MultiFold:
A user's perspective

Youqi Song (youqi.song@yale.edu)

2024 RHIC/AGS annual users' meeting, BNL

ML&AI workshop, 6/11/2024

Supported in part by U.S. DEPARTMENT OF ENERGY Office of Science

Outline

- What is MultiFold?
- What are some applications of MultiFold?
- How does MultiFold work?

Outline

- What is MultiFold? → the bare minimum to get started
- What are some applications of MultiFold? → proof that the algorithm works
- How does MultiFold work? → peeking into the black box

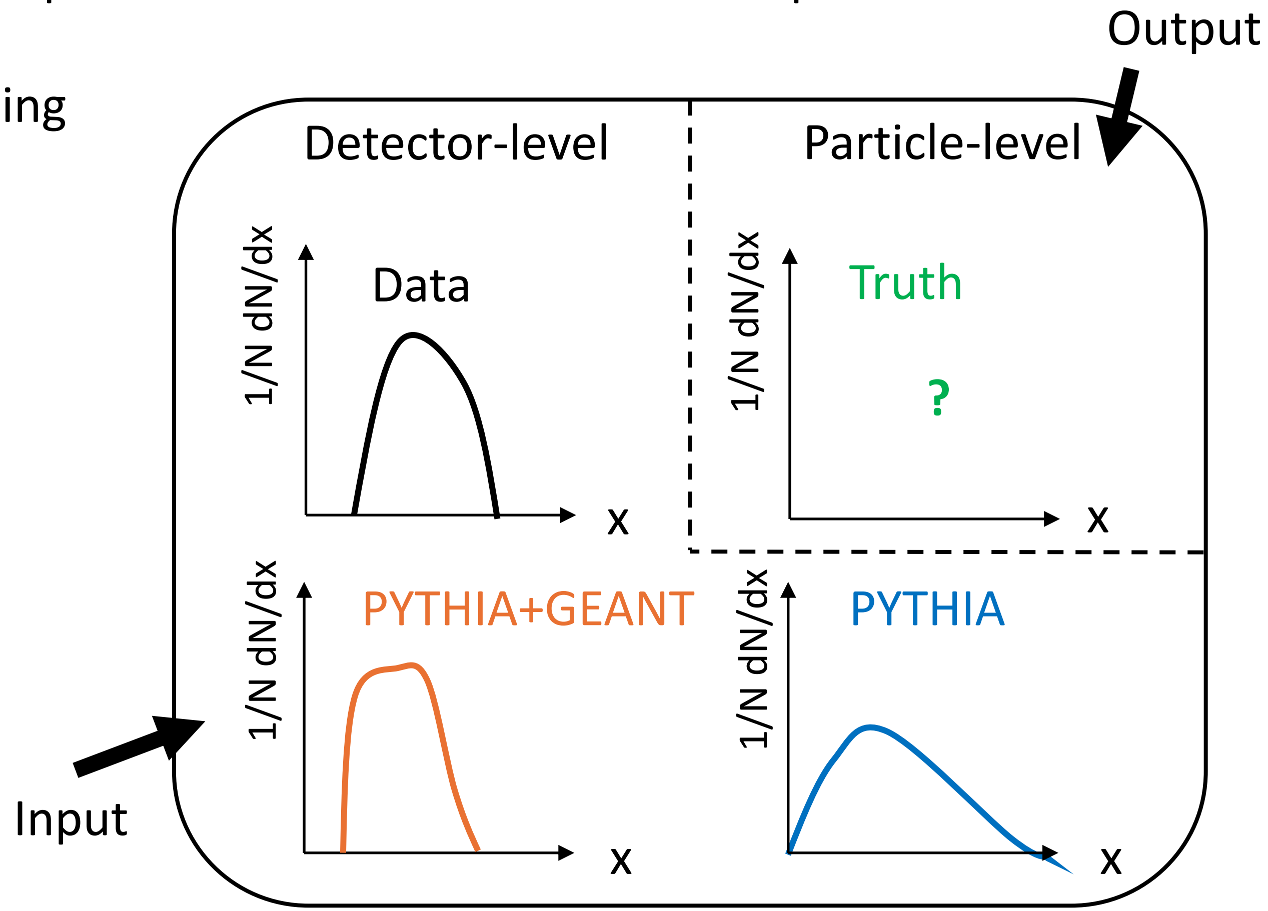
Unfolding

- **Corrects for detector effects**, due to inefficiency, finite resolution, …
  → Allows for result comparison with theories and other experiments

Unfolding

- **Corrects for detector effects**, due to inefficiency, finite resolution, …
  → Allows for result comparison with theories and other experiments

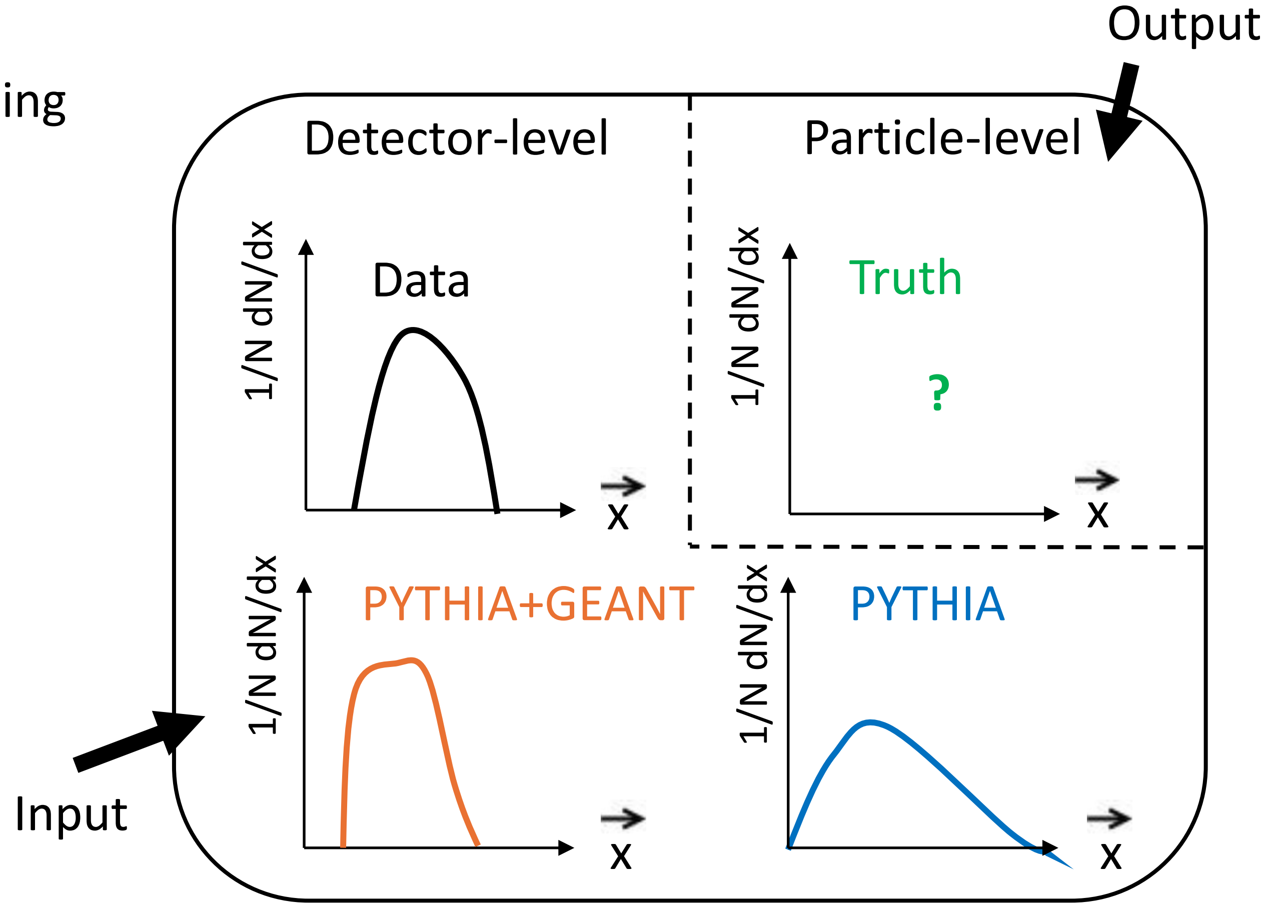
- **Ingredients** for unfolding



Unfolding

- **Corrects for detector effects**, due to inefficiency, finite resolution, …
  → Allows for result comparison with theories and other experiments

- **Ingredients** for unfolding

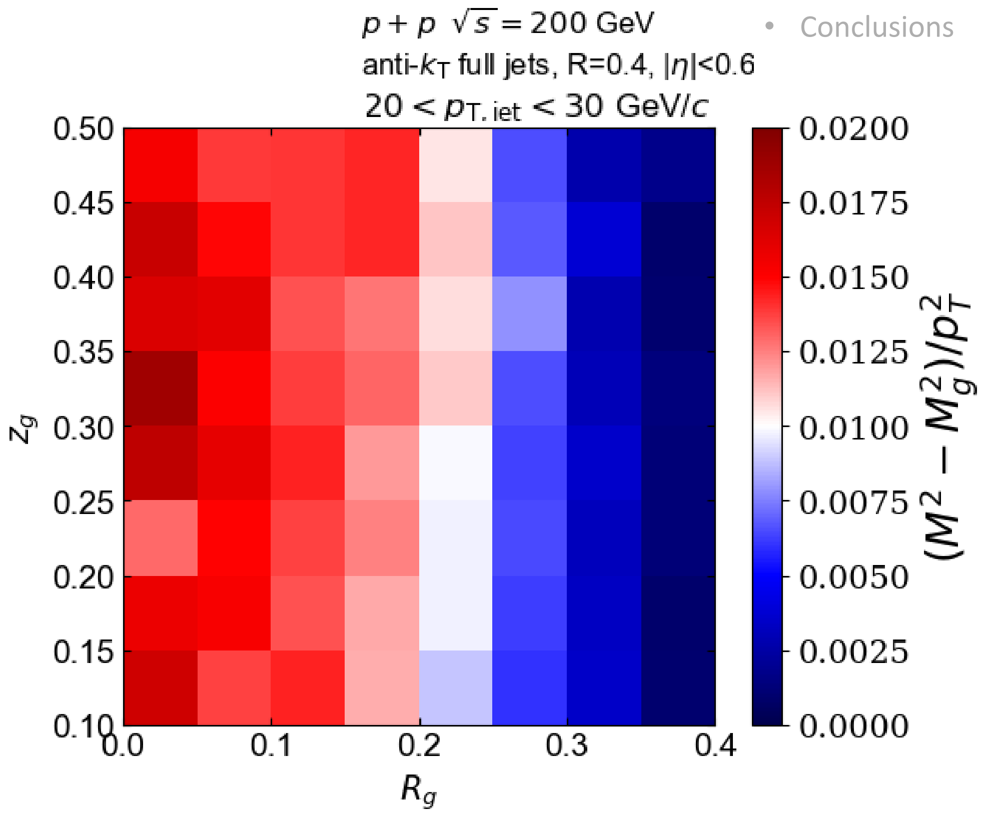


Unfolding

- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - Machine learning driven
    - Unbinned
    - **Simultaneously unfolds many observables**
      → Correlation information is retained!

Unfolding

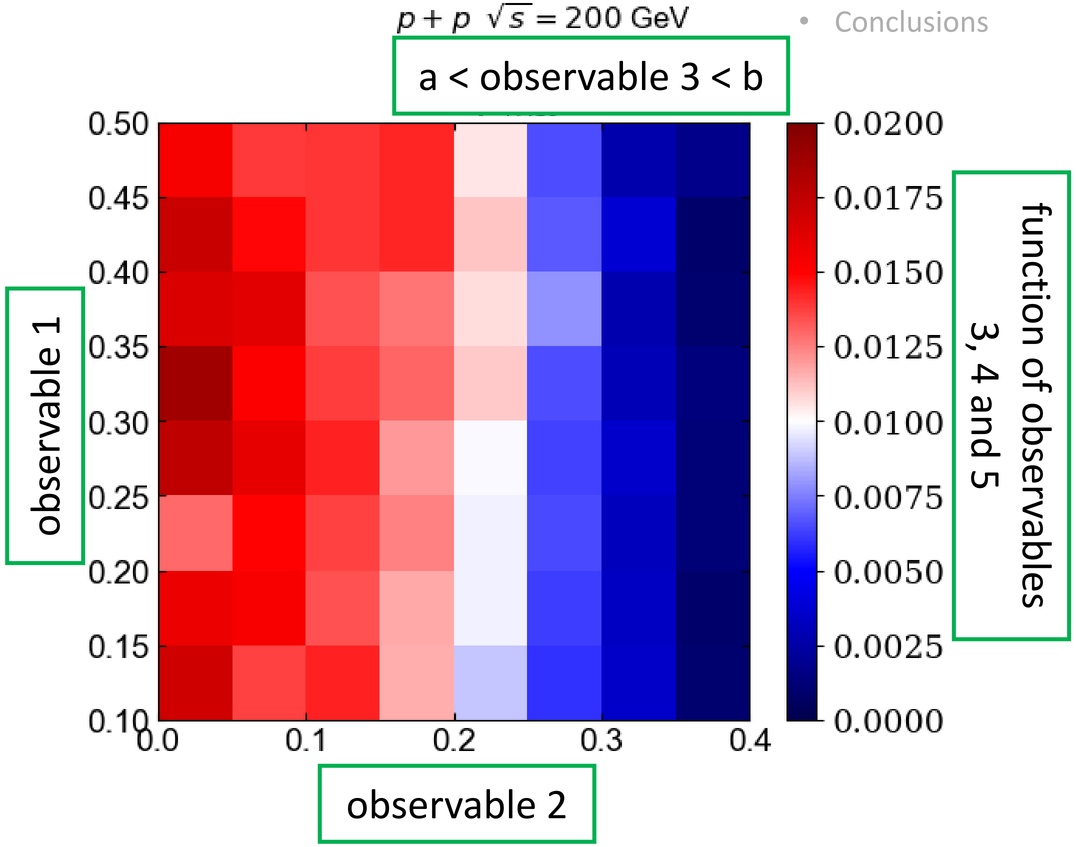
- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - Machine learning driven
    - Unbinned
    - **Simultaneously unfolds many observables**
      → Correlation information is retained!



Unfolding

- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - Machine learning driven
    - Unbinned
    - **Simultaneously unfolds many observables**

      → Correlation information is retained!

