Treatment of Detector Systematics via Likelihood-free Inference

NPML Workshop, Oct. 31st, 2025

Alexandra Trettin, University of Manchester
Leander Fischer, DESY
Richard Naab, DESY

Paper: JINST 18 (2023) 10, P10019
GitHub: LeanderFischer/ultrasurfaces

The University of Manchester

TREATMENT OF DETECTOR SYSTEMATICS VIA LIKELIHOOD-FREE INFERENCE

NPML Workshop, Oct. 31st, 2025

Alexandra Trettin, University of Manchester
Leander Fischer, DESY
Richard Naab, DESY

Paper: JINST 18 (2023) 10, P10019
GitHub: LeanderFischer/ultrasurfaces

MONTE-CARLO FORWARD-FOLDING

I'm using a neutrino oscillation fit as an example throughout this talk, but the method is completely general!

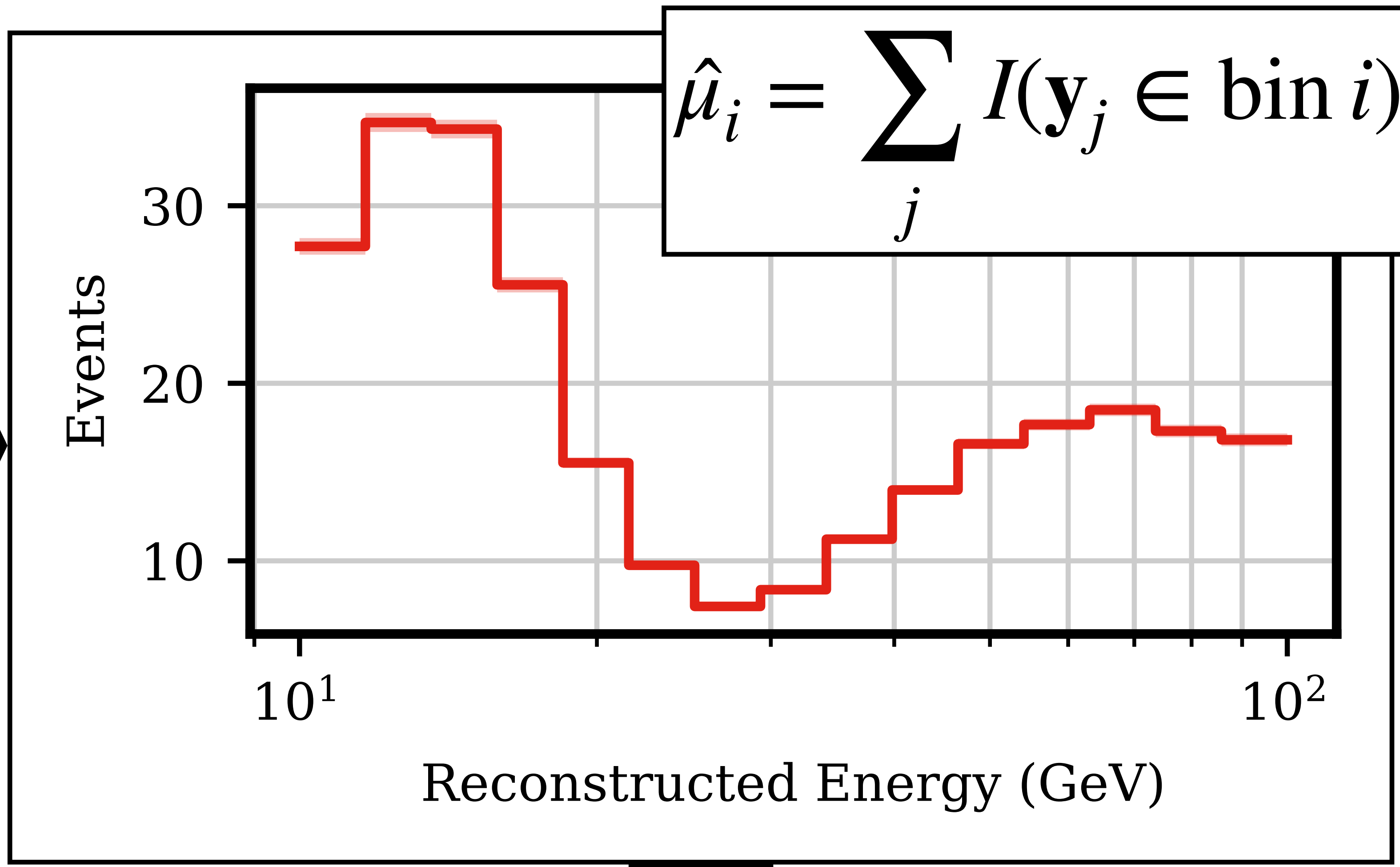
Monte-Carlo

Weight

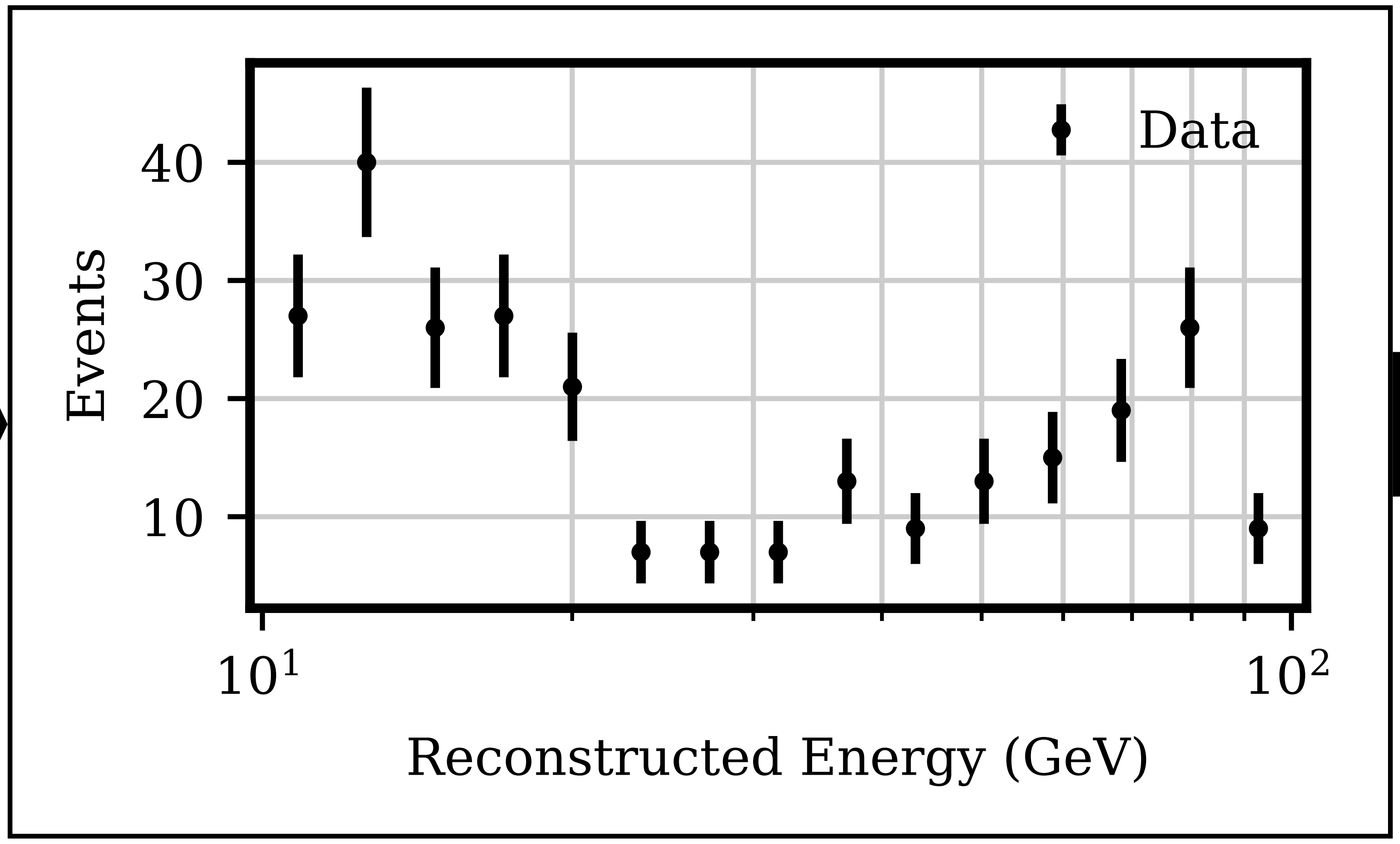
$\mathbf{x}$ = *true neutrino properties*

$$w(\mathbf{x}, \theta) = \underbrace{\frac{\Phi(\mathbf{x}\,|\,\theta_{\text{flux}})}{\Phi_{\text{sim}}(\mathbf{x})} \times \frac{\sigma(\mathbf{x}, \theta_{\text{xsec}})}{\sigma_{\text{sim}}(\mathbf{x})}}_{\textit{nuisance parameters}} \times \ldots \times \underbrace{f\left(\mathbf{x}, \theta_{\text{physics}}\right)}_{\textit{physics}}$$

