P. Bryant, J. Alison, M. Kuusela, and L. Wasserman. Background modeling for double Higgs boson production: Density ratios and optimal transport. *The Annals of Applied Statistics*, 18(4):2950–2978, 2024.

References III

H. N. Mülthei and B. Schorr. On an iterative method for a class of integral equations of the first kind. *Mathematical Methods in the Applied Sciences*, 9:137–168, 1987a.

H. N. Mülthei and B. Schorr. On an iterative method for the unfolding of spectra. *Nuclear Instruments and Methods in Physics Research A*, 257:371–377, 1987b.

H. N. Mülthei and B. Schorr. On properties of the iterative maximum likelihood reconstruction method. *Mathematical Methods in the Applied Sciences*, 11:331–342, 1989.

H. N. Mülthei. Iterative continuous maximum-likelihood reconstruction method. *Mathematical Methods in the Applied Sciences*, 15:275–286, 1992.

A. Ore and T. Plehn. Unfolding without iterations, adversaries, or surrogates. 2 2026.

S. Schmitt. TUnfold, an algorithm for correcting migration effects in high energy physics. *Journal of Instrumentation*, 7:T10003, 2012.

J. Walchessen, A. Lenzi, and M. Kuusela. Neural likelihood surfaces for spatial processes with computationally intensive or intractable likelihoods. *Spatial Statistics*, 62: 100848, 2024.

Backup

Classical unfolding methods: Discretization

- Problem usually discretized using histograms (splines are also sometimes used)
- Let $\{T_i\}_{i=1}^{p}$ and $\{S_i\}_{i=1}^{n}$ be binnings of the true space T and the smeared space S
- Smeared histogram $\boldsymbol{y} = [y_1, \ldots, y_n]^{\mathrm{T}}$ with mean

$$\boldsymbol{\mu} = \left[\int_{S_1} g(s)\,\mathrm{d}s, \ldots, \int_{S_n} g(s)\,\mathrm{d}s \right]^{\mathrm{T}}$$

- Quantity of interest:

$$\boldsymbol{\lambda} = \left[\int_{T_1} f(t)\,\mathrm{d}t, \ldots, \int_{T_p} f(t)\,\mathrm{d}t \right]^{\mathrm{T}}$$

- The mean histograms are related by $\boldsymbol{\mu} = \boldsymbol{K}\boldsymbol{\lambda}$, where the elements of the *response matrix* $\boldsymbol{K}$ are given by

$$K_{i,j} = \frac{\int_{S_i}\int_{T_j} k(s,t) f(t)\,\mathrm{d}t\,\mathrm{d}s}{\int_{T_j} f(t)\,\mathrm{d}t} = P(\text{smeared event in bin } i \,|\, \text{true event in bin } j)$$

- The discretized statistical model becomes

$$\boldsymbol{y} \sim \mathrm{Poisson}(\boldsymbol{K}\boldsymbol{\lambda})$$

and we wish to make inferences about $\boldsymbol{\lambda}$ under this model

Classical unfolding methods: Regularization

- When the number of true bins p is large, the response matrix $\boldsymbol{K}$ is severely ill-conditioned
- The unfolded histogram $\boldsymbol{\lambda}$ is therefore typically estimated using a *regularized* estimator
  - Main idea: bias $\uparrow$, variance $\downarrow$ $\Rightarrow$ $\mathrm{MSE} \downarrow$
- Two main approaches:
  1. Tikhonov regularization (e.g., SVD by Höcker and Kartvelishvili (1996) and TUnfold by Schmitt (2012)):

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}^p} (\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\lambda})^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\lambda}) + \delta P(\boldsymbol{\lambda})$$

     with

$$P_{\mathrm{SVD}}(\boldsymbol{\lambda}) = \left\| \boldsymbol{L} \begin{bmatrix} \lambda_1/\lambda_1^{\mathrm{MC}} \\ \lambda_2/\lambda_2^{\mathrm{MC}} \\ \vdots \\ \lambda_p/\lambda_p^{\mathrm{MC}} \end{bmatrix} \right\|^2 \quad \text{or} \quad P_{\mathrm{TUnfold}}(\boldsymbol{\lambda}) = \|\boldsymbol{L}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{\mathrm{MC}})\|^2,$$

     where $\boldsymbol{L}$ is usually the discretized second derivative (other choices also possible)
  2. Expectation-maximization iteration with early stopping (D'Agostini, 1995):

$$\lambda_j^{(t+1)} = \frac{\lambda_j^{(t)}}{\sum_{i=1}^n K_{i,j}} \sum_{i=1}^n \frac{K_{i,j} y_i}{\sum_{k=1}^p K_{i,k} \lambda_k^{(t)}}, \text{ with } \boldsymbol{\lambda}^{(0)} = \boldsymbol{\lambda}^{\mathrm{MC}}$$