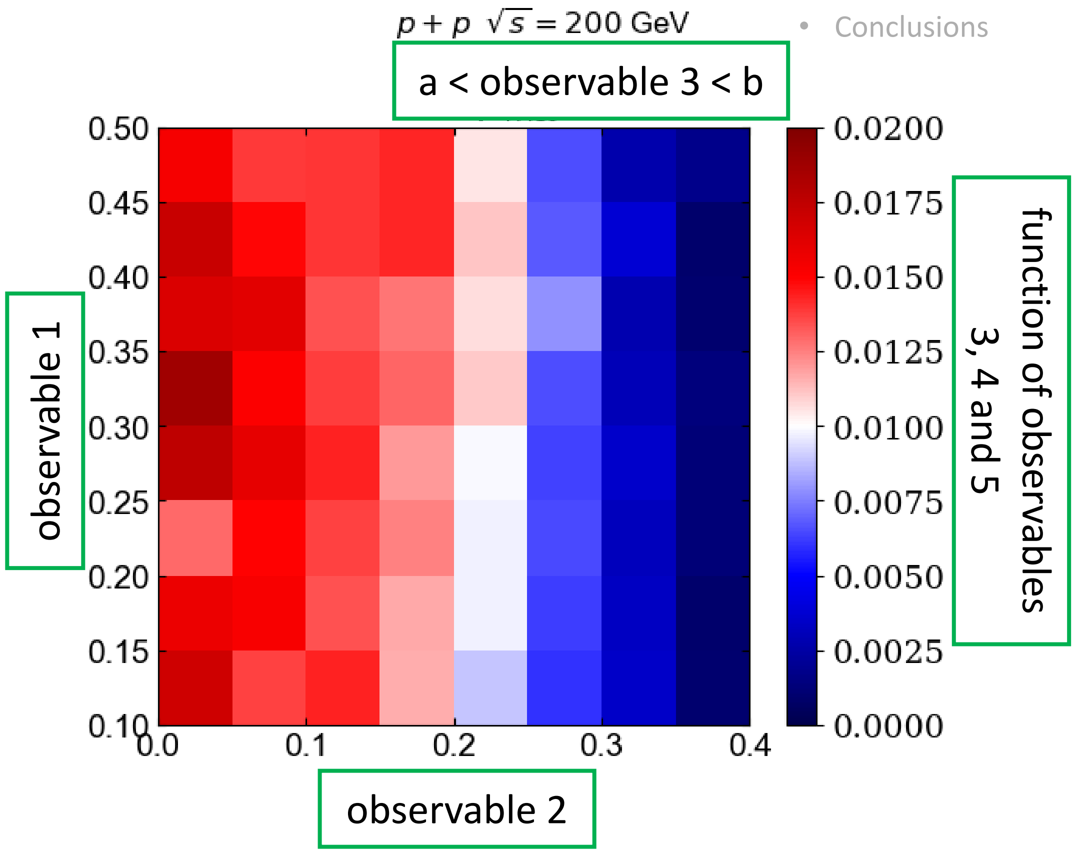


Unfolding

- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - Machine learning driven
    - Unbinned
    - **Simultaneously unfolds many observables**
      → Correlation information is retained!

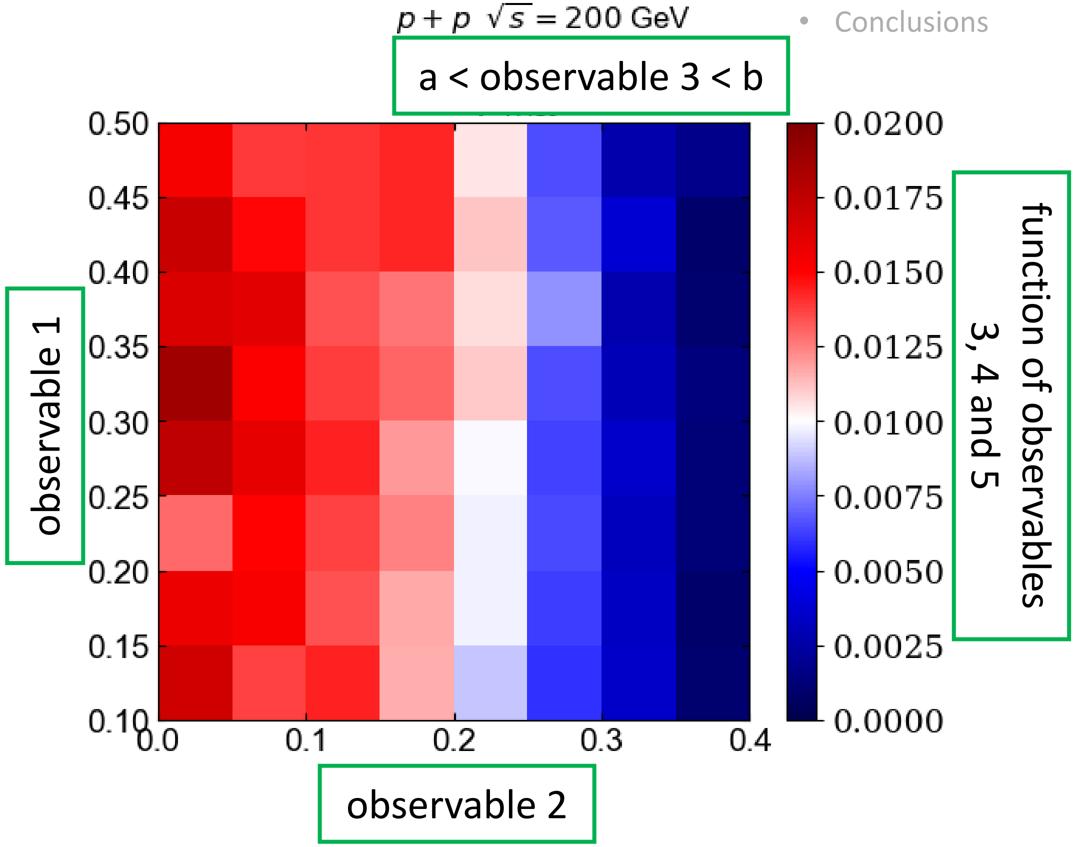
- Variations of the MultiFold/OmniFold algorithm:

Variation	Input
UniFold	One event observable
MultiFold	**Many event observables**
OmniFold	Full phase space of the event



Unfolding

- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - Machine learning driven
    - Unbinned
    - **Simultaneously unfolds many observables**
      → Correlation information is retained!



- Variations of the MultiFold/OmniFold algorithm:

Variation	Input
UniFold	One event observable
MultiFold	**Many event observables**
OmniFold	Full phase space of the event

To get started

- git clone git@github.com:ericmetodiev/OmniFold.git
  - More updated repo at https://github.com/hep-lbdl/OmniFold
- Run OmniFold Demo.ipynb
- Replace example files with your own trees

We also have to specify `itnum`: how many iterations of the unfolding procedure we want to do.

Customize: Change `itnum` to your desired number of unfolding iterations.

In [5]:
```
# how many iterations of the unfolding process
itnum = 3
```

There are three flavors of OmniFold. In order of increasing sophistication, they are:

- **UniFold**: Represent the jet as a single observable.
- **MultiFold**: Represent the jet as multiple observables.
- **OmniFold**: Represent the jet as a set of particles.

By default, we will implement MultiFold and represent the jet using six jet substructure observbles:

- `'Mass'`, Jet Mass m: the invaraiant mass of the jet four-vector
- `'Mult'`, Constituent Multiplicity M: the number of particles in the jet
- `'Width'`, Jet Width w: a measure of how broad the jet is
- `'Tau21'`, N-subjettiness Ratio τ_{21}: a measure of how two-pronged the jet is
- `'zg'`, Groomed Momentum Fraction z_g: the energy-sharing of the prongs after grooming
- `'SDMass'`, Groomed Jet Mass m_{SD}: the invariant mass of the jet four-vector after grooming

Customize: Change which observables are used in MultiFold. UniFold corresponds to using a single observable.

In [6]:
```
obs_multifold = ['Mass', 'Mult', 'Width', 'Tau21', 'zg', 'SDMass']
```

The observables are already computed in the samples. We will read them in as an observable dictionary `obs` and also specify histogram style information.

Customize: Add entries to `obs` to define your own observables to be used in MultiFold or to see the unfolding performance on them.

In [7]:
```
# a dictionary to hold information about the observables
obs = {}
```

Snippet of Python notebook from https://github.com/ericmetodiev/OmniFold/blob/master/OmniFold%20Demo.ipynb

Applications

- MultiFold has been applied to several measurements…

in e+p collisions in HERA

Measurement of Lepton-Jet Correlation in Deep-Inelastic Scattering with the H1 Detector Using Machine Learning for Unfolding
H1 Collaboration • V. Andreev (LPI, Moscow (main)) et al. (Aug 27, 2021)
Published in: *Phys.Rev.Lett.* 128 (2022) 13, 132002 • e-Print: 2108.12376 [hep-ex]

8 observables unfolded

Unbinned deep learning jet substructure measurement in high Q2ep collisions at HERA
H1 Collaboration • V. Andreev (Lebedev Inst.) et al. (Mar 23, 2023)
Published in: *Phys.Lett.B* 844 (2023) 138101 • e-Print: 2303.13620 [hep-ex]

6 observables unfolded

in p+p and heavy-ion collisions at RHIC

Measurement of CollinearDrop jet mass and its correlation with SoftDrop groomed jet substructure observables in $\sqrt{s} = 200$ GeV pp collisions by STAR
STAR Collaboration • Youqi Song for the collaboration. (Jul 15, 2023)
e-Print: 2307.07718 [nucl-ex]

6 observables unfolded

Generalized angularities measurements from STAR at $\sqrt{s_{\mathrm{NN}}} = 200$ GeV
STAR Collaboration • Tanmay Pani (Rutgers U., Piscataway) for the collaboration. (Mar 20, 2024)
Contribution to: Quark Matter 2023 • e-Print: 2403.13921 [nucl-ex]

7 observables unfolded

in p+p collisions at LHC

Multidifferential study of identified charged hadron distributions in Z-tagged jets in proton-proton collisions at $\sqrt{s}$ =13 TeV
LHCb Collaboration • Roel Aaij (Nikhef, Amsterdam) et al. (Aug 24, 2022)
Published in: *Phys.Rev.D* 108 (2023) L031103 • e-Print: 2208.11691 [hep-ex]

4 observables unfolded

A simultaneous unbinned differential cross section measurement of twenty-four Z+jets kinematic observables with the ATLAS detector
ATLAS Collaboration • Georges Aad (Marseille, CPPM) et al. (May 30, 2024)
e-Print: 2405.20041 [hep-ex]

24 observables unfolded

Measurement of event shapes in minimum bias events from pp collisions at 13 TeV
The CMS Collaboration

8 observables unfolded

Applications

- MultiFold has been applied to several measurements…

in e+p collisions in HERA

Measurement of Lepton-Jet Correlation in Deep-Inelastic Scattering with the H1 Detector Using Machine Learning for Unfolding
H1 Collaboration • V. Andreev (LPI, Moscow (main)) et al. (Aug 27, 2021)
Published in: *Phys.Rev.Lett.* 128 (2022) 13, 132002 • e-Print: 2108.12376 [hep-ex]

8 observables unfolded

Unbinned deep learning jet substructure measurement in high Q2ep collisions at HERA
H1 Collaboration • V. Andreev (Lebedev Inst.) et al. (Mar 23, 2023)
Published in: *Phys.Lett.B* 844 (2023) 138101 • e-Print: 2303.13620 [hep-ex]

6 observables unfolded

in p+p and heavy-ion collisions at RHIC

Measurement of CollinearDrop jet mass and its correlation with SoftDrop groomed jet substructure observables in $\sqrt{s} = 200$ GeV pp collisions by STAR
STAR Collaboration • Youqi Song for the collaboration. (Jul 15, 2023)
e-Print: 2307.07718 [nucl-ex]

6 observables unfolded

Generalized angularities measurements from STAR at $\sqrt{s_{\mathrm{NN}}} = 200$ GeV
STAR Collaboration • Tanmay Pani (Rutgers U., Piscataway) for the collaboration. (Mar 20, 2024)
Contribution to: Quark Matter 2023 • e-Print: 2403.13921 [nucl-ex]

7 observables unfolded

in p+p collisions at LHC

Multidifferential study of identified charged hadron distributions in Z-tagged jets in proton-proton collisions at $\sqrt{s} = 13$ TeV
LHCb Collaboration • Roel Aaij (Nikhef, Amsterdam) et al. (Aug 24, 2022)
Published in: *Phys.Rev.D* 108 (2023) L031103 • e-Print: 2208.11691 [hep-ex]

4 observables unfolded

A simultaneous unbinned differential cross section measurement of twenty-four Z+jets kinematic observables with the ATLAS detector
ATLAS Collaboration • Georges Aad (Marseille, CPPM) et al. (May 30, 2024)
e-Print: 2405.20041 [hep-ex]

24 observables unfolded

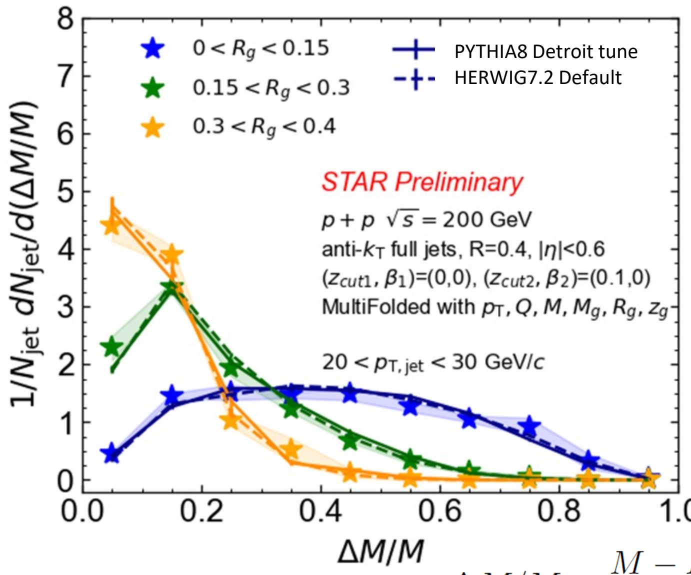
Measurement of event shapes in minimum bias events from pp collisions at 13 TeV
The CMS Collaboration

8 observables unfolded

Applications

- Probing the correlation between perturbative and nonperturbative components within jets at STAR

arxiv: 2307.07718



$$\Delta M/M = \frac{M - M_{\mathrm{g}}}{M}$$

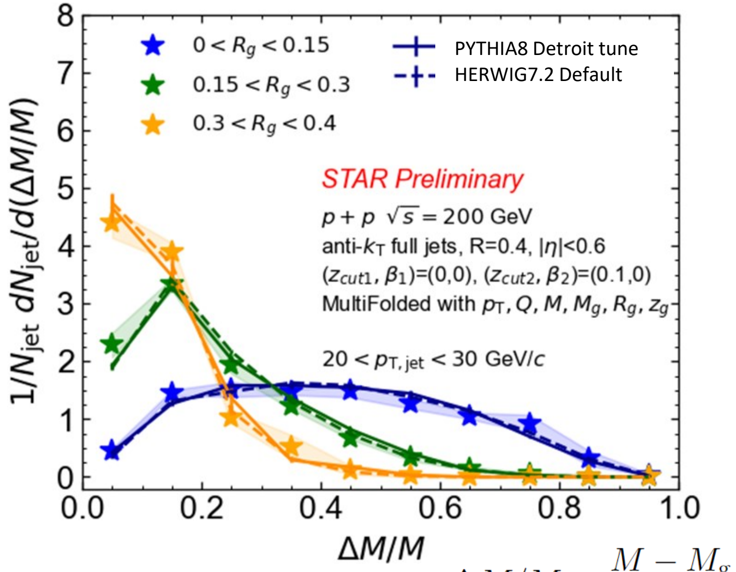
- Simultaneously correct for:
  - p_{T}: transverse momentum
  - $Q^{\kappa} = \frac{1}{(p_{\mathrm{Tjet}})^{\kappa}} \sum_{i\in\mathrm{jet}} q_i \cdot (p_{\mathrm{T}i})^{\kappa}$
  - $M = |\Sigma_{i\in\mathrm{jet}}\, p_i| = \sqrt{E^2 - |\vec{p}|^2}$
  - R_{g}: groomed jet radius
  - M_{g}: groomed jet mass
  - z_{g}: shared momentum fraction

$$z_{\mathrm{g}} = \frac{\min(p_{\mathrm{T},1}, p_{\mathrm{T},2})}{p_{\mathrm{T},1} + p_{\mathrm{T},2}} > z_{\mathrm{cut}}(R_{\mathrm{g}}/R_{\mathrm{jet}})^{\beta}$$