$$\hat{\mu}_i = \sum_j I(\mathbf{y}_j \in \text{bin } i)\, w_j(\theta)$$



Data



Minimizer

➡ Find parameters θ that optimize test statistic between data and MC histograms

MONTE-CARLO FORWARD-FOLDING

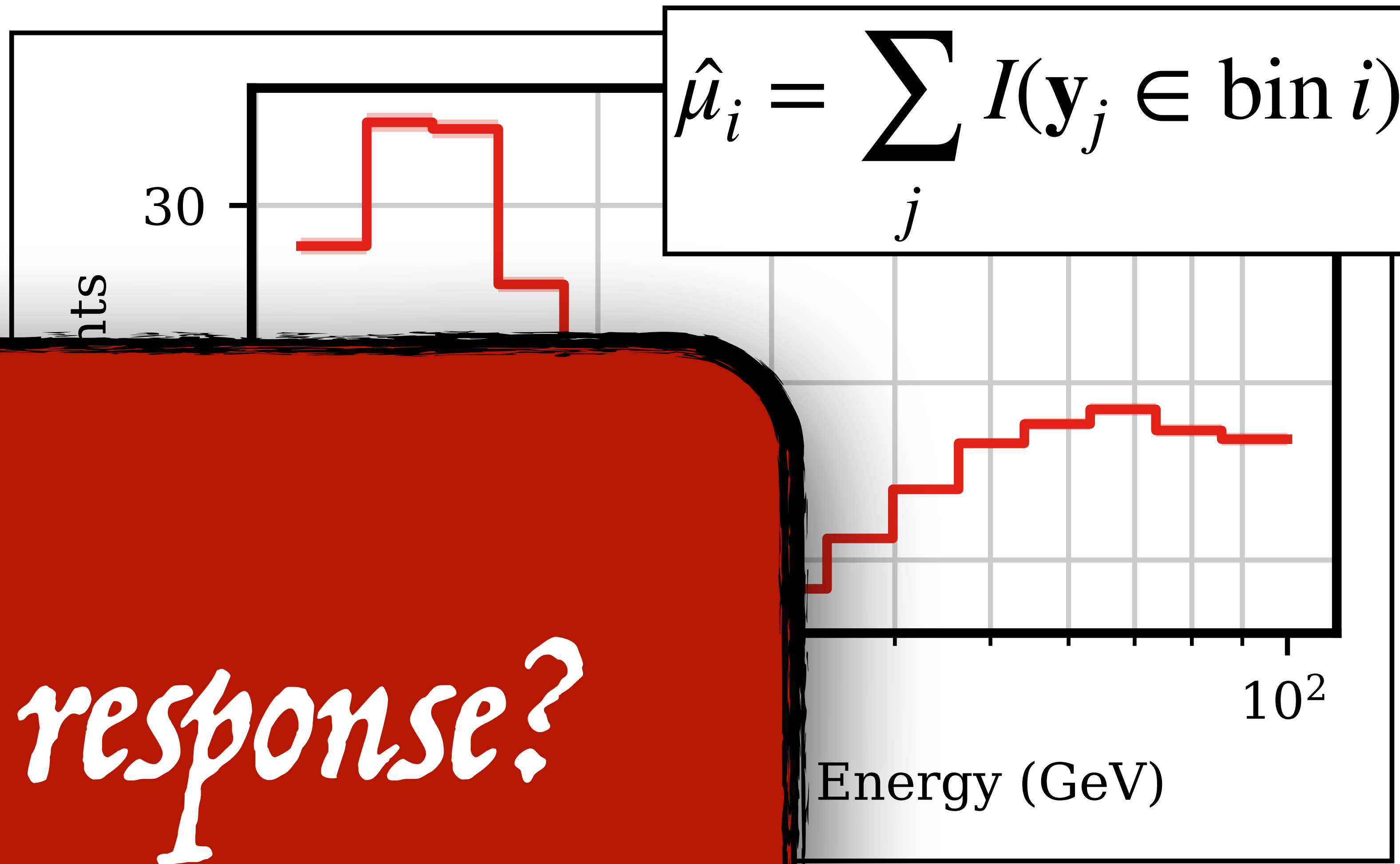
I'm using a neutrino oscillation fit as an example throughout this talk, but the method is completely general!

Monte-Carlo

Weight

$\mathbf{x}$ = *true neutrino properties*

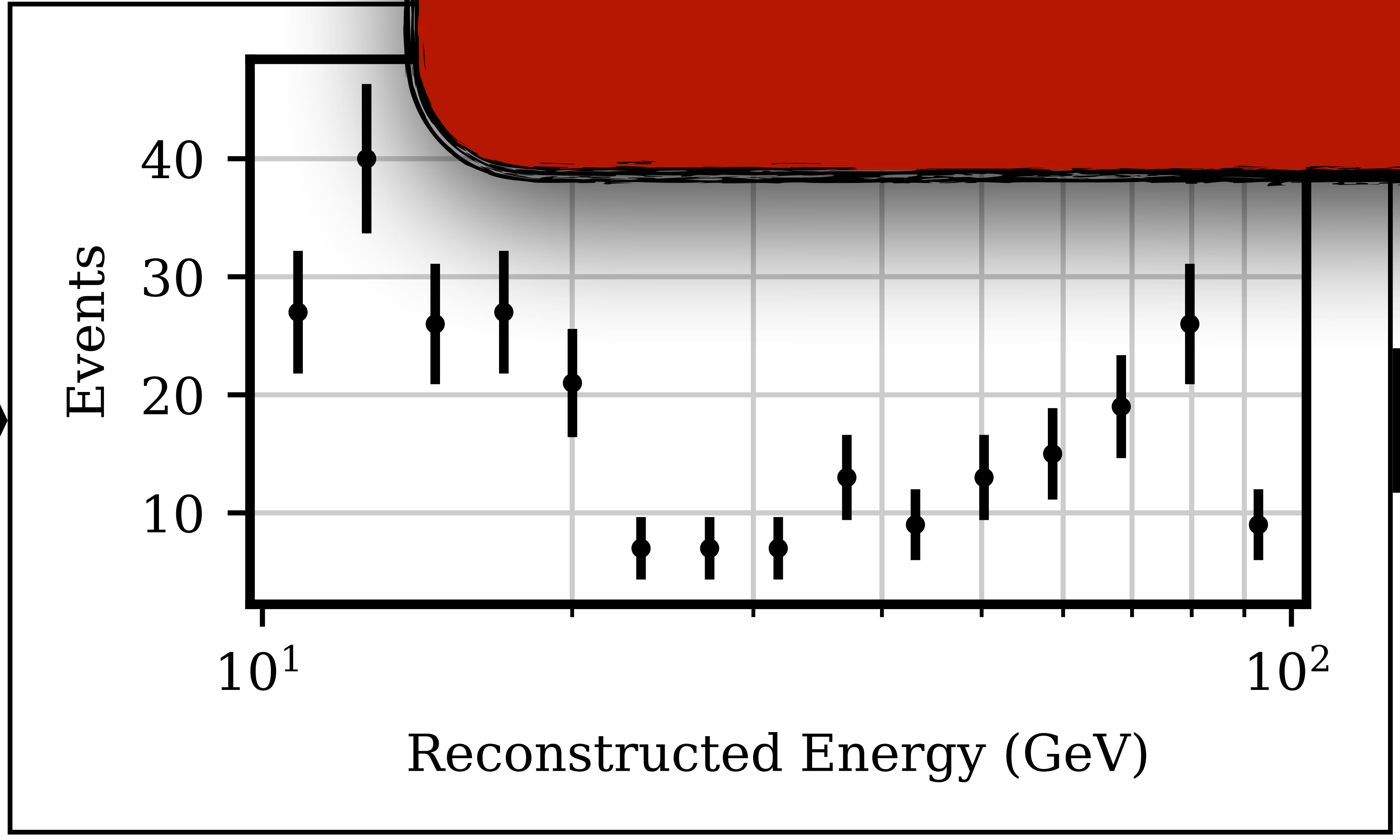
$w(\mathbf{x}, \theta) =$



$$\hat{\mu}_i = \sum_j I(\mathbf{y}_j \in \text{bin } i) w_j(\theta)$$

But where is the detector response?