- These methods typically regularize by creating a bias toward a MC ansatz $\boldsymbol{\lambda}^{\mathrm{MC}}$

Aside: The EM algorithm

Assume that we are interested in finding the MLE of parameter $\boldsymbol{\theta}$ in the model $p(\boldsymbol{y}|\boldsymbol{\theta})$

Assume that the model is such that maximizing $\ell(\boldsymbol{\theta}; \boldsymbol{y}) = \log p(\boldsymbol{y}|\boldsymbol{\theta})$ is difficult but there exists a random variable $\boldsymbol{x}$, with $\boldsymbol{y} = g(\boldsymbol{x})$ for some many-to-one mapping g, such that $\ell(\boldsymbol{\theta}; \boldsymbol{x}) = \log p(\boldsymbol{x}|\boldsymbol{\theta})$ is easier to handle

Here, $\boldsymbol{x}$ and $\boldsymbol{y}$ are called the *complete* and *incomplete* data, respectively

- For example, we could have $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$ for some latent variable $\boldsymbol{z}$

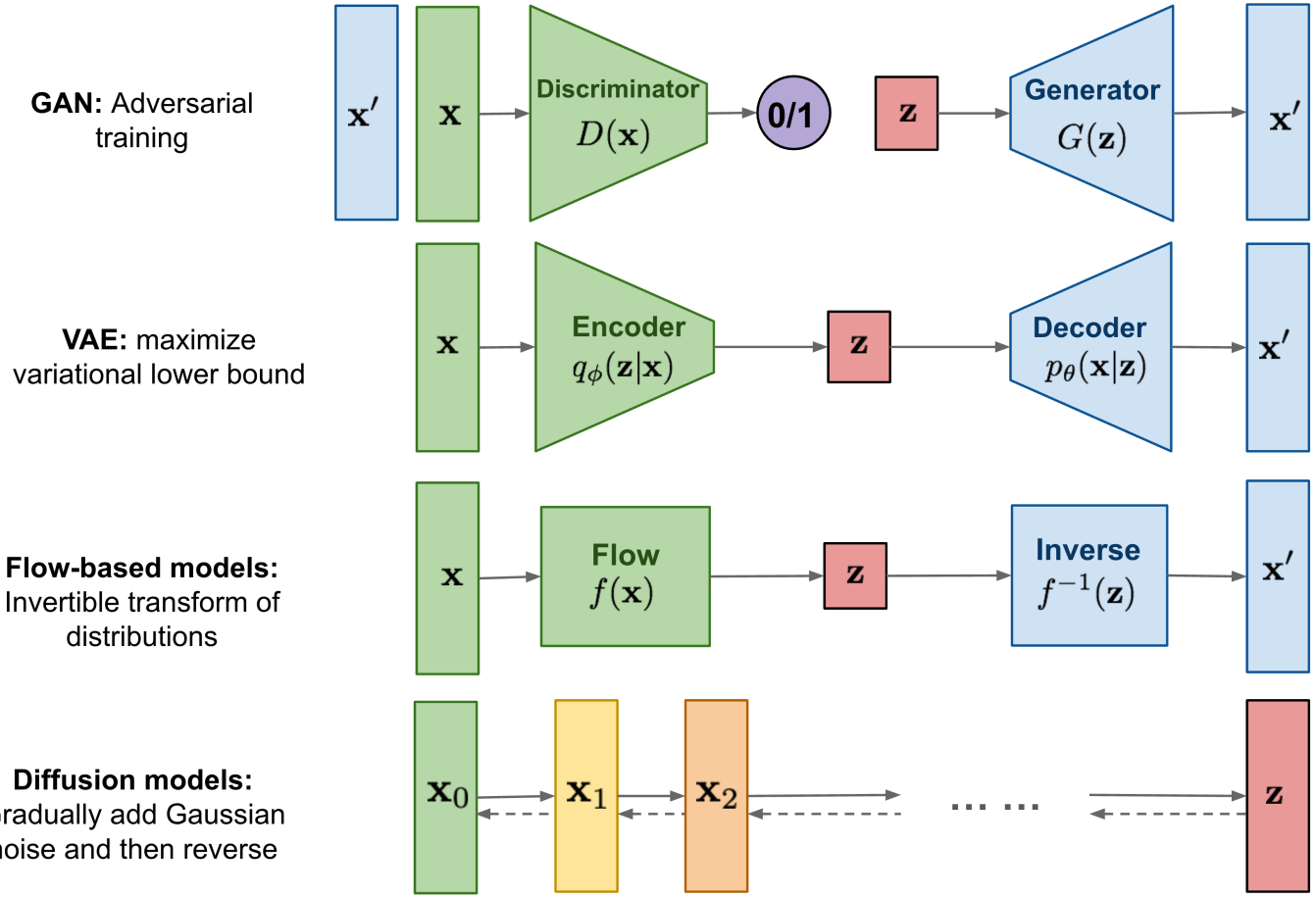
Then the following expectation–maximization (EM) algorithm can be used to iteratively maximize $\ell(\boldsymbol{\theta}; \boldsymbol{y})$:

- **E-step:** Compute $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) = \mathbb{E}(\log p(\boldsymbol{x}|\boldsymbol{\theta})|\boldsymbol{y}, \boldsymbol{\theta}^{(k)})$
- **M-step:** Set $\boldsymbol{\theta}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$

Theorem (Dempster et al. (1977)): $\ell(\boldsymbol{\theta}^{(k+1)}; \boldsymbol{y}) \geq \ell(\boldsymbol{\theta}^{(k)}; \boldsymbol{y})$ for all $k = 0, 1, 2, \ldots$

Aside: Generative models

The basic idea in most ML generative models is to train a neural model to map from a latent variable $\boldsymbol{z}$ generated from some tractable distribution (usually $N(\mathbf{0}, \boldsymbol{I})$) to the data distribution



(Source: `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`)

Generative unfolding

Let's consider again the unbinned EM update:

$$f^{(k+1)}(t) = \int \frac{p(s|t)f^{(k)}(t)}{\int p(s|t')f^{(k)}(t')dt'} g(s)ds$$

The key idea in generative unfolding (e.g., Bellagente et al., 2020; Backes et al., 2024) is to use generative ML to sample from $f^{(k+1)}(t)$

By Bayes' rule[1],

$$p(t|s, f^{(k)}) = \frac{p(s|t)f^{(k)}(t)}{\int p(s|t')f^{(k)}(t')dt'},$$

which enables us to write the EM update as

$$f^{(k+1)}(t) = \int p(t|s, f^{(k)})g(s)ds$$

So, to sample from $f^{(k+1)}(t)$, it suffices to sample $Y_i \sim g(s)$ followed by $X_i^{(k)} \sim p(t|s = Y_i, f^{(k)})$

[1] Note that the use of Bayes' rule doesn't make this procedure Bayesian because X and Y are random variables here.

Generative unfolding

The smeared observations already give us the sample $Y_i \sim g(s)$

The sample $X_i^{(k)} \sim p(t|s = Y_i, f^{(k)})$ is obtained using a conditional generative ML model trained using the MC sample $\{X_i', Y_i'\}_{i=1}^n$ reweighted according to $f^{(k)}(t)$

Once the sample $\{X_i^{(k)}\}_{i=1}^n$ has been obtained, one can train a classifier to separate $\{X_i^{(k)}\}_{i=1}^n$ vs. $\{X_i'\}_{i=1}^n$ to obtain a reweighting of the MC sample

This reweighted sample is then used to retrain the generative model for the next iteration

After K iterations, the output of the procedure can be taken to be either the sample $\{X_i^{(K)}\}_{i=1}^n$ or the corresponding reweighting of the MC sample

Tikhonov regularization, $P(\boldsymbol{\lambda}) = \|\boldsymbol{\lambda}\|^2$, varying δ