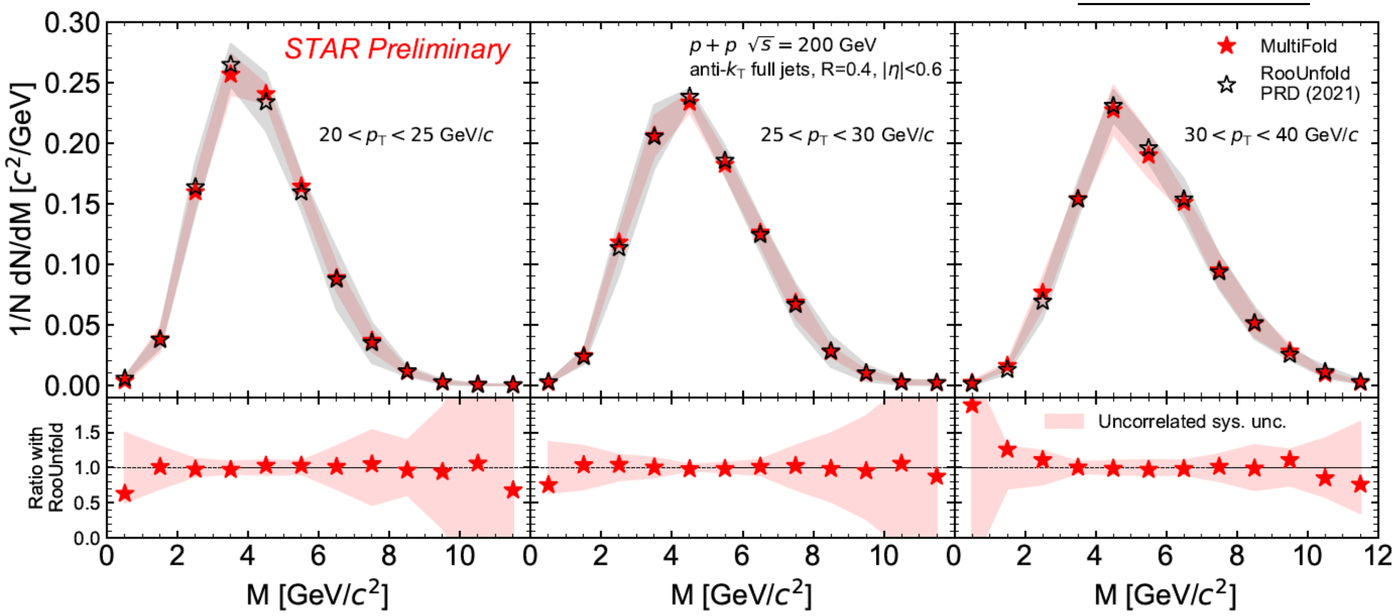
Applications

- Probing the correlation between perturbative and nonperturbative components within jets at STAR

arxiv: 2307.07718



$$\Delta M/M = \frac{M - M_g}{M}$$

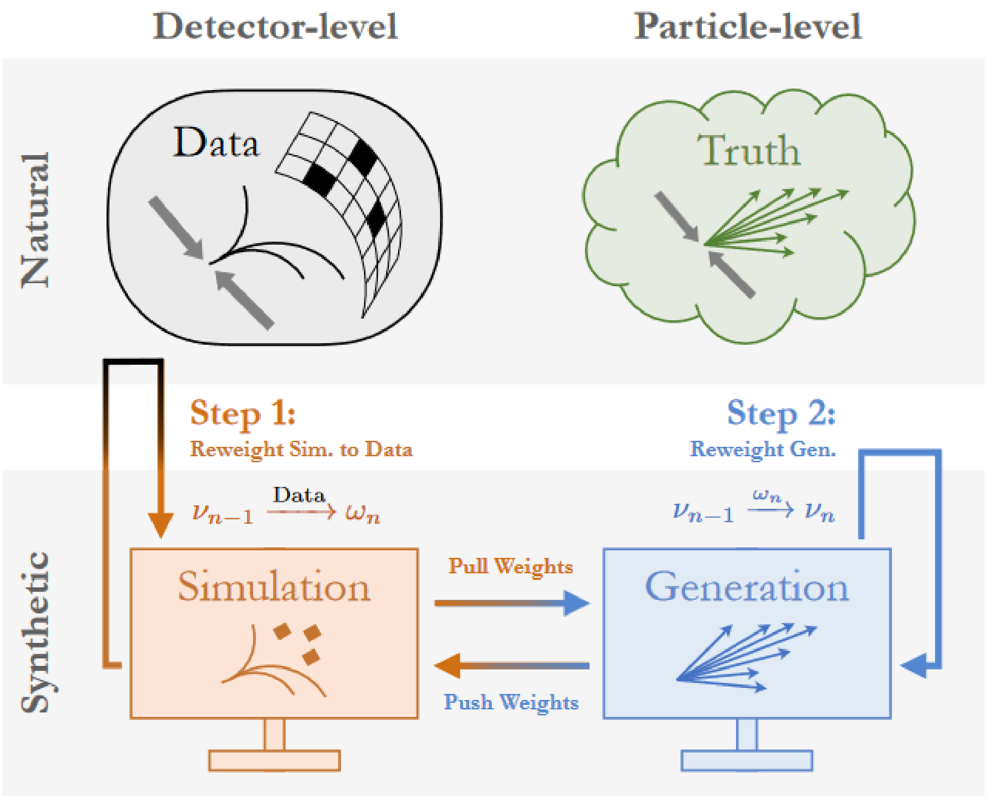


- Simultaneously correct for:
  - p_T: transverse momentum
  - $Q^\kappa = \frac{1}{(p_{\mathrm{Tjet}})^\kappa} \sum_{i \in \mathrm{jet}} q_i \cdot (p_{\mathrm{T}i})^\kappa$
  - $M = |\Sigma_{i \in \mathrm{jet}}\, p_i| = \sqrt{E^2 - |\vec{p}|^2}$
- R_g: groomed jet radius
- M_g: groomed jet mass
- z_g: shared momentum fraction

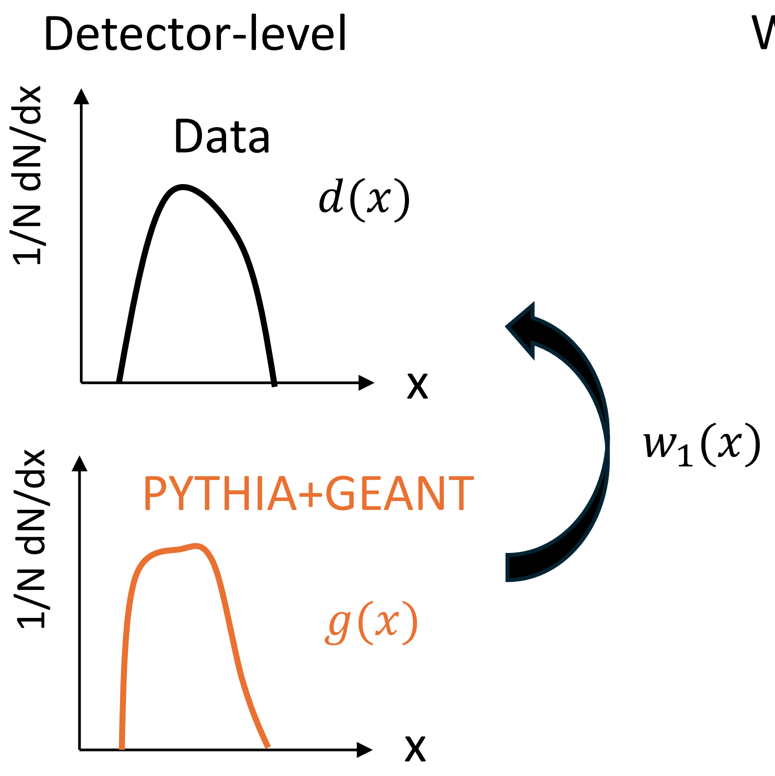
$$z_g = \frac{\min(p_{T,1}, p_{T,2})}{p_{T,1} + p_{T,2}} > z_{\mathrm{cut}} (R_g / R_{\mathrm{jet}})^\beta$$

Good agreement between MultiFold and IBU verified

How does MultiFold work?



Iterative reweighting: Step 1, iteration 1

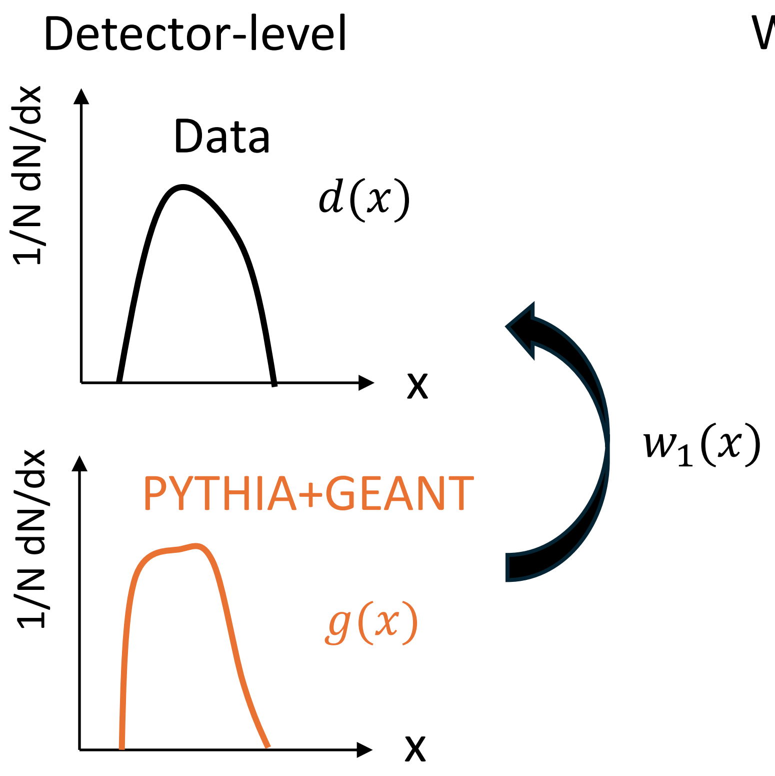


Weights: $w_1(x) = d(x)/g(x)$

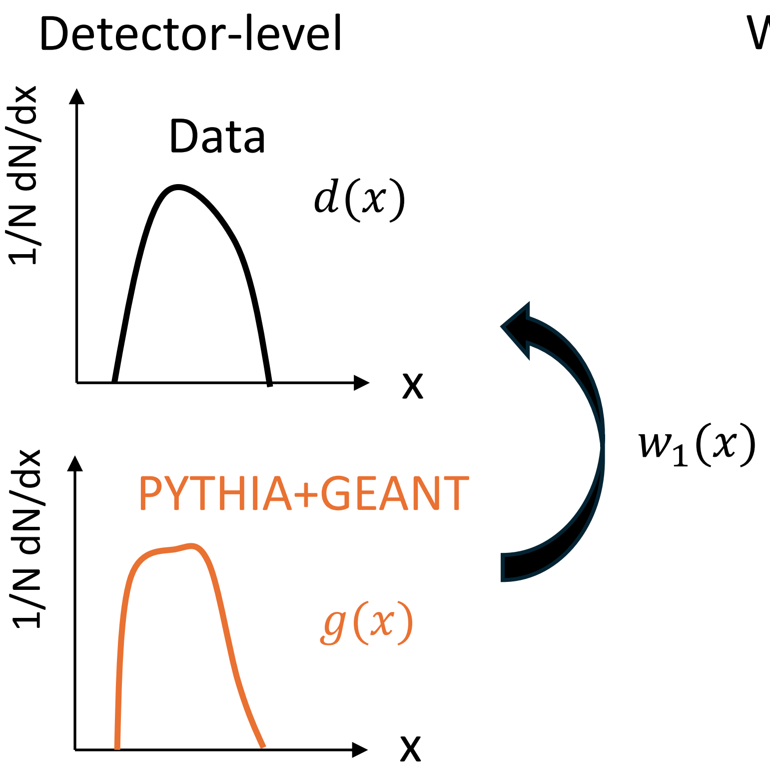
Iterative reweighting: Step 1, iteration 1

Weights: $w_1(x) = d(x)/g(x)$

Ok for the binned case



Iterative reweighting: Step 1, iteration 1

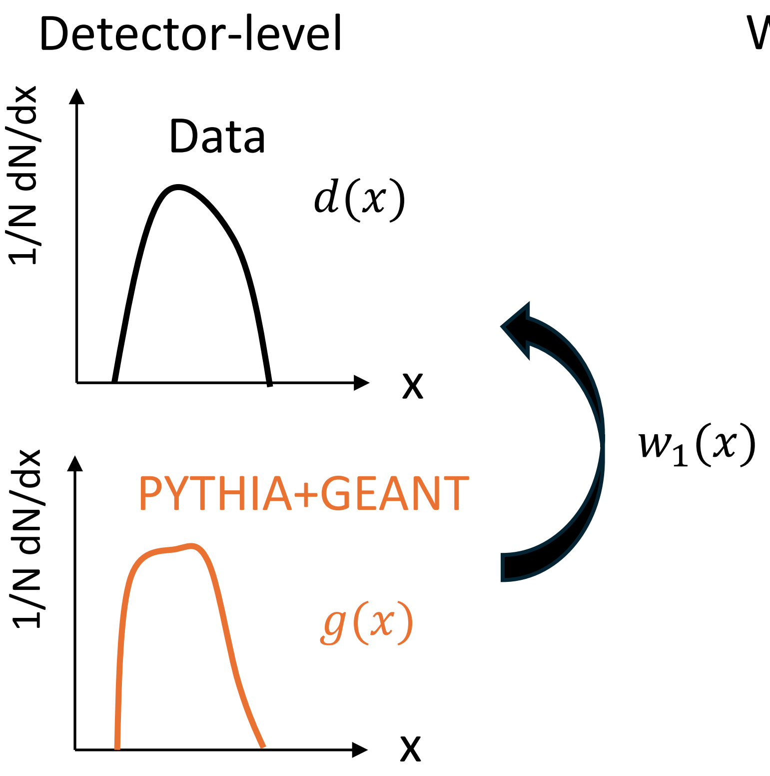


Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

Iterative reweighting: Step 1, iteration 1



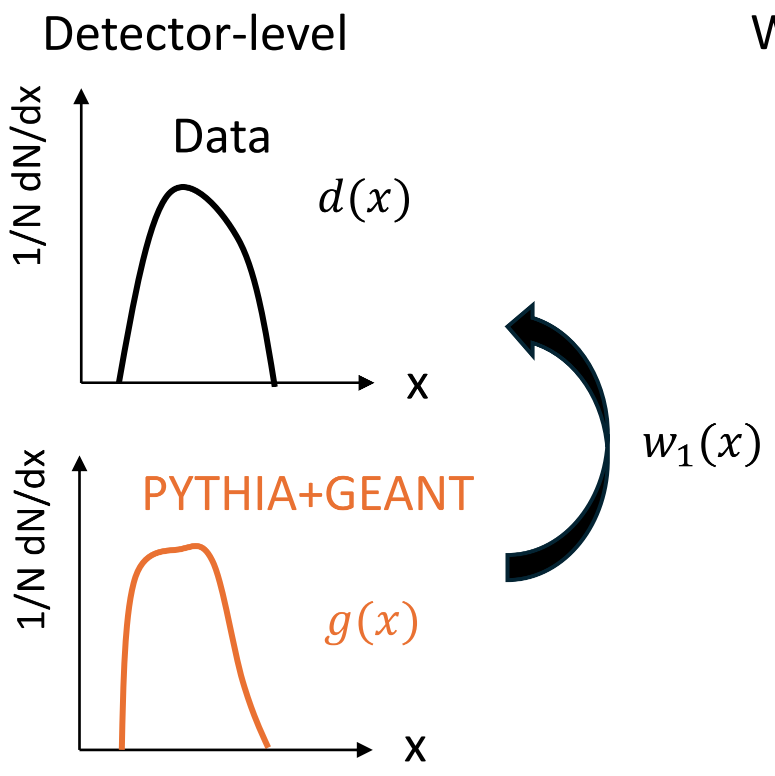
Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

Iterative reweighting: Step 1, iteration 1



Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

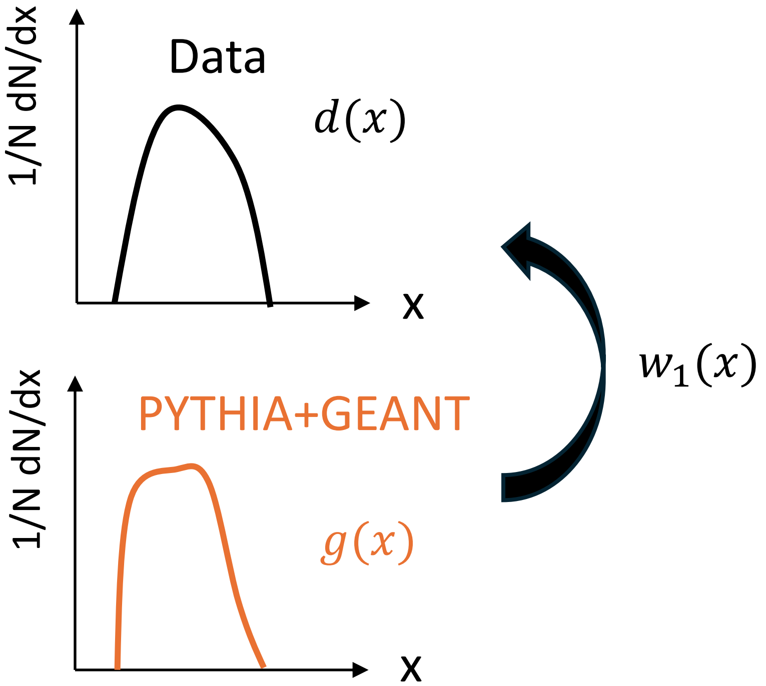
where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

Density estimation → Classification!