Data

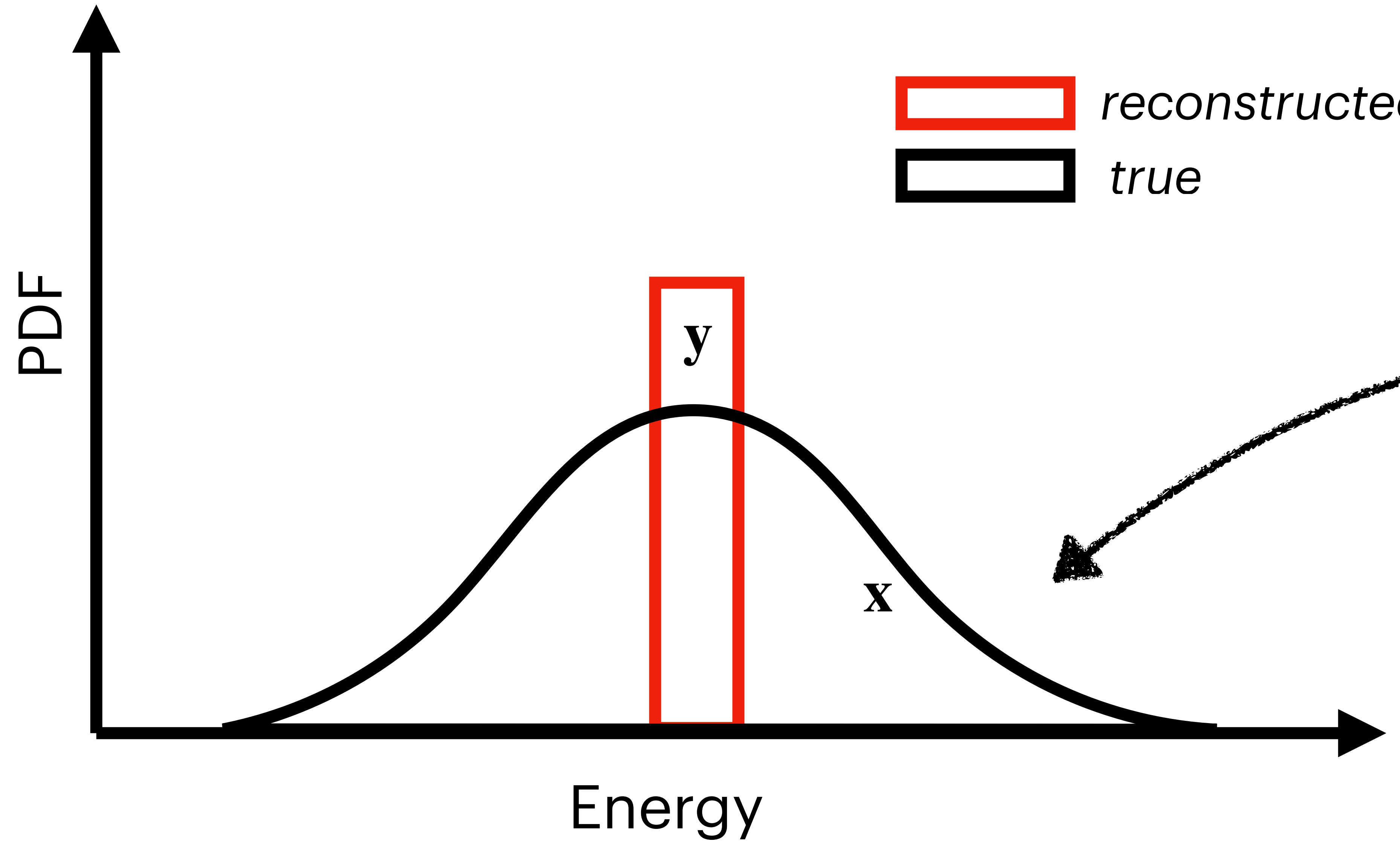


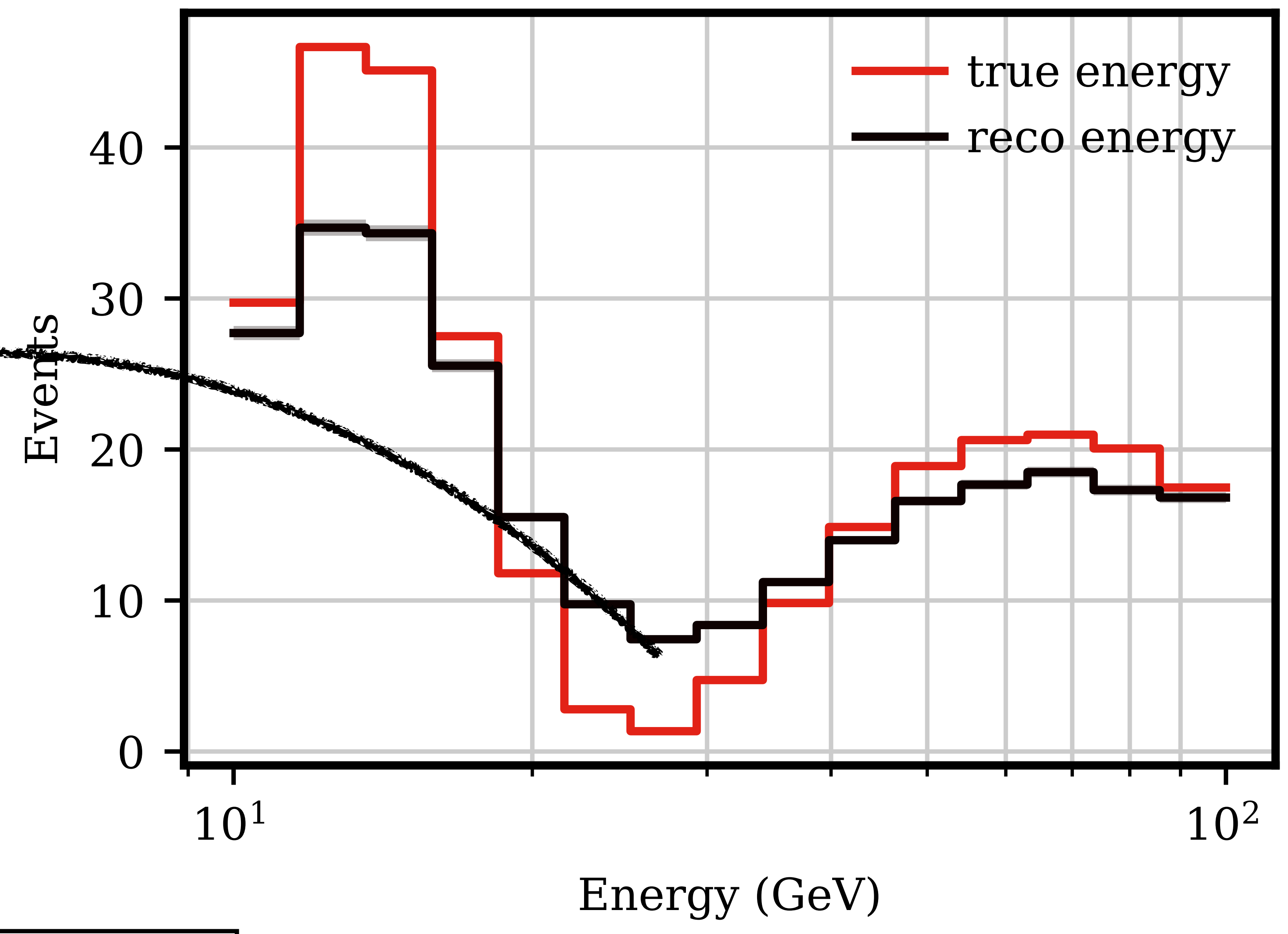
Minimizer

➡ Find parameters θ that optimize test statistic between data and MC histograms

DETECTOR SYSTEMATICS

Oh no, it's all baked in together!





Expectation value in bin i:

α = detector properties

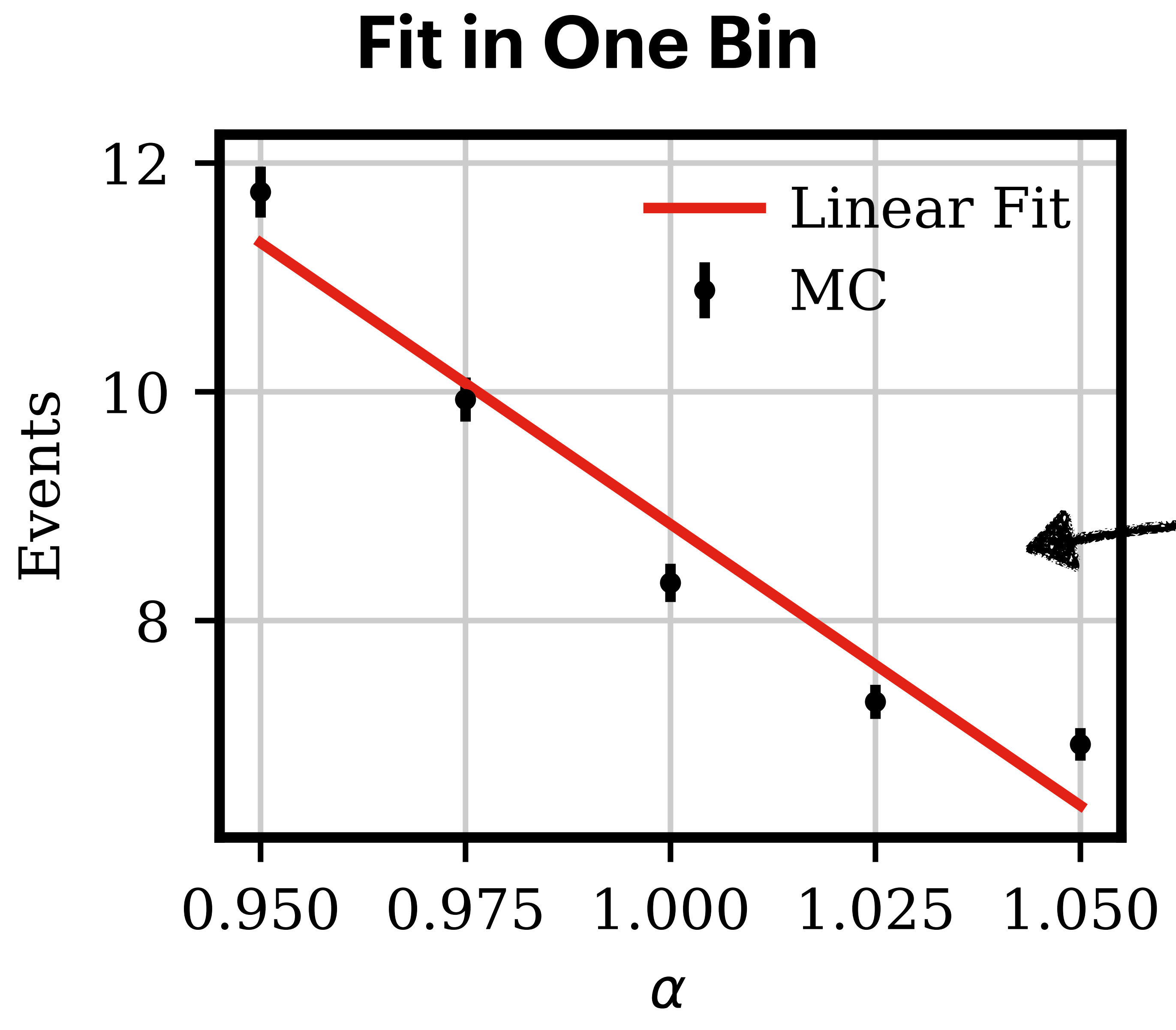
Probability that event with *true* properties $\mathbf{x}$ is accepted