Iterative reweighting: Step 1, iteration 1

Detector-level

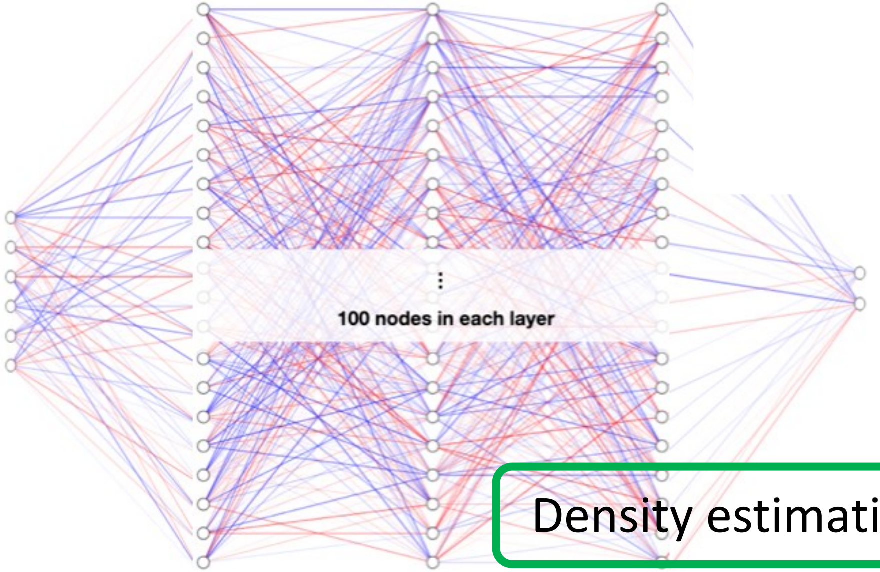


Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

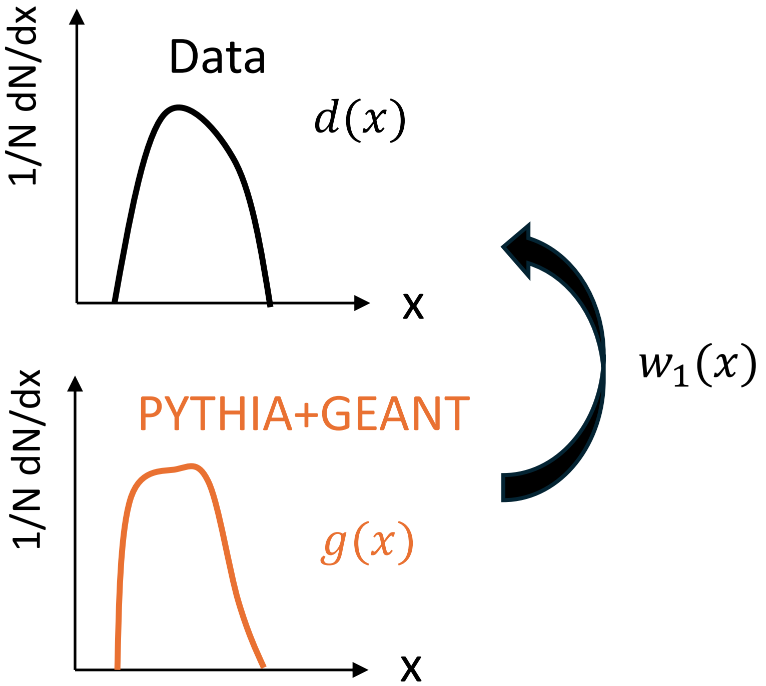
to distinguish events coming from data vs from PYTHIA+GEANT



Density estimation → Classification!

Iterative reweighting: Step 1, iteration 1

Detector-level



Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

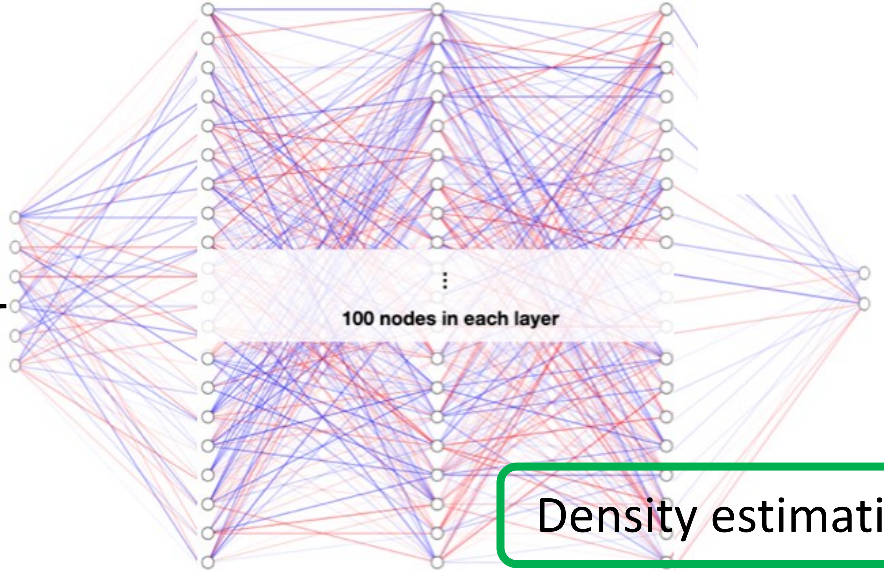
$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

NN input

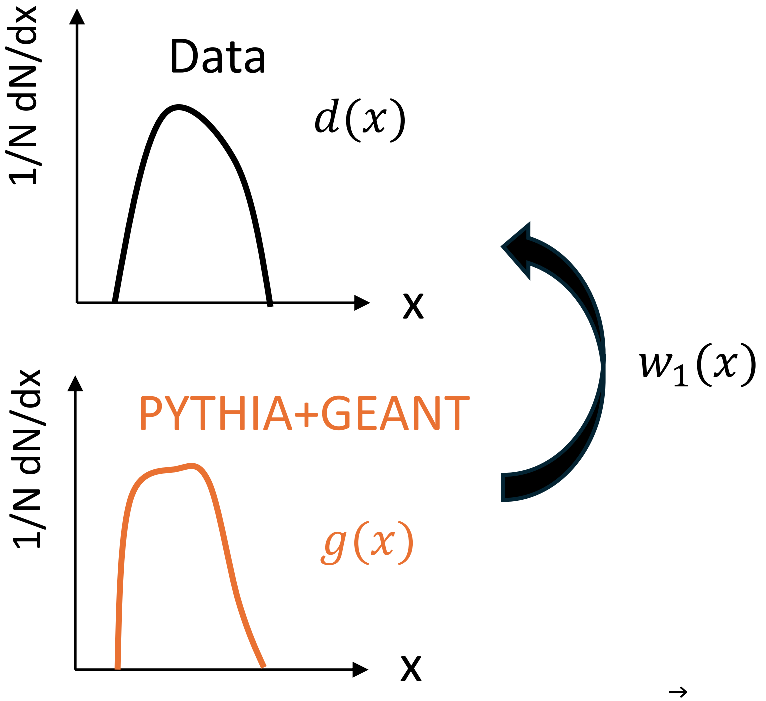
$$\vec{x} = \begin{cases} \text{event observable 1} \\ \dots \\ \text{event observable 6} \end{cases}$$



Density estimation → Classification!

Iterative reweighting: Step 1, iteration 1

Detector-level



Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

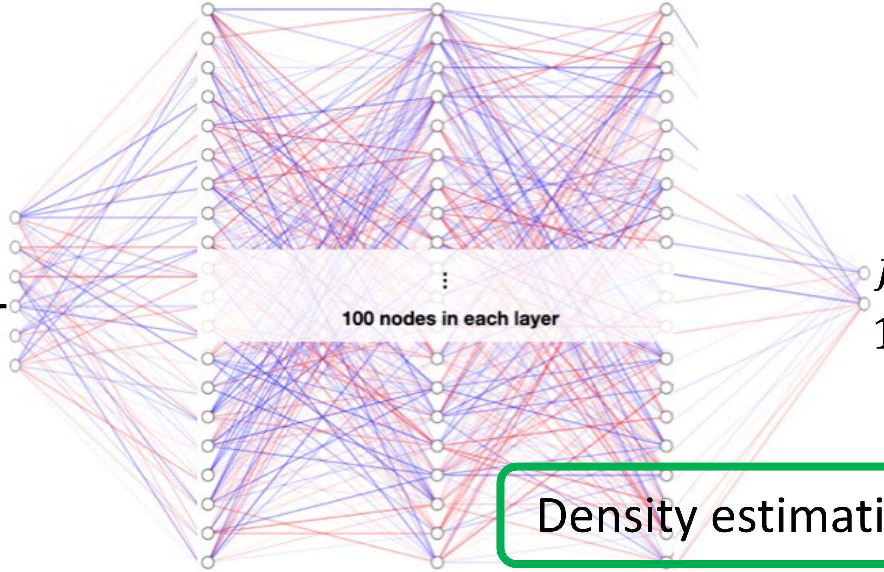
$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

NN input

$$\vec{x} = \begin{cases} \text{event observable 1} \\ \dots \\ \text{event observable 6} \end{cases}$$



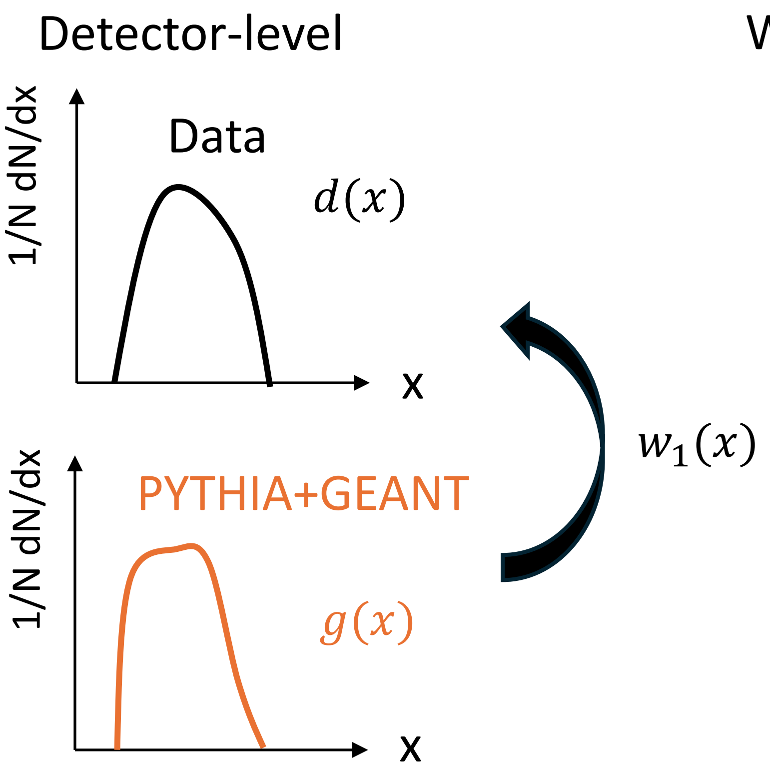
NN output

$f(\vec{x}) =$ probability that event is from data

$1 - f(\vec{x}) =$ probability event is from GEANT

Density estimation → Classification!

Iterative reweighting: Step 1, iteration 1



Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1-f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

NN output

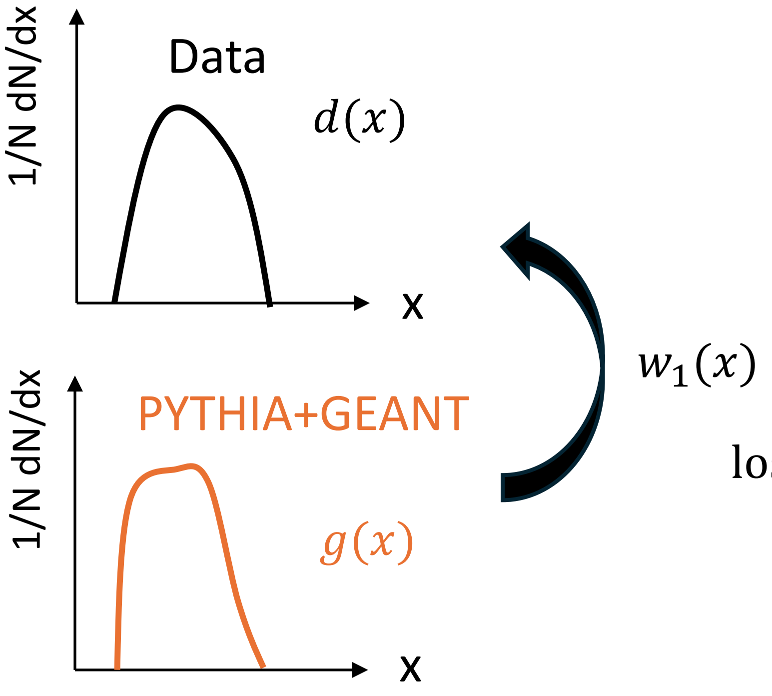
$f(\vec{x}) =$ probability that event is from data

$1 - f(\vec{x}) =$ probability event is from GEANT

Density estimation → Classification!

Iterative reweighting: Step 1, iteration 1

Detector-level



Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1-f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

$$\text{loss}(f(x)) = -\sum_{x \in data} \log(f(x)) - \sum_{x \in geant} \log(1-f(x))$$

NN output

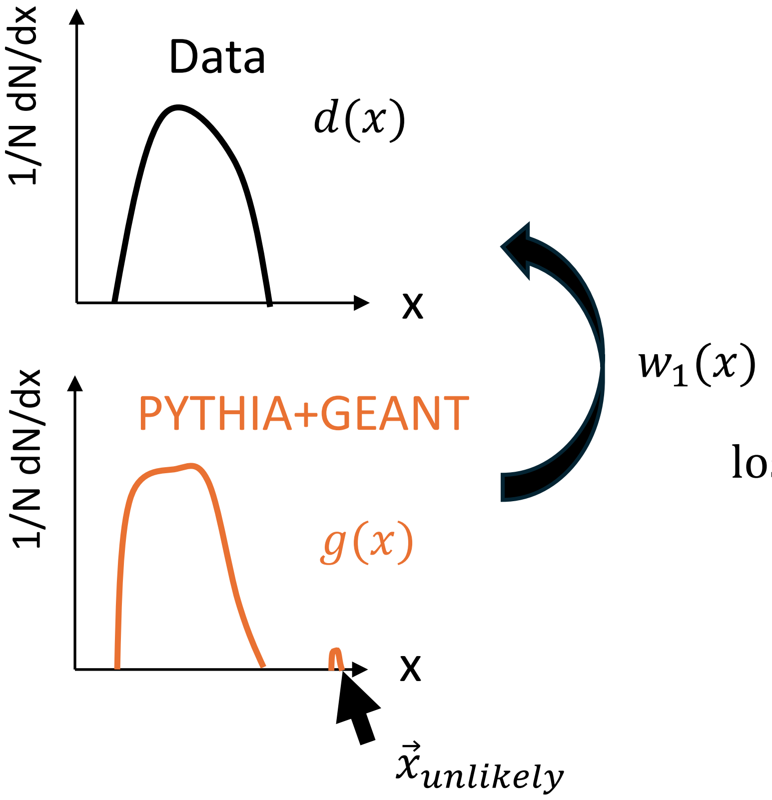
$f(\vec{x}) =$ probability that event is from data

$1 - f(\vec{x}) =$ probability event is from GEANT

Density estimation → Classification!

Iterative reweighting: Step 1, iteration 1

Detector-level



$$f(\vec{x}_{unlikely}) = 0$$

$$\text{loss}\left(f(\vec{x}_{unlikely})\right) = 0$$

Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

$$\text{loss}(f(x)) = -\sum_{x \in data} \log(f(x)) - \sum_{x \in geant} \log(1 - f(x))$$

NN output

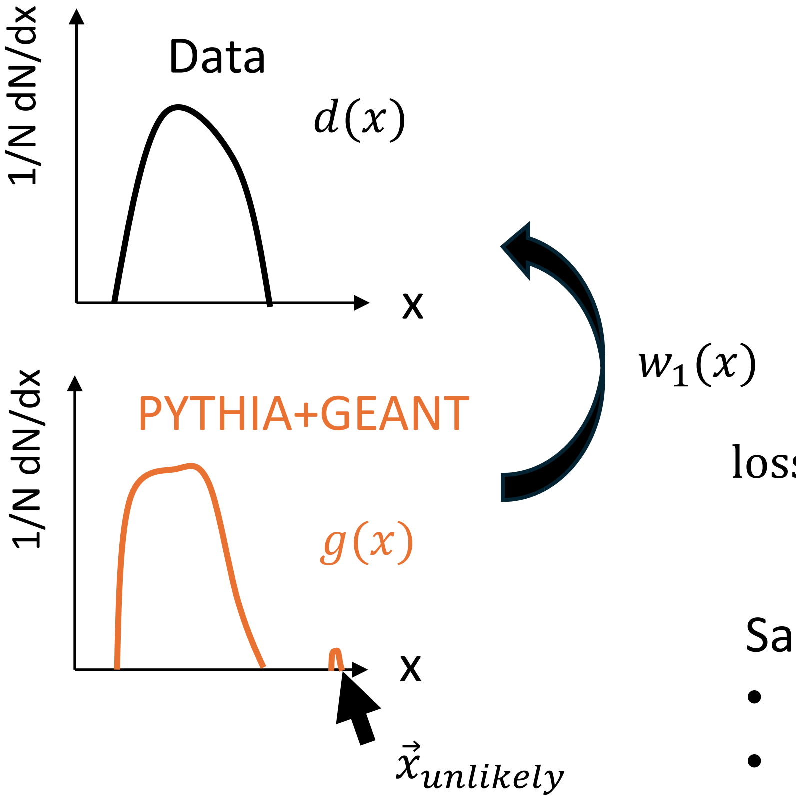
$f(\vec{x}) =$ probability that event is from data

$1 - f(\vec{x}) =$ probability event is from GEANT

Density estimation → Classification!

Iterative reweighting: Step 1, iteration 1

Detector-level



$$f(\vec{x}_{unlikely}) = 0$$

$$\text{loss}\left(f(\vec{x}_{unlikely})\right) = 0$$

Weights: $w_1(x) = d(x)/g(x)$ Ok for the binned case

$\approx f(x)/(1 - f(x))$ Using Bayes' Theorem; See derivation in backup

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

to distinguish events coming from data vs from PYTHIA+GEANT

$$\text{loss}(f(x)) = -\sum_{x \in data} \log(f(x)) - \sum_{x \in geant} \log(1 - f(x))$$

Sanity checks:

- Correct classification minimizes the loss function
- Unlikely events get weighted down

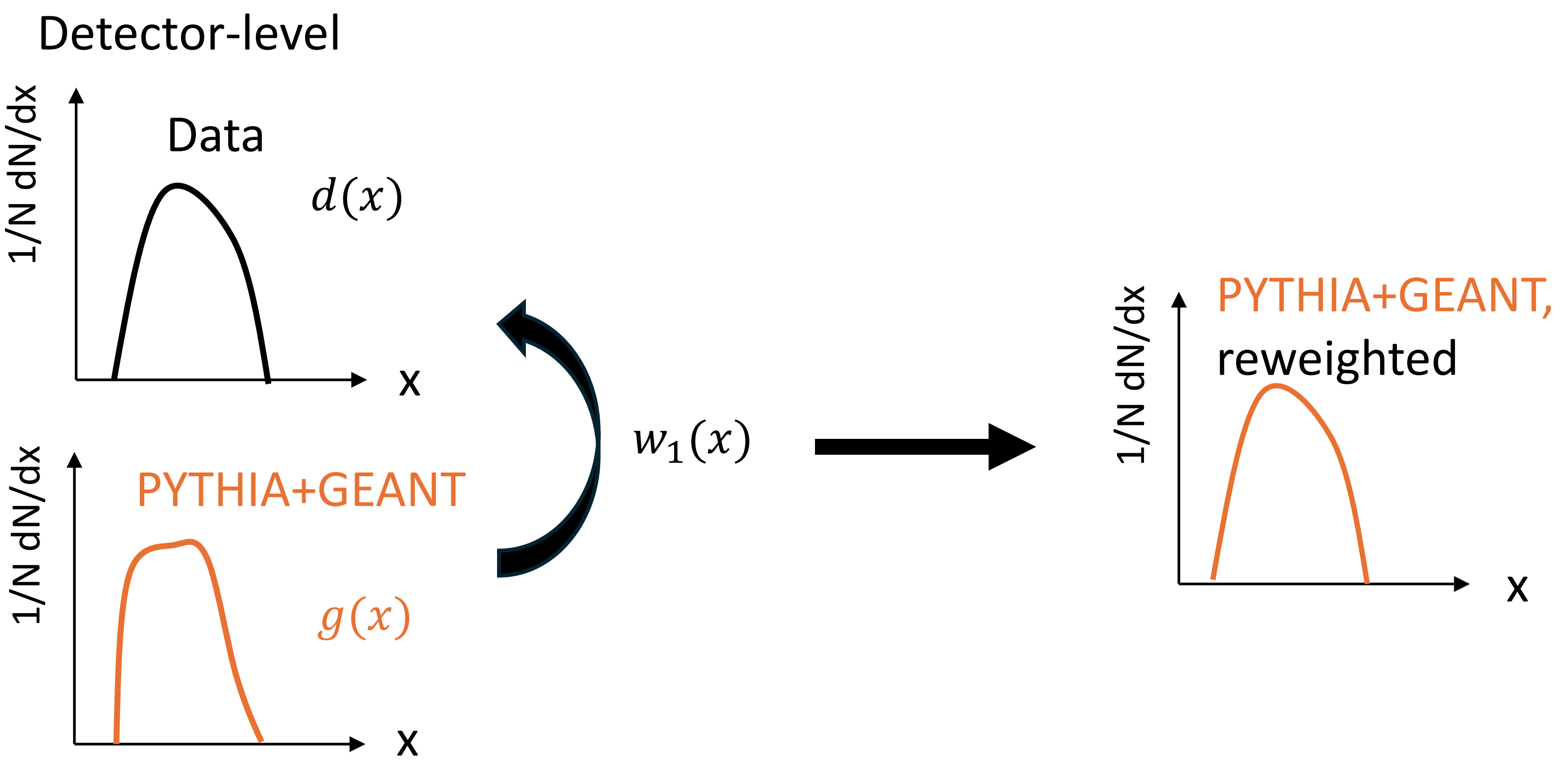
NN output

$f(\vec{x}) =$ probability that event is from data

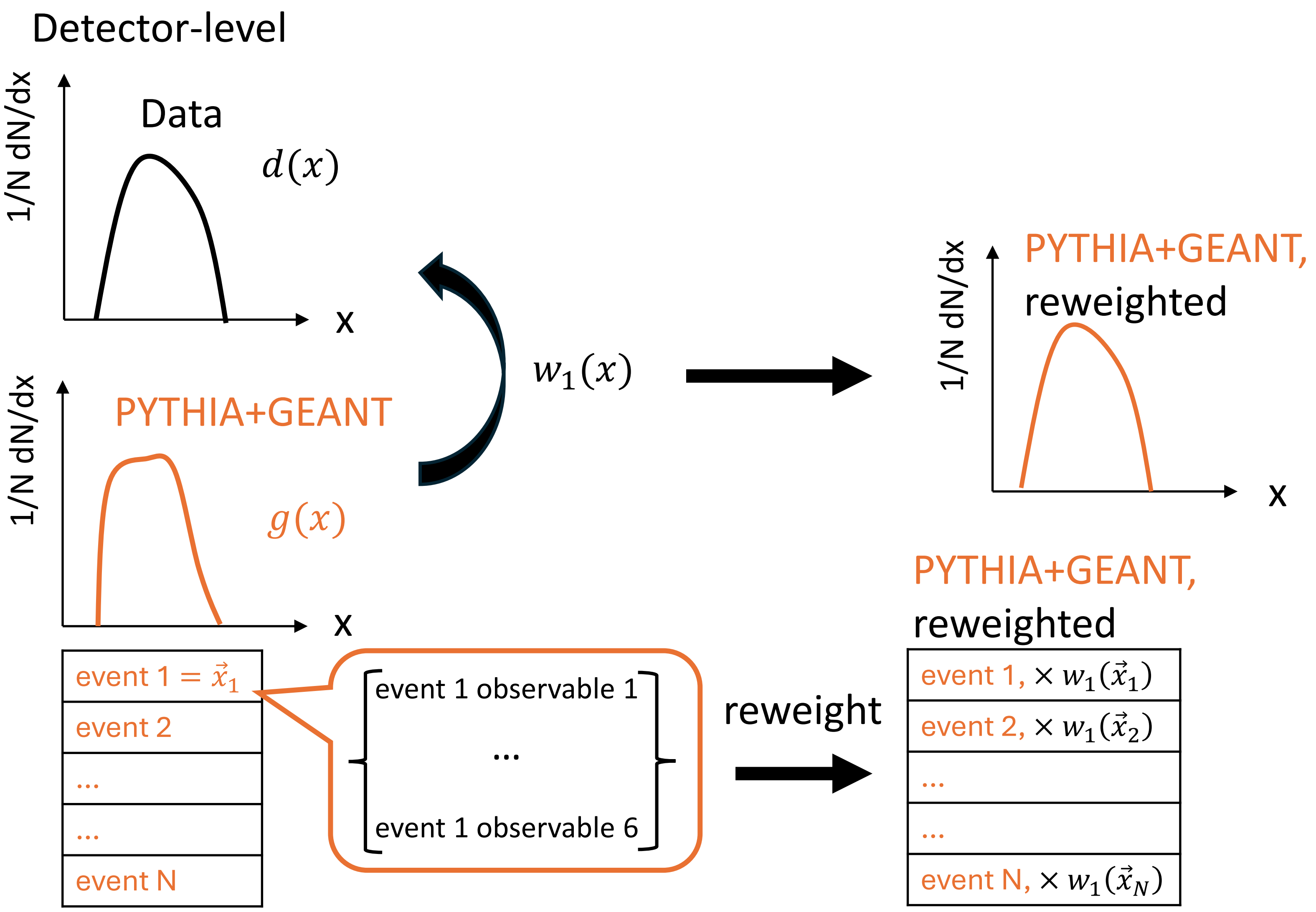
$1 - f(\vec{x}) =$ probability event is from GEANT

Density estimation → Classification!

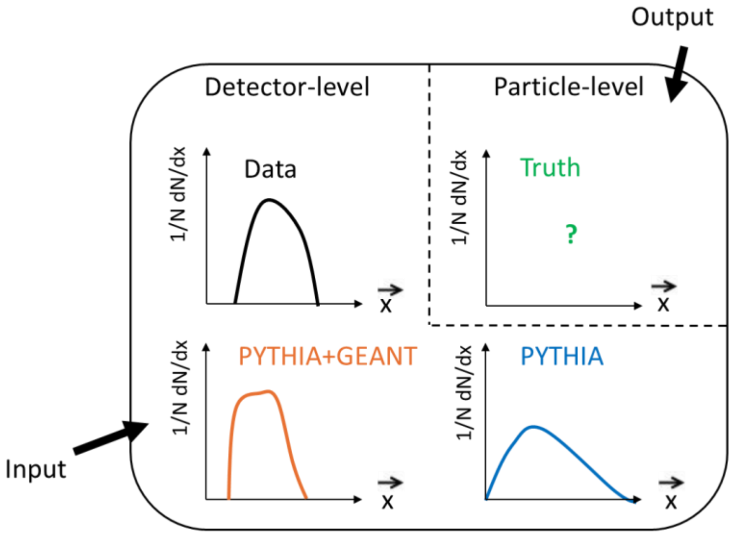
Iterative reweighting: Step 1, iteration 1