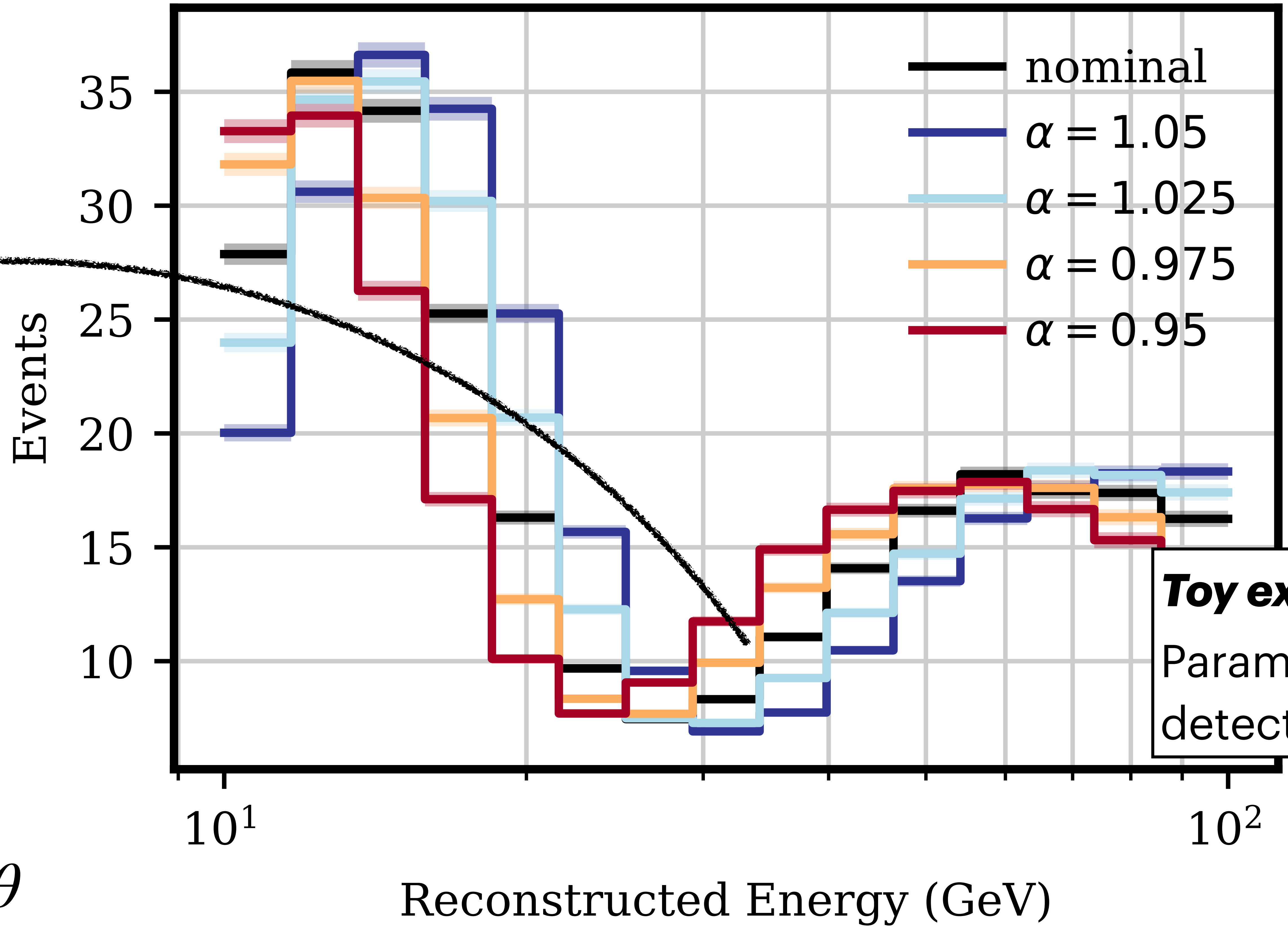
$$\mu_i(\theta) = \int_{\mathbf{y}\in \mathrm{bin}\, i} \mathrm{d}\mathbf{y} \int \mathrm{d}\mathbf{x} P(\mathbf{y}\,|\,\mathbf{x}, \alpha) P(\mathrm{acc}\,|\,\mathbf{x}, \alpha) w(\mathbf{x}, \theta)$$

Probability that event with *true* properties $\mathbf{x}$ is reconstructed with *reconstructed* properties $\mathbf{y}$

Modeling of Detector Effects

Bin-wise weighting method



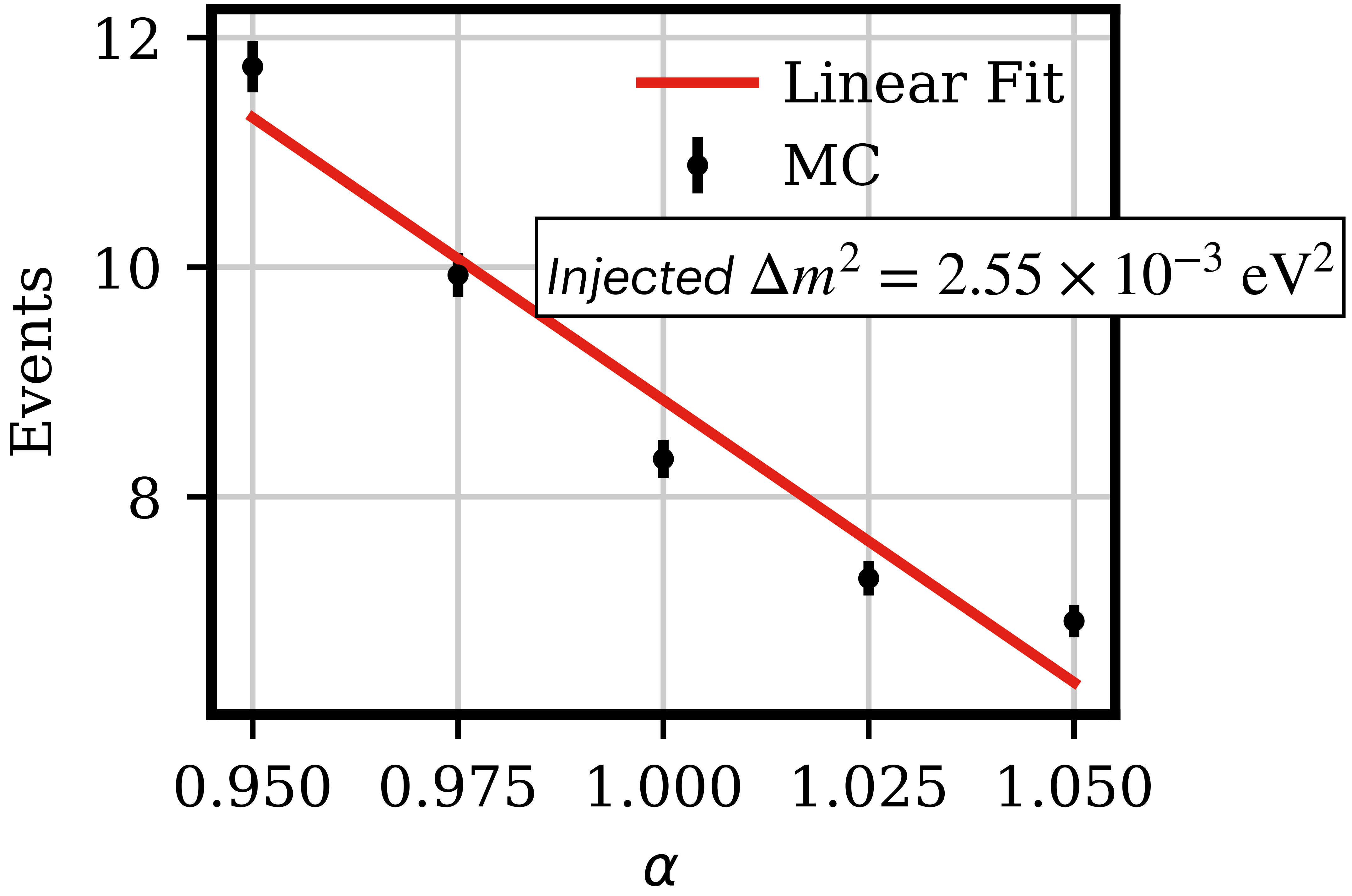


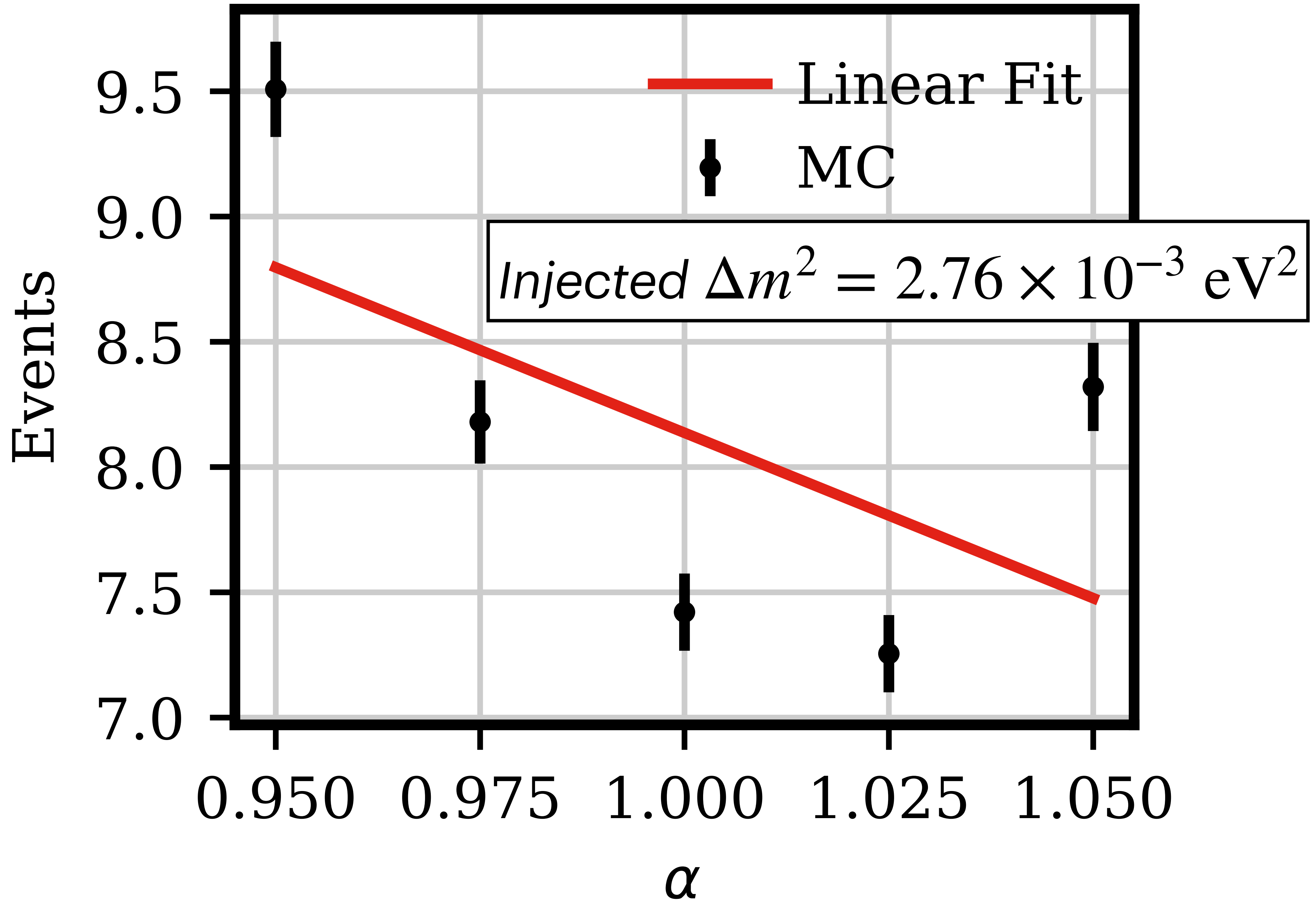
Toy example:
Parameter α = detector efficiency

- **Simplest method:** Bin-wise gradients in reconstructed event properties
- **Problem:** Depend on physics parameters θ

BIN-WISE WEIGHTING METHODS

Gradients' dependence on Physics Parameters





Need to re-fit the detector systematics every time we change the injected physics parameters θ.

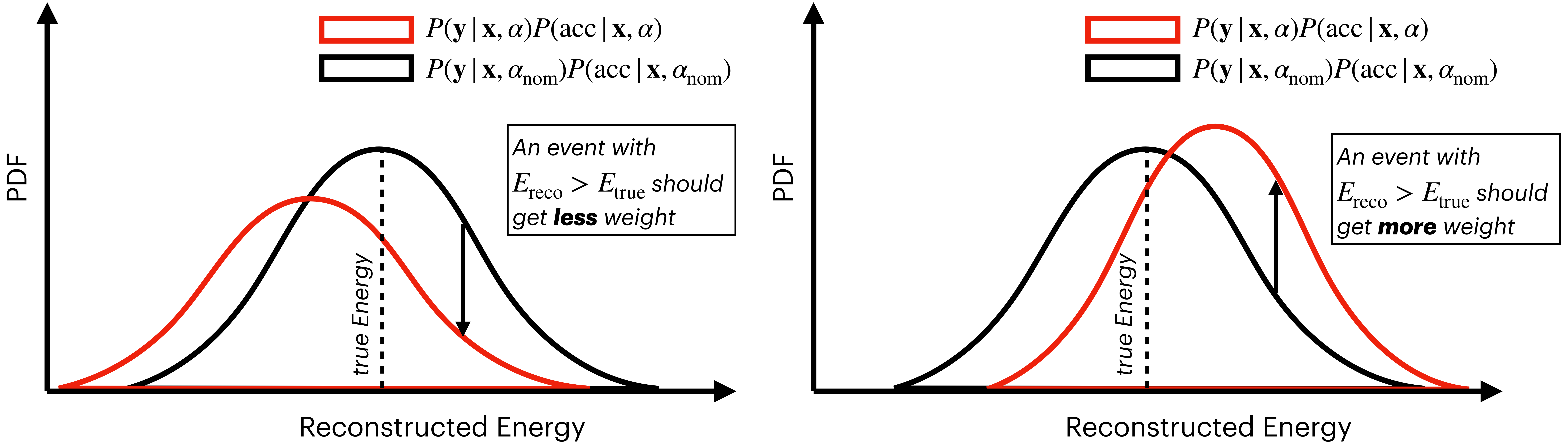
➡ **We need a method to decouple detector effects from physics!**

Luckily, you're in the right talk!

Decoupling Detector Effects

Event weights should model the relationship between true and reco variables

Example: α = detector efficiency, $\alpha_{\text{nom}} = 1$



$\alpha < 1$: *Reconstructed energy tends to be smaller than true energy*

$\alpha > 1$: *Reconstructed energy tends to be larger than true energy*

DECOUPLING DETECTOR EFFECTS

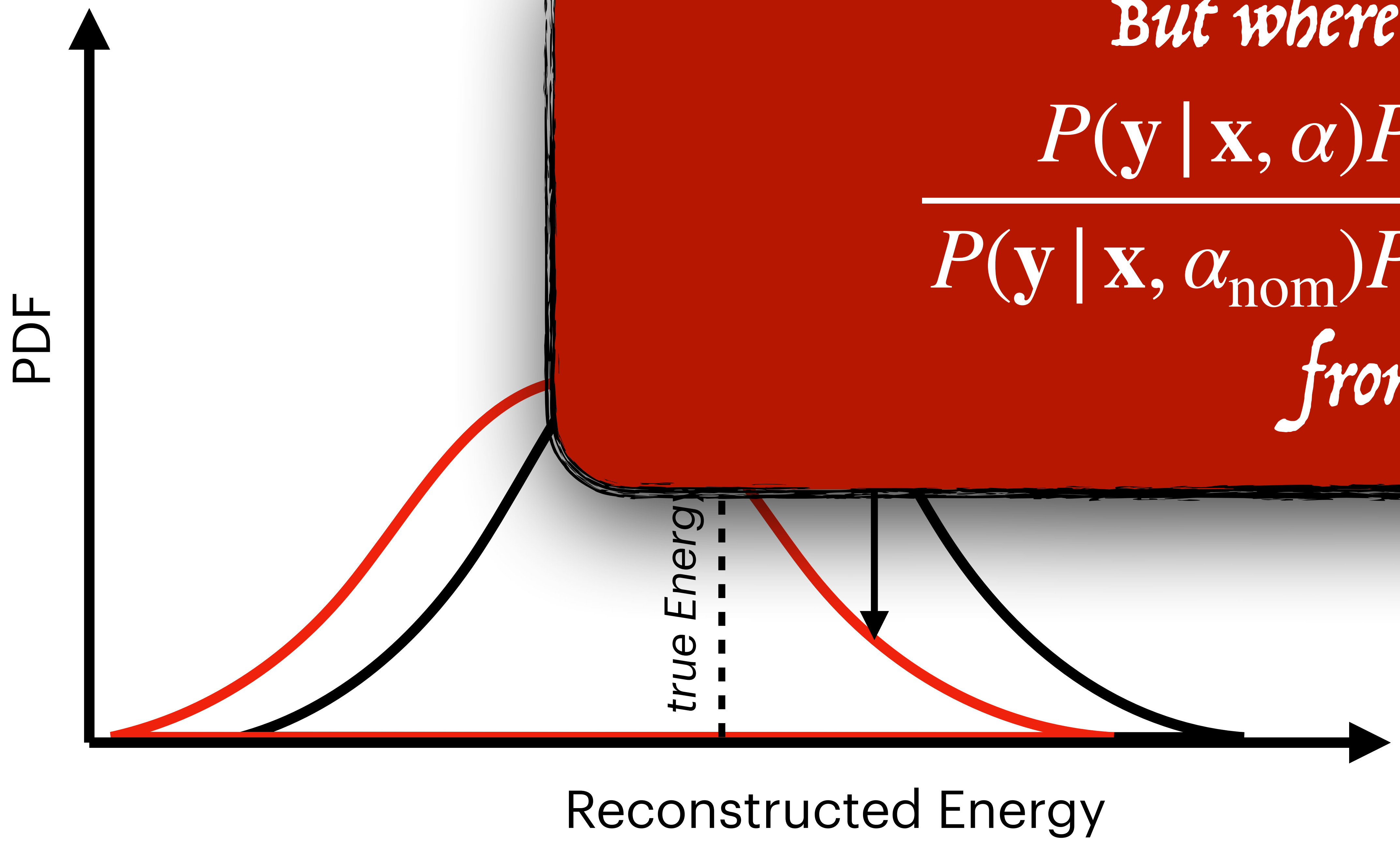
Event weights should model the relationship between true and reco variables