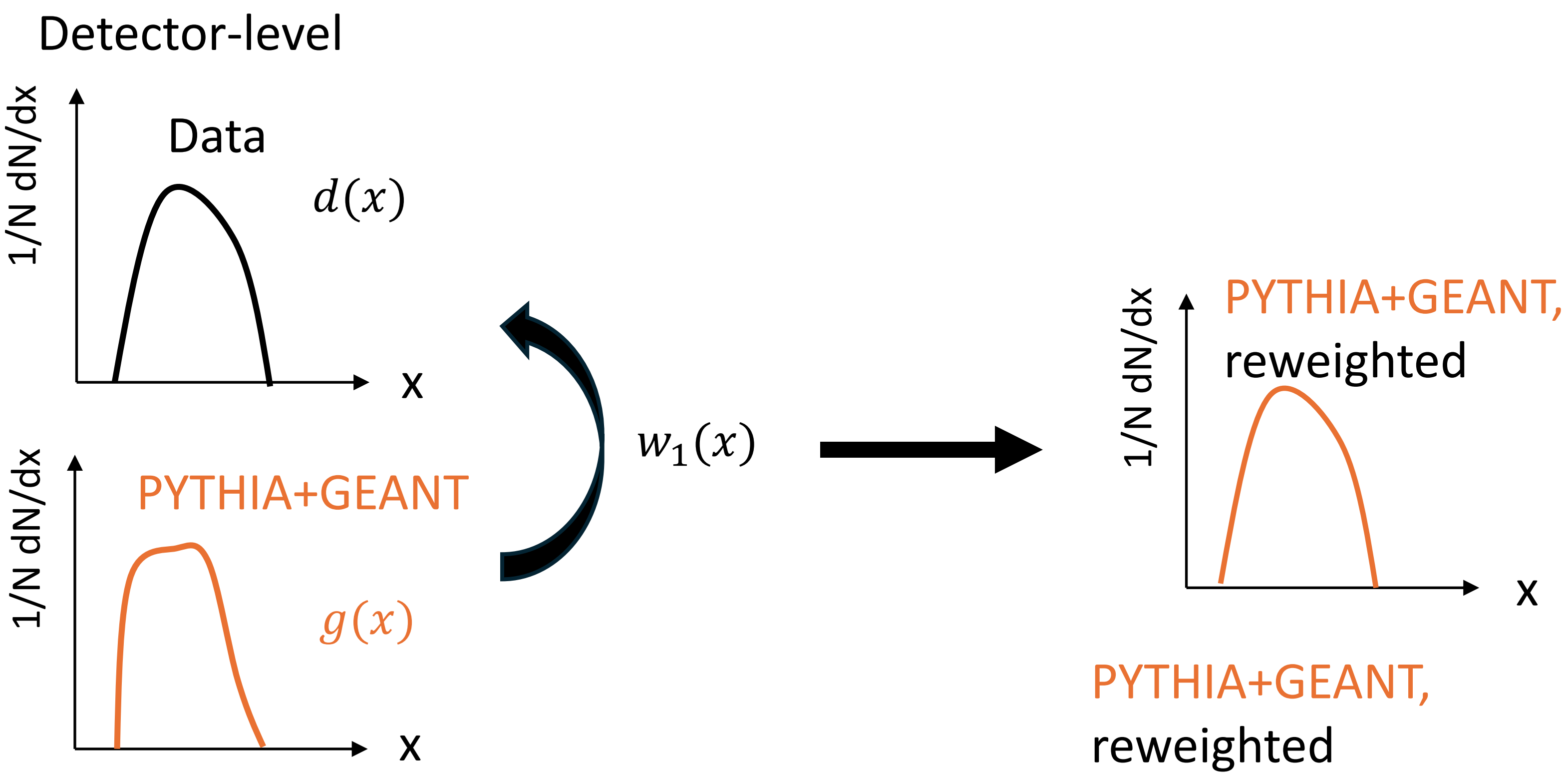


Iterative reweighting: Step 1, iteration 1



Iterative reweighting: Step 1, iteration 1





event 1 = $\vec{x}_1$
event 2
…
…
event N

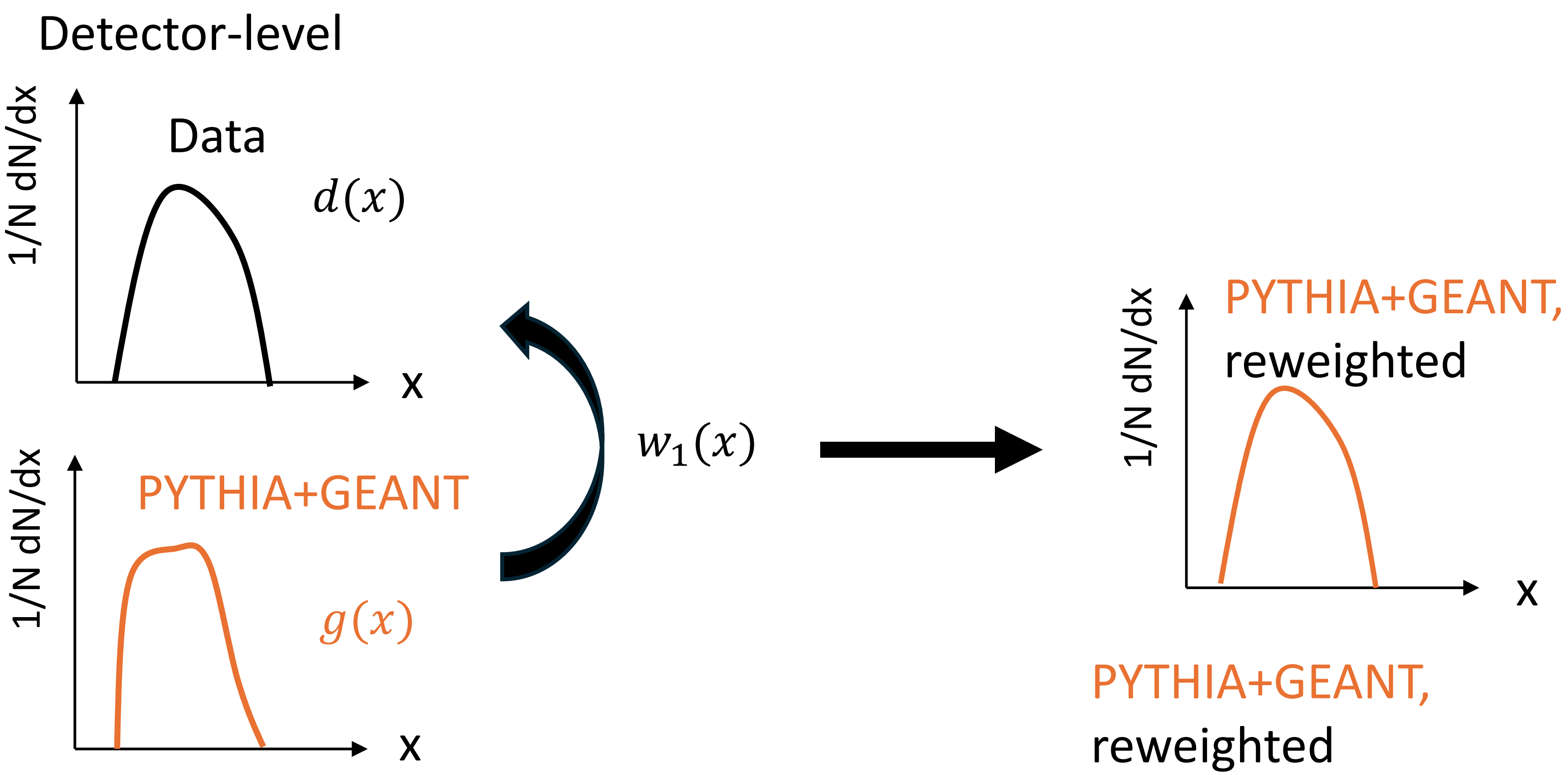
$$\left[\begin{array}{c} \text{event 1 observable 1} \\ \ldots \\ \text{event 1 observable 6} \end{array}\right]$$

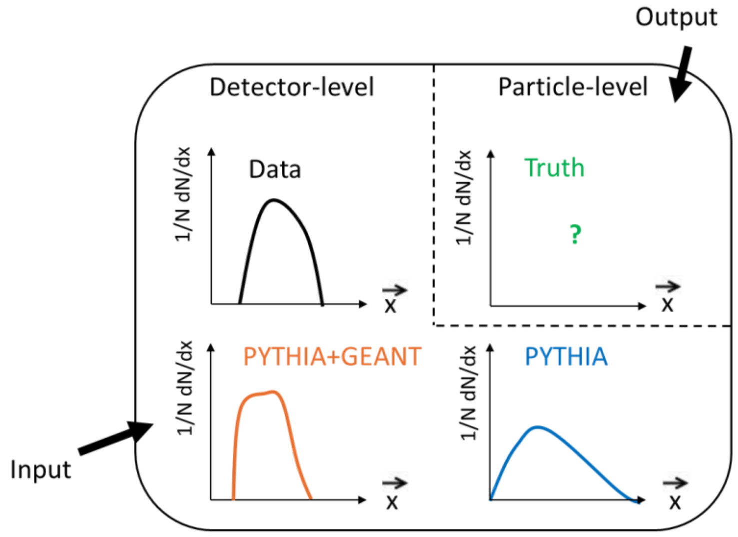
reweight

PYTHIA+GEANT,
reweighted

event 1, $\times\, w_1(\vec{x}_1)$
event 2, $\times\, w_1(\vec{x}_2)$
…
…
event N, $\times\, w_1(\vec{x}_N)$

Iterative reweighting: Step 1, iteration 1





event 1 $= \vec{x}_1$
event 2
…
…
event N

$$\left\{ \begin{array}{c} \text{event 1 observable 1} \\ \dots \\ \text{event 1 observable 6} \end{array} \right\}$$

reweight

PYTHIA+GEANT, reweighted

event 1, $\times w_1(\vec{x}_1)$
event 2, $\times w_1(\vec{x}_2)$
…
…
event N, $\times w_1(\vec{x}_N)$

event matching

PYTHIA+GEANT, reweighted	PYTHIA, reweighted
event 1, $\times w_1(\vec{x}_1)$	event 1, $\times w_1(\vec{x}_1)$
event 2, $\times w_1(\vec{x}_2)$	event 2, $\times w_1(\vec{x}_2)$?
…	…
…	…
event N, $\times w_1(\vec{x}_N)$	event N, $\times w_1(\vec{x}_N)$

Iterative reweighting: Step 2, iteration 1

- Detector response is stochastic
  - Two identical **particle-level** events might not get mapped to identical **detector-level** events

PYTHIA+GEANT, reweighted	PYTHIA, reweighted
event 1, $\times\, w_1(\vec{x}_1)$	event 1, $\times\, w_1(\vec{x}_1)$
event 2, $\times\, w_1(\vec{x}_2)$	event 2, $\times\, w_1(\vec{x}_2)$?
…	…
…	…
event N, $\times\, w_1(\vec{x}_N)$	event N, $\times\, w_1(\vec{x}_N)$

Iterative reweighting: Step 2, iteration 1

- Detector response is stochastic
  - Two identical **particle-level** events might not get mapped to identical **detector-level** events

- $w_1(x)$ is a weighting function of **detector-level** events
- Want $v_1(y)$, a weighting function of **particle-level** events

PYTHIA+GEANT, reweighted	PYTHIA, reweighted
event 1, $\times\, w_1(\vec{x}_1)$	event 1, $\times\, w_1(\vec{x}_1)$
event 2, $\times\, w_1(\vec{x}_2)$	event 2, $\times\, w_1(\vec{x}_2)$
…	…
…	…
event N, $\times\, w_1(\vec{x}_N)$	event N, $\times\, w_1(\vec{x}_N)$

Iterative reweighting: Step 2, iteration 1

- Detector response is stochastic
  - Two identical **particle-level** events might not get mapped to identical **detector-level** events

- $w_1(x)$ is a weighting function of **detector-level** events
- Want $v_1(y)$, a weighting function of **particle-level** events

$v_1(y)$

PYTHIA

event 1 = $\vec{y}_1$
event 2
…
…
event N

PYTHIA+GEANT, reweighted	PYTHIA, reweighted
event 1, $\times\, w_1(\vec{x}_1)$	event 1, $\times\, w_1(\vec{x}_1)$
event 2, $\times\, w_1(\vec{x}_2)$	event 2, $\times\, w_1(\vec{x}_2)$
…	…
…	…
event N, $\times\, w_1(\vec{x}_N)$	event N, $\times\, w_1(\vec{x}_N)$

Iterative reweighting: Step 2, iteration 1

- Detector response is stochastic
  - Two identical **particle-level** events might not get mapped to identical **detector-level** events

- $w_1(x)$ is a weighting function of **detector-level** events
- Want $v_1(y)$, a weighting function of **particle-level** events

  $\approx h(y)/(1-h(y))$,

  where $h(y)$ is a neural network and trained with the binary cross-entropy loss function

$v_1(y)$

PYTHIA

event 1 $= \vec{y}_1$
event 2
…
…
event N

PYTHIA+GEANT, reweighted	PYTHIA, reweighted
event 1, $\times\, w_1(\vec{x}_1)$	event 1, $\times\, w_1(\vec{x}_1)$
event 2, $\times\, w_1(\vec{x}_2)$	event 2, $\times\, w_1(\vec{x}_2)$
…	…
…	…
event N, $\times\, w_1(\vec{x}_N)$	event N, $\times\, w_1(\vec{x}_N)$

Iterative reweighting: Step 2, iteration 1

PYTHIA, w/ weights from step 1

event 1, $\times\, w_1(\vec{x}_1)$
event 2, $\times\, w_1(\vec{x}_2)$
…
…
event N, $\times\, w_1(\vec{x}_N)$

PYTHIA

event 1 $= \vec{y}_1$
event 2
…
…
event N

$v_1(y)$

PYTHIA, w/ proper weighting function

event 1, $\times\, v_1(\vec{y}_1)$
event 2, $\times\, v_1(\vec{y}_2)$
…
…
event N, $\times\, v_1(\vec{y}_N)$

Unfolding result after 1 iteration

Iterative reweighting: Step 2, iteration 1

PYTHIA, w/ weights from step 1

event 1, $\times\, w_1(\vec{x}_1)$
event 2, $\times\, w_1(\vec{x}_2)$
…
…
event N, $\times\, w_1(\vec{x}_N)$

PYTHIA

event 1 $= \vec{y}_1$
event 2
…
…
event N

$v_1(y)$

PYTHIA, w/ proper weighting function

event 1, $\times\, v_1(\vec{y}_1)$
event 2, $\times\, v_1(\vec{y}_2)$
…
…
event N, $\times\, v_1(\vec{y}_N)$

Unfolding result after 1 iteration

- $v_1(y)$ used to reweight both **particle** and **detector-level** events in iteration 2
  - $w_2(x)$, $v_2(y)$, …

Iterative reweighting: Result

- Result: **Particle-level** events, reweighted by $\nu_N(y)$ – step 2 output of the last iteration

Iterative reweighting: Result

- Result: **Particle-level** events, reweighted by $\nu_N(y)$ – step 2 output of the last iteration
- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - Machine learning driven
    - Unbinned
    - **Simultaneously unfolds many observables**

Iterative reweighting: Result

- Result: **Particle-level** events, reweighted by $v_N(y)$ – step 2 output of the last iteration
- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - ✓ Machine learning driven
    - Unbinned
    - **Simultaneously unfolds many observables**

Iterative reweighting: Result

- Result: **Particle-level** events, reweighted by $\nu_N(y)$ – step 2 output of the last iteration
- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - ✓ Machine learning driven
    - ✓ Unbinned → reweighting is done event-by-event
    - **Simultaneously unfolds many observables**

Iterative reweighting: Result

- Result: **Particle-level** events, reweighted by $\nu_N(y)$ – step 2 output of the last iteration
- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - ✓ Machine learning driven
    - ✓ Unbinned → reweighting is done event-by-event
    - ✓ **Simultaneously unfolds many observables** → can adjust the input dimension of neural networks

Iterative reweighting: Result

- Result: **Particle-level** events, reweighted by $\nu_N(y)$ – step 2 output of the last iteration
- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - ✓ Machine learning driven
    - ✓ Unbinned → reweighting is done event-by-event
    - ✓ **Simultaneously unfolds many observables** → can adjust the input dimension of neural networks

Appendix: OmniFold as a Maximum Likelihood Estimate

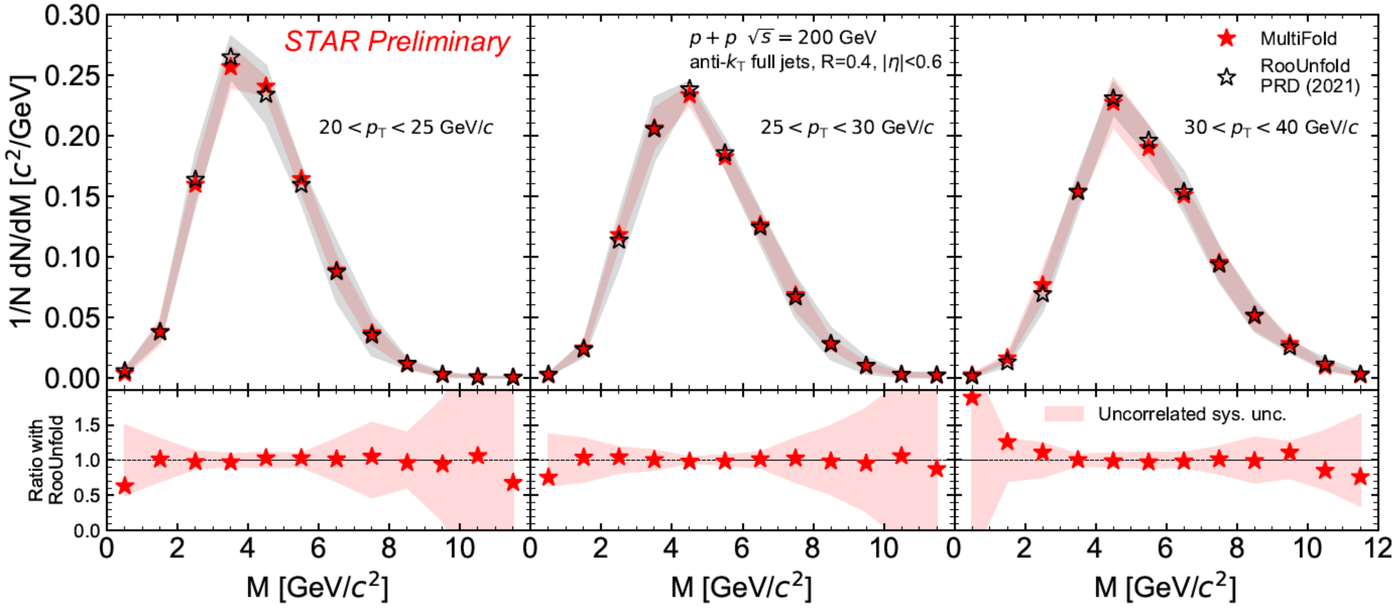
In this Appendix, we review the statistical underpinnings of Iterative Bayesian Unfolding (IBU) [5] as well as OmniFold and confirm that they converge to the maximum likelihood estimate of the true particle-level distribution.