Example: α = detector efficiency, $\alpha_{\text{nom}} = 1$

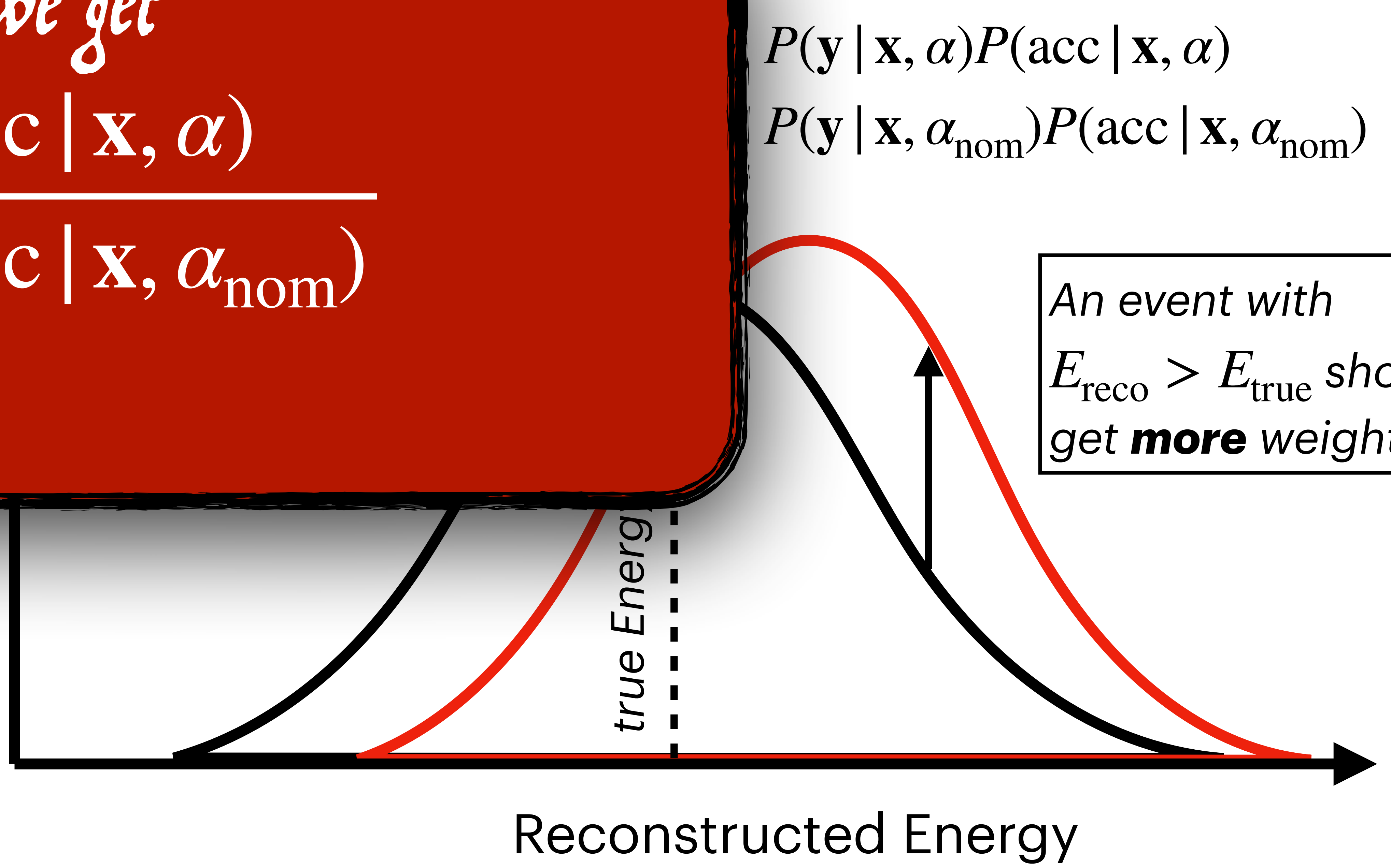
But where do we get

$$\frac{P(\mathbf{y}\,|\,\mathbf{x},\alpha)P(\text{acc}\,|\,\mathbf{x},\alpha)}{P(\mathbf{y}\,|\,\mathbf{x},\alpha_{\text{nom}})P(\text{acc}\,|\,\mathbf{x},\alpha_{\text{nom}})}$$

from?



$\alpha < 1$: *Reconstructed energy tends to be smaller than true energy*



An event with $E_{\text{reco}} > E_{\text{true}}$ should get ***more*** *weight*

$\alpha > 1$: *Reconstructed energy tends to be larger than true energy*

THE LIKELIHOOD-FREE INFERENCE TRICK

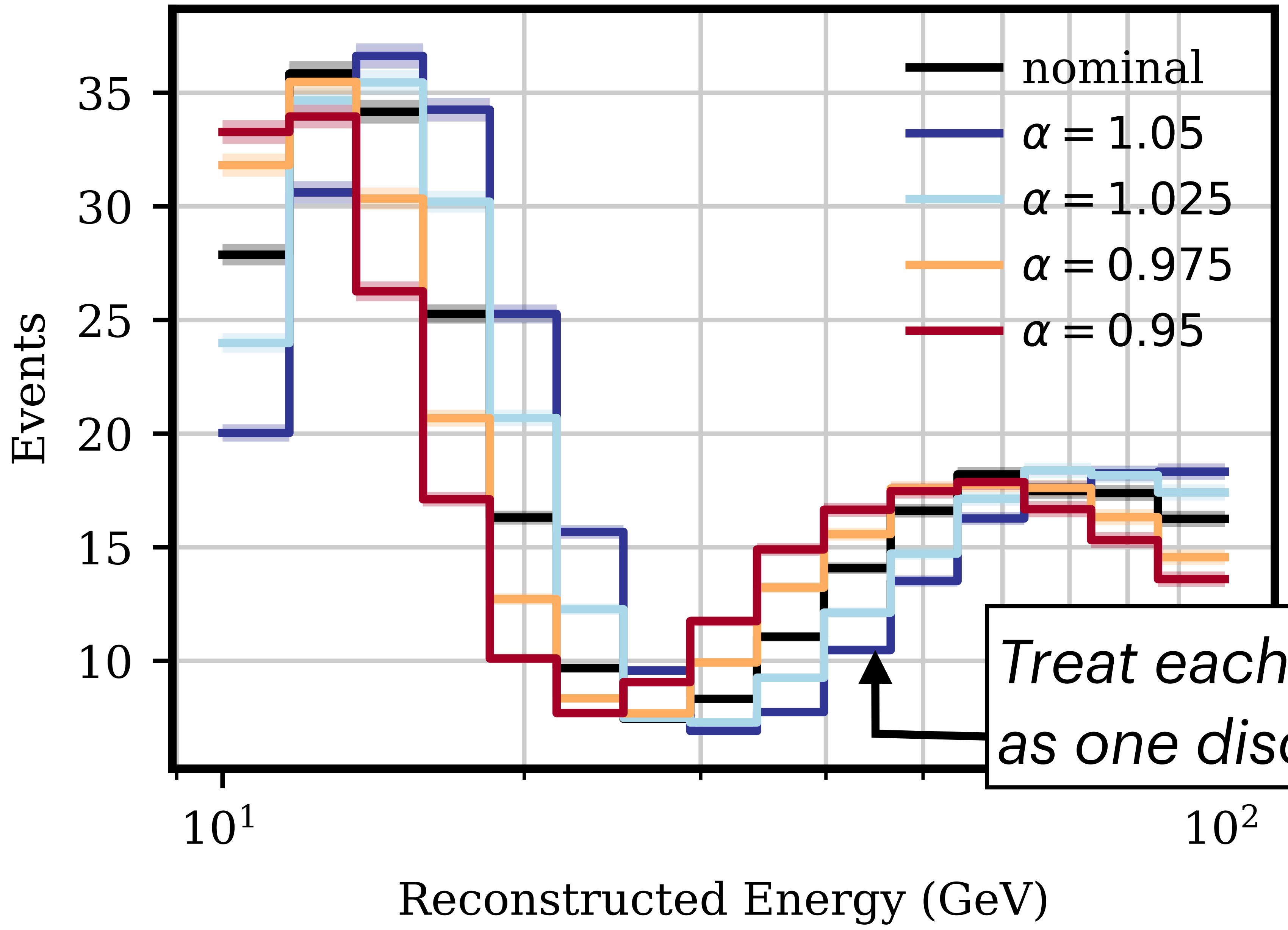
Weighting from Nominal to Any Off-Nominal MC Set

Applying Bayes' Theorem:

index k = index of MC set

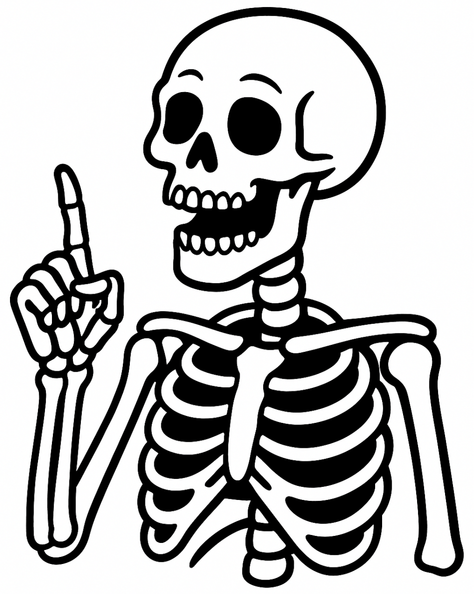
$$\frac{P(\mathbf{y}\,|\,\mathbf{x},\alpha_k)P(\mathrm{acc}\,|\,\mathbf{x},\alpha_k)}{P(\mathbf{y}\,|\,\mathbf{x},\alpha_{\mathrm{nom}})P(\mathrm{acc}\,|\,\mathbf{x},\alpha_{\mathrm{nom}})} = \frac{P(\alpha_k\,|\,\mathbf{x},\mathbf{y})}{P(\alpha_{\mathrm{nom}}\,|\,\mathbf{x},\mathbf{y})}$$

Train a classifier to estimate posterior that an event with given $\mathbf{x}, \mathbf{y}$ belongs to set k



Treat each MC set as one discrete class

We convert the difficult problem of learning conditional probability distributions to the easy problem of training a classifier!



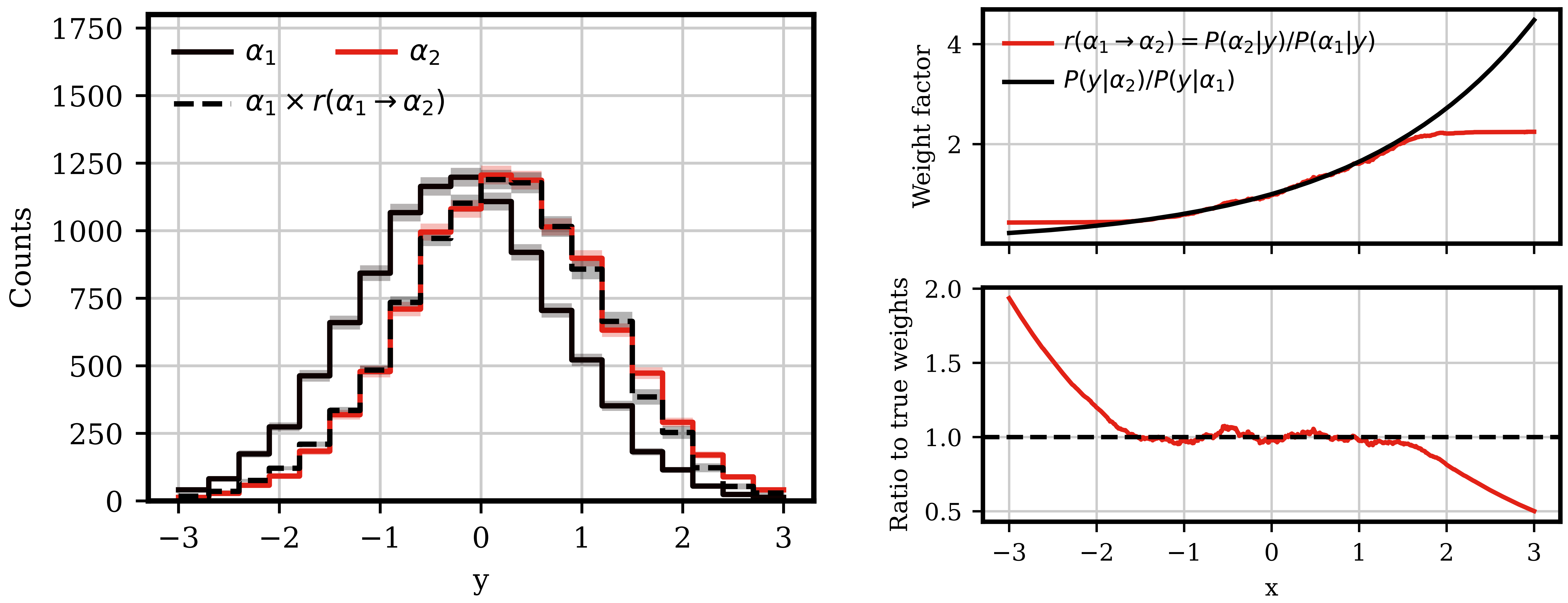
KNN Classifier Example

Simple and Robust Posterior Estimate

KNN Classifier Equation:

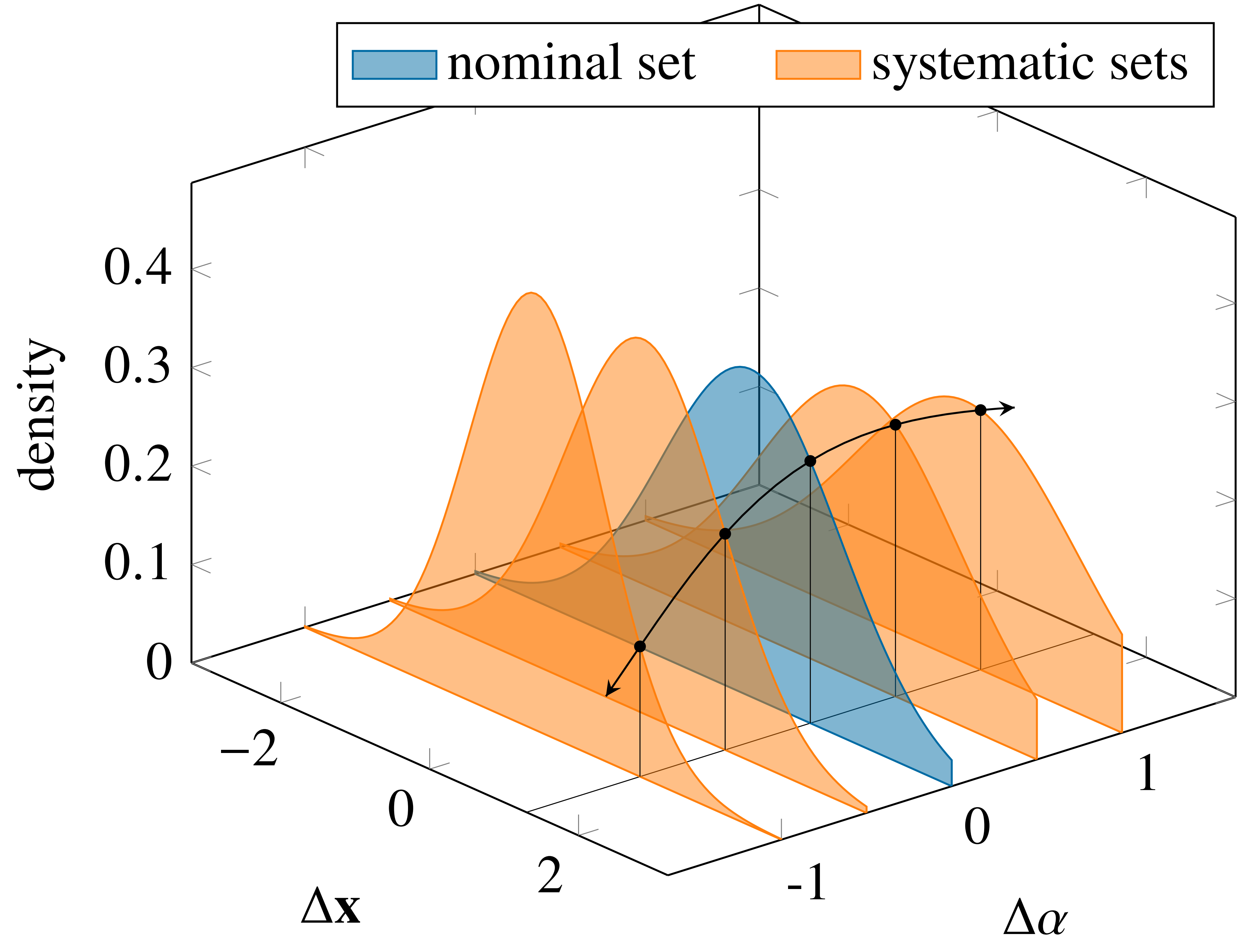
Sum over indices in neighborhood around $(\mathbf{x}, \mathbf{y})$ belonging to set k

$$P(\alpha = \alpha_k \,|\, \mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{j \in \mathcal{N}_k(\mathbf{x},\mathbf{y})} 1$$



Making Event-Wise Gradients

Interpolating between Discrete MC Sets



- **Probability estimate** using softmax to normalize

$$\hat{P}(\alpha_k \,|\, \mathbf{x}_j, \mathbf{y}_j) = \text{softmax}\left(\mathbf{g}_j A\right) = \frac{\exp(\sum_n g_{jn} A_{nk})}{\sum_{k'} \exp(\sum_n g_{jn} A_{nk'})}$$

g_{jn} = gradient w.r.t. α_n for event j

$$A_{nk} = \alpha_{n,k} - \alpha_{n,\,\text{nom}}$$

- **Loss function** to fit gradients g_{jn} is cross-entropy:

$$H_j = -\sum_k \log(\hat{P}(\alpha_k \,|\, \mathbf{x}_j, \mathbf{y}_j)) P_{\text{KNN}}(\alpha_k \,|\, \mathbf{x}_j, \mathbf{y}_j)$$

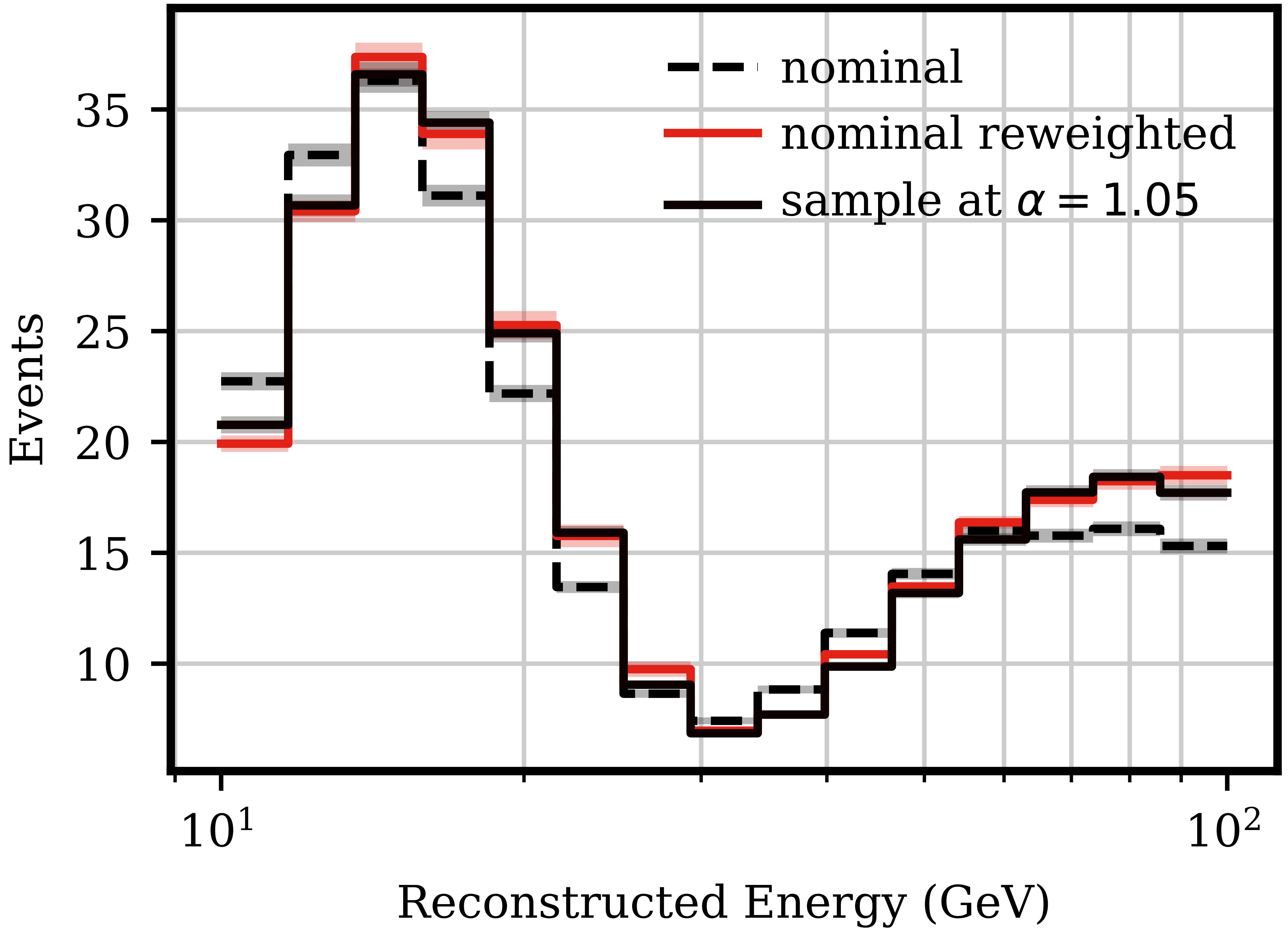
- **Interpolated weight** for every MC event during evaluation