Iterative reweighting: Result

- Result: **Particle-level** events, reweighted by $\nu_N(y)$ – step 2 output of the last iteration
- Unfolding methods:
  - Iterative Bayesian unfolding (D'Agostini. arXiv:1010.0632(2010))
  - **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
    - ✓ Machine learning driven
    - ✓ Unbinned → reweighting is done event-by-event
    - ✓ **Simultaneously unfolds many observables** → can adjust the input dimension of neural networks

Appendix: OmniFold as a Maximum Likelihood Estimate

In this Appendix, we review the statistical underpinnings of Iterative Bayesian Unfolding (IBU) [5] as well as OmniFold and confirm that they converge to the maximum likelihood estimate of the true particle-level distribution.



arxiv: 2307.07718

Good agreement between MultiFold and RooUnfold verified with data.

Conclusions

- **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
  - ✓ Machine learning driven
  - ✓ Unbinned → reweighting is done event-by-event
  - ✓ **Simultaneously unfolds many observables** → can adjust the input dimension of neural networks

Conclusions

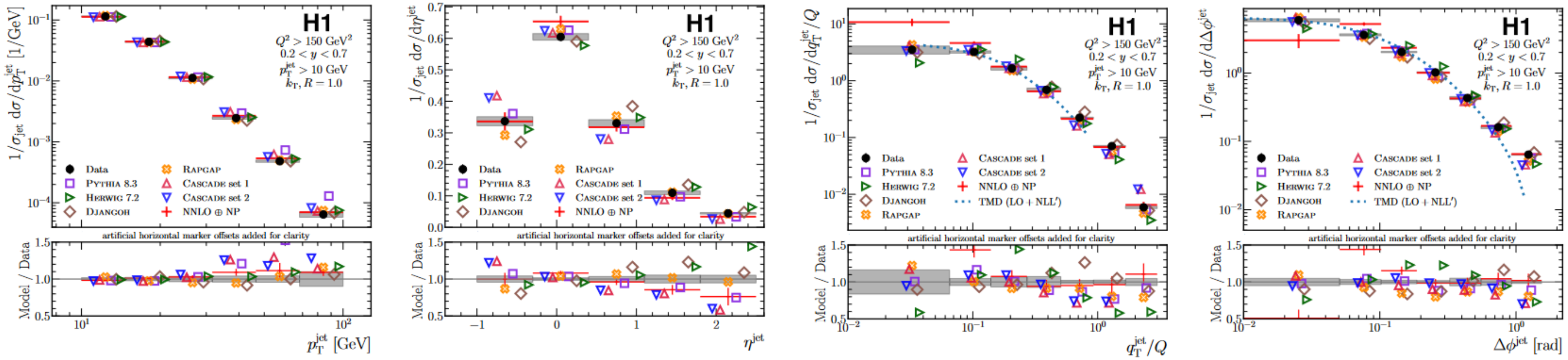
- **MultiFold** (Andreassen et al. PRL 124, 182001 (2020))
  - ✓ Machine learning driven
  - ✓ Unbinned → reweighting is done event-by-event
  - ✓ **Simultaneously unfolds many observables** → can adjust the input dimension of neural networks

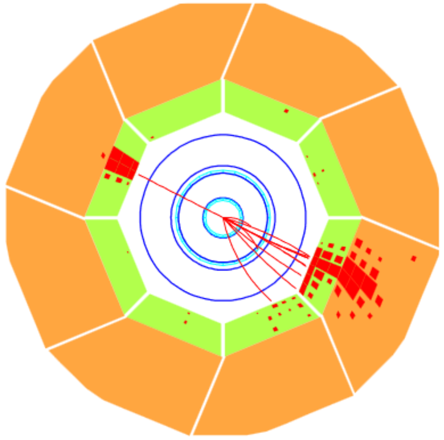
- Resources readily available, e.g., https://github.com/ericmetodiev/OmniFold and https://github.com/hep-lbdl/OmniFold
- Successful applications in H1, STAR, LHCb, ATLAS and CMS
- Easy access to correlation information among observables
- Promising potential for multi-differential measurements

Backup

Applications

- Probing transverse-momentum dependent (TMD) parton distribution functions at H1





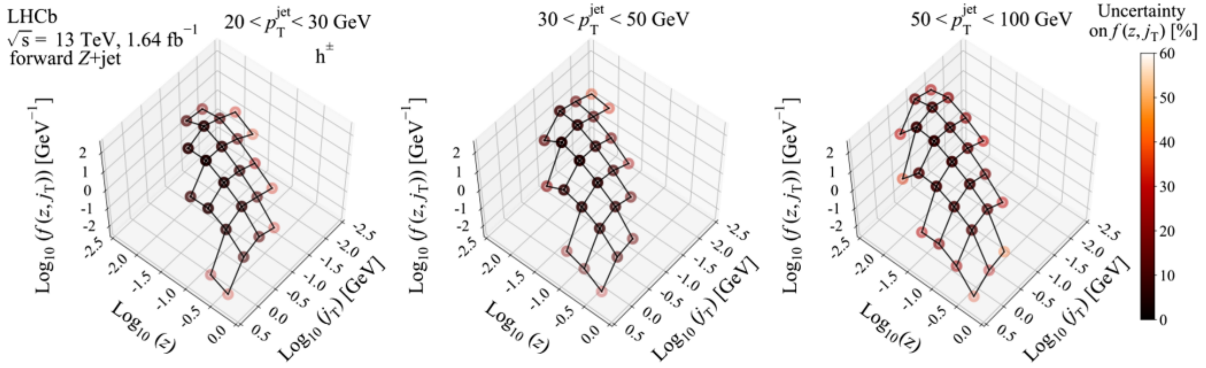
[Andreev et al. PRL 128, 132002 (2022)]

Simultaneously correct for:

· Jet p_T · Jet η · Jet ϕ · Electron p_T · Electron p_z · Electron-jet imbalance · Electron-jet azimuthal angle correlation

- Probing TMD jet fragmentation functions at LHCb

[Aaij et al. Phys. Rev. D 108, L031103 (2023)]



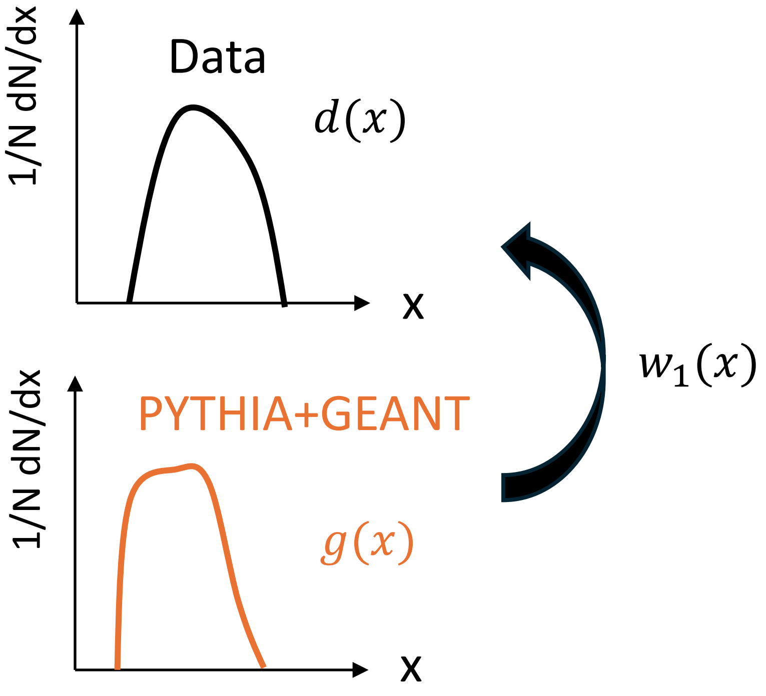
- Similar measurement ongoing at STAR, see talk by Hannah Harrison-Smith

Simultaneously correct for:

· Jet p_T · Jet η · Hadron in jet longitudinal momentum fraction · Hadron momentum wrt jet axis

Iterative reweighting: Step 1, iteration 1

Detector-level



$$w_1(x) = d(x)/g(x)$$

$$= \frac{P(\vec{x}\,|\,\mathrm{data})}{P(\vec{x}\,|\,\mathrm{geant})}$$

$$= \frac{P(\mathrm{data}\,|\,\vec{x})\cdot \cancel{P(\vec{x})}}{P(\mathrm{data})} \cdot \frac{P(\mathrm{geant})}{P(\mathrm{geant}\,|\,\vec{x})\cdot \cancel{P(\vec{x})}}$$

$$= \frac{P(\mathrm{data}\,|\,\vec{x})}{P(\mathrm{geant}\,|\,\vec{x})} \cdot \boxed{\frac{P(\mathrm{geant})}{P(\mathrm{data})}}$$ → Normalized to 1

$$= \frac{\text{probability that } \vec{x} \text{ is from data}}{\text{probability that } \vec{x} \text{ is from GEANT}}$$

$$\approx f(x)/(1-f(x))$$

Using Bayes' Theorem

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

Derivation from Chapter 4, Probabilistic classification, of M. Sugiyama, T. Suzuki, and T. Kanamori, Density Ratio Estimation in Machine Learning (Cambridge University Press, 2012).

where $f(x)$ is a neural network and trained with the binary cross-entropy loss function

Iterative reweighting: Step 1, iteration 2

Data

event 1
event 2
…
…
event N

PYTHIA+GEANT,
w/ weights from iteration 1

event 1, $\times\, v_1(\vec{y}_1)$
event 2, $\times\, v_1(\vec{y}_2)$
…
…
event N, $\times\, v_1(\vec{y}_N)$

$w_2(x)$

PYTHIA+GEANT,
reweighted to data

event 1, $\times\, w_2(\vec{x}_1)$
event 2, $\times\, w_2(\vec{x}_2)$
…
…
event N, $\times\, w_2(\vec{x}_N)$

Iterative reweighting: Step 2, iteration 2

PYTHIA, w/ weights from step 1, iteration 2

event 1, $\times\, w_2(\vec{x}_1)$
event 2, $\times\, w_2(\vec{x}_2)$
…
…
event N, $\times\, w_2(\vec{x}_N)$

PYTHIA, w/ weights from step 2, iteration 1

event 1, $\times\, v_1(\vec{y}_1)$
event 2, $\times\, v_1(\vec{y}_2)$
…
…
event N, $\times\, v_1(\vec{y}_N)$

$v_2(y)$

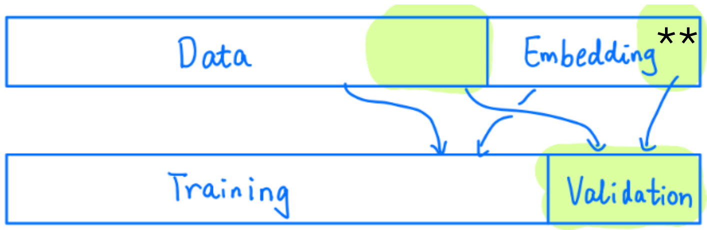
PYTHIA, w/ proper weighting function

event 1, $\times\, v_2(\vec{y}_1)$
event 2, $\times\, v_2(\vec{y}_2)$
…
…
event N, $\times\, v_2(\vec{y}_N)$

Unfolding result after 2 iterations

Iterative reweighting: Iteration n

- Iteration n, Step 1:

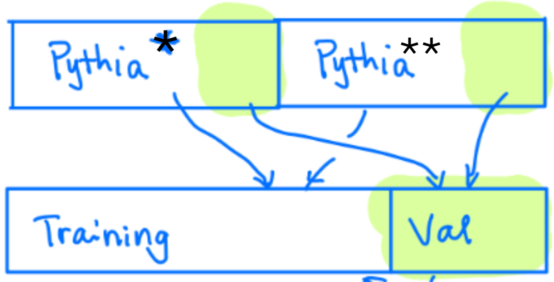


Run model.fit() on ↑ ↳ Decides when training should stop

Run model.predict() on ↺. Obtain weights.

Weights corresponding to Embedding are pulled to step 2.

- Iteration n, Step 2:



Run model.fit() on ↖ ↳ Decides when training should stop

Run model.predict() on ↺. Obtain weights.

Weights corresponding to Pythia are pushed to step 1.