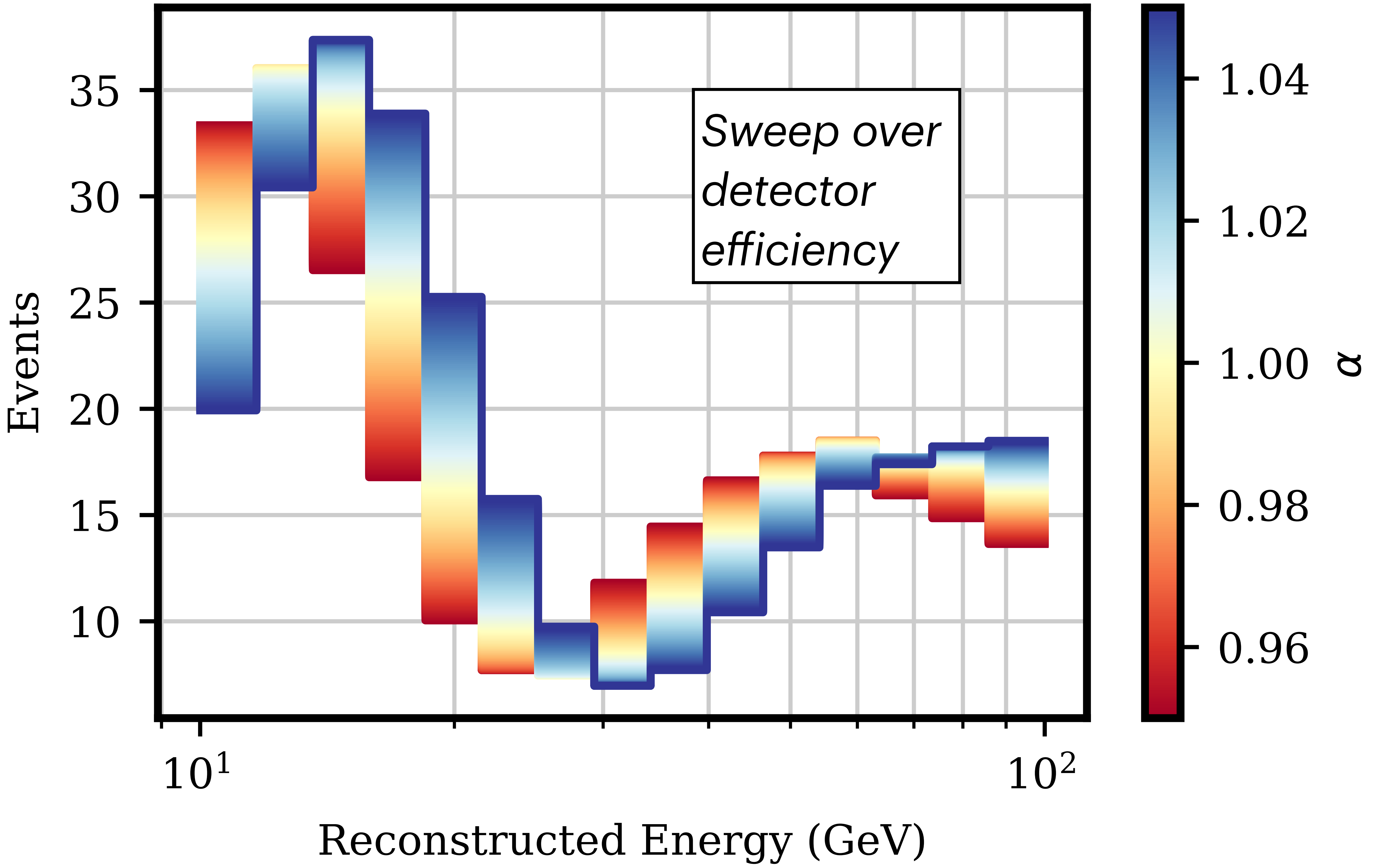
$$\hat{r}_j(\alpha) = \frac{\hat{P}(\alpha \,|\, \mathbf{x}_j, \mathbf{y}_j)}{\hat{P}(\alpha_{\text{nom}} \,|\, \mathbf{x}_j, \mathbf{y}_j)} = \exp\left(\sum_n g_{jn}(\alpha_n - \alpha_{n,\,\text{nom}})\right)$$

For each event, we train the last layer of a neural network with one weight per systematic parameter.

TOY MC EXAMPLE

Reweighting between and in between MC sets

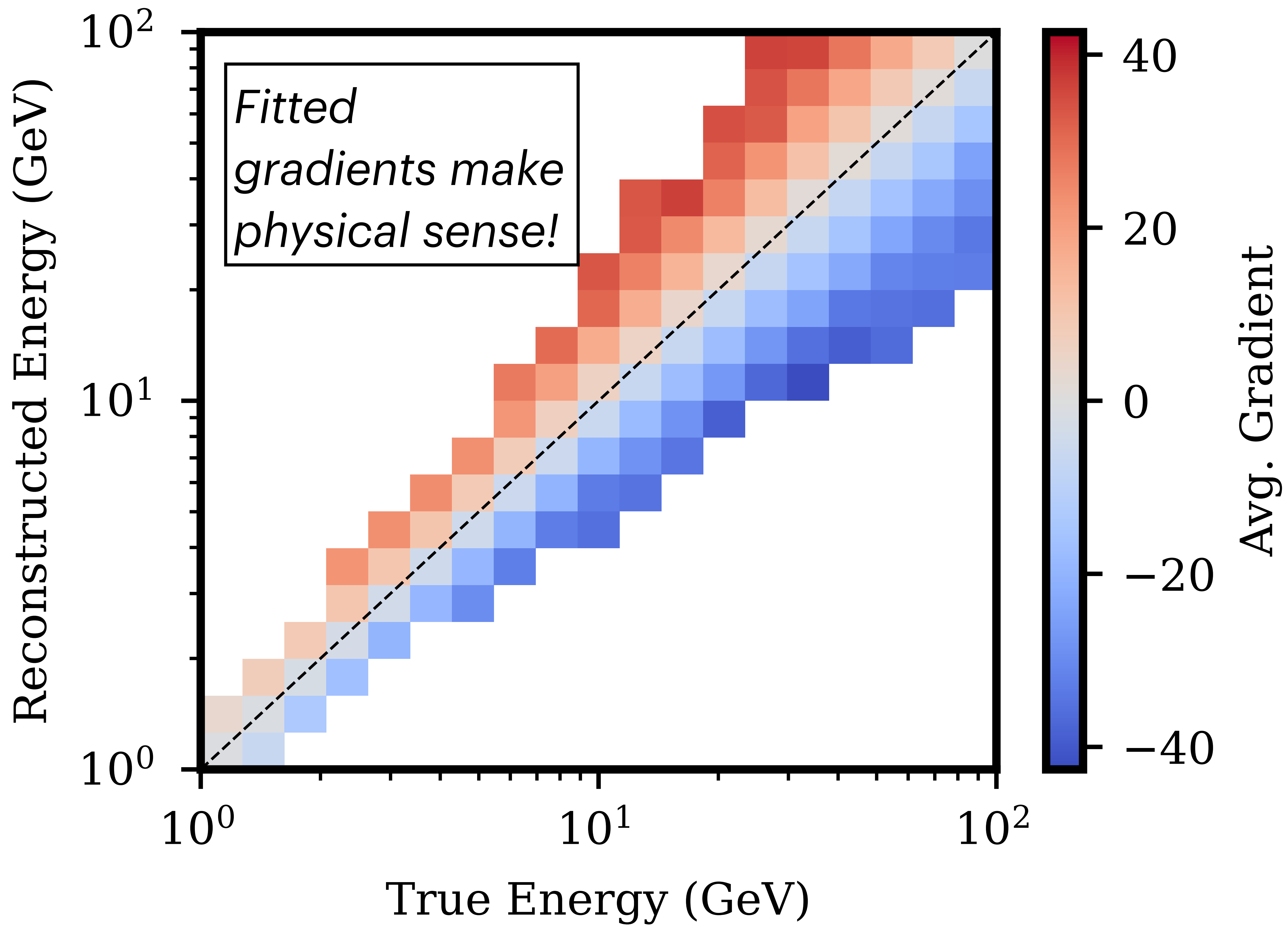