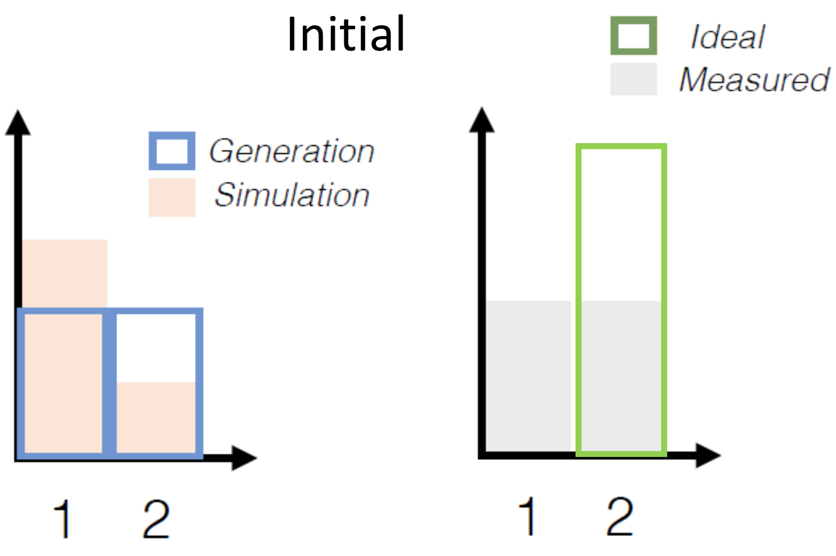
or taken as the final weights if this is the last iteration

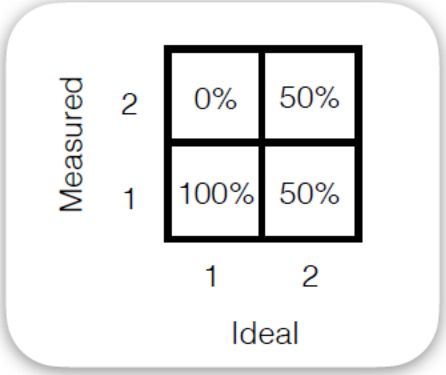
*: (With weights pulled from step 1 of iteration n)

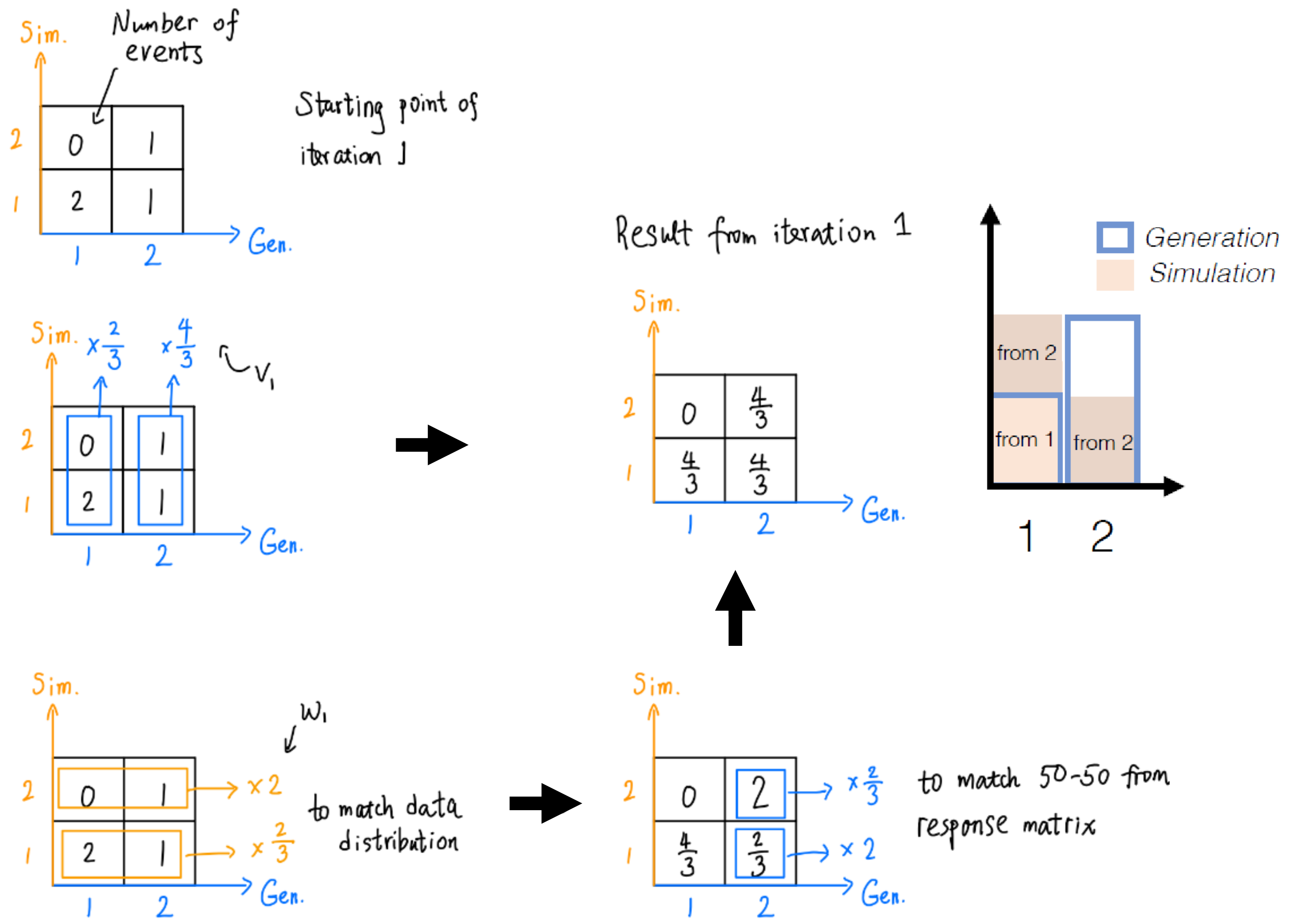
**: (With weights pushed from step 2 of iteration (n-1))

Iterative reweighting: Toy example

- Adapted from slides by Ben Nachman



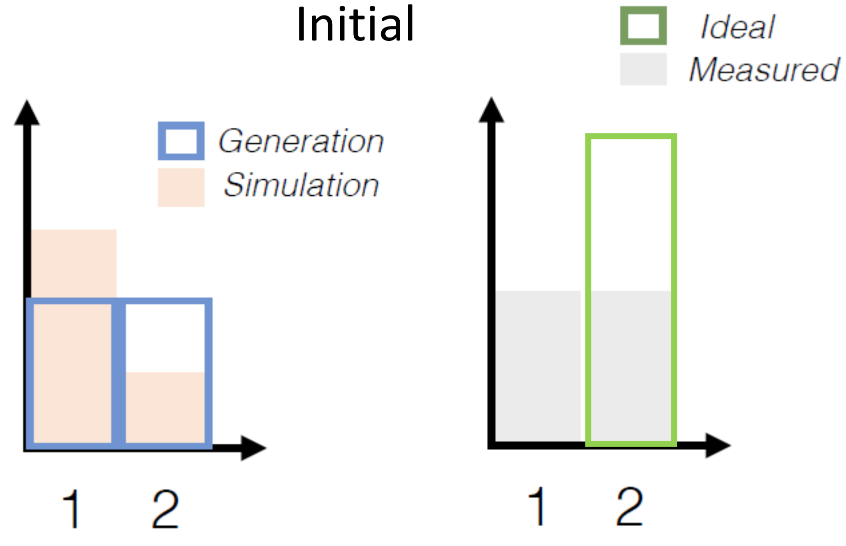


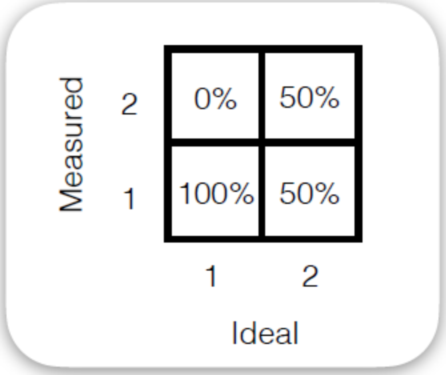


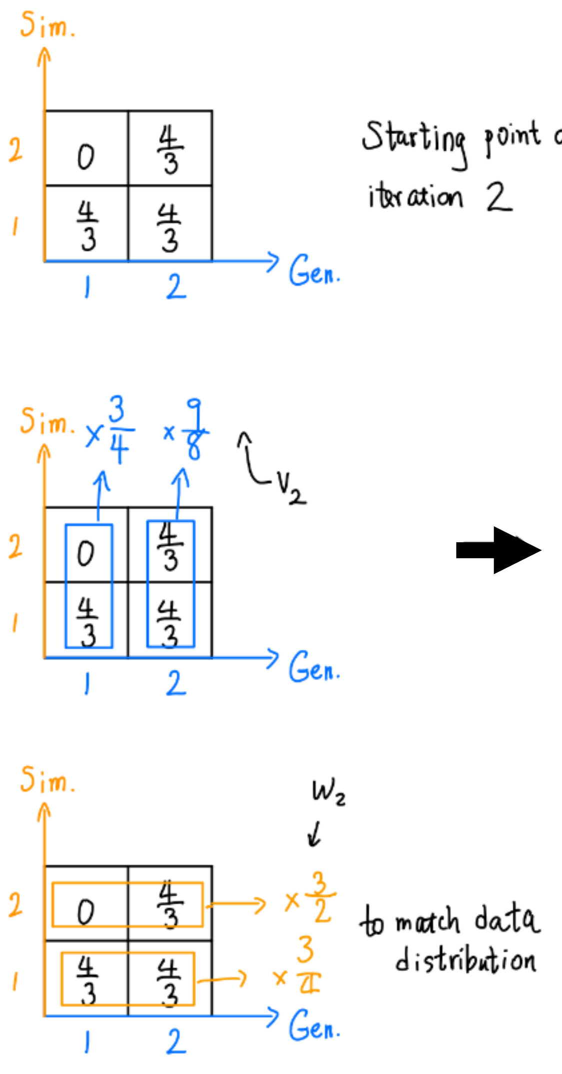
Iterative reweighting: Toy example

- Adapted from slides by Ben Nachman

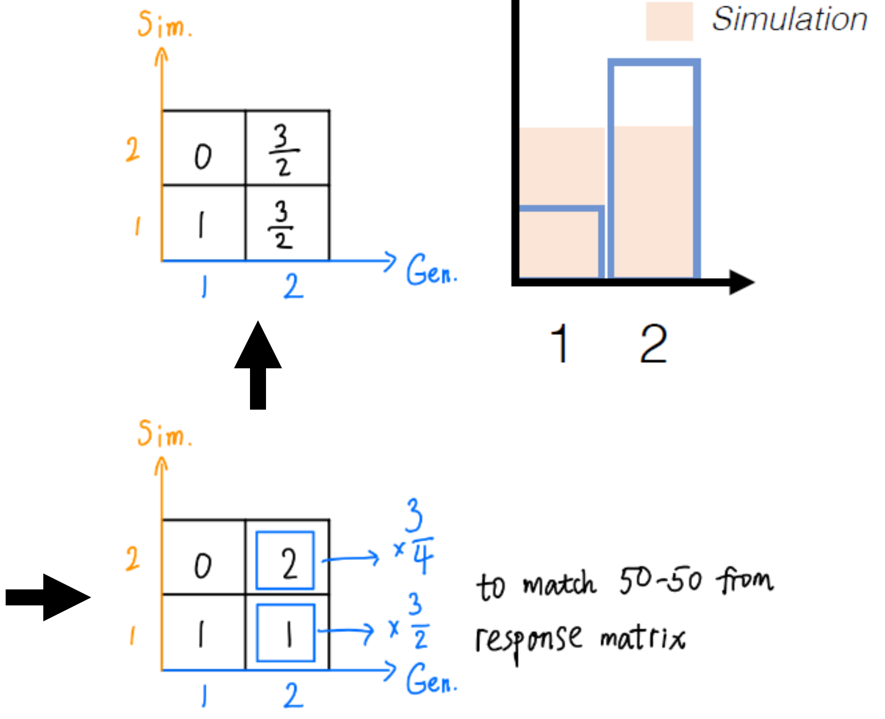
Initial





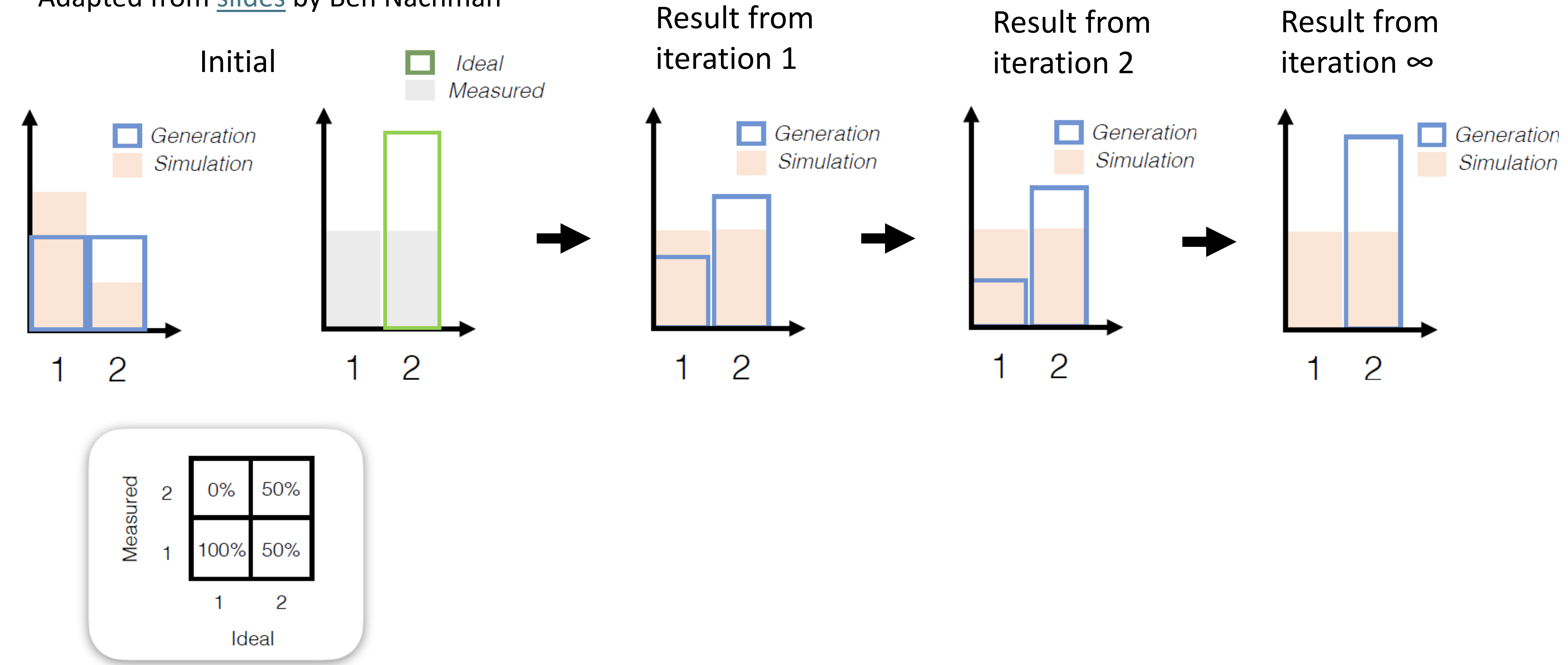


Result from iteration 2



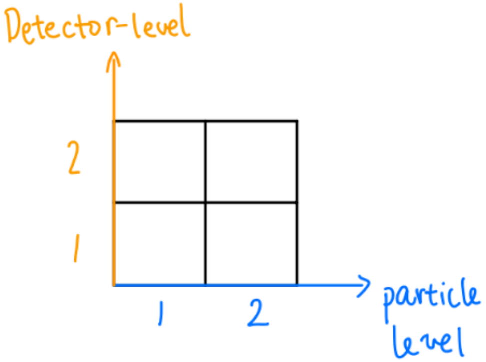
Iterative reweighting: Toy example

- Adapted from slides by Ben Nachman

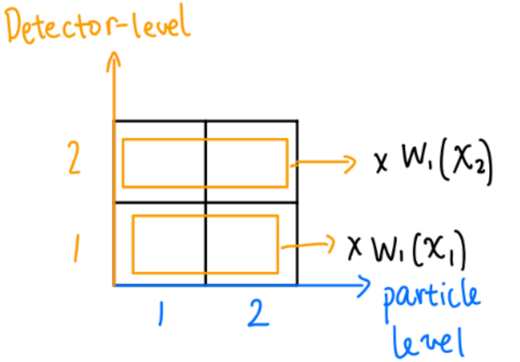


Iterative reweighting

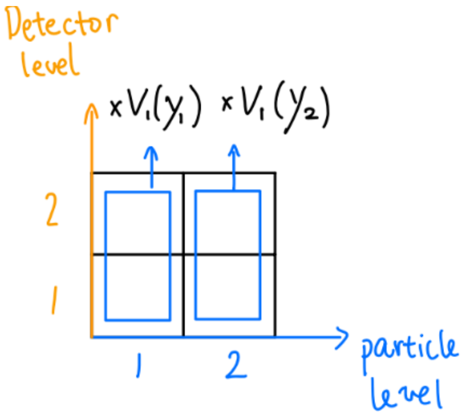
- Why do we iterate?



original



matches data, but breaks the response matrix



fixes the response matrix, but doesn't match data

Challenges

- Computationally expensive
- How to publish an unbinned result? arxiv:2109.13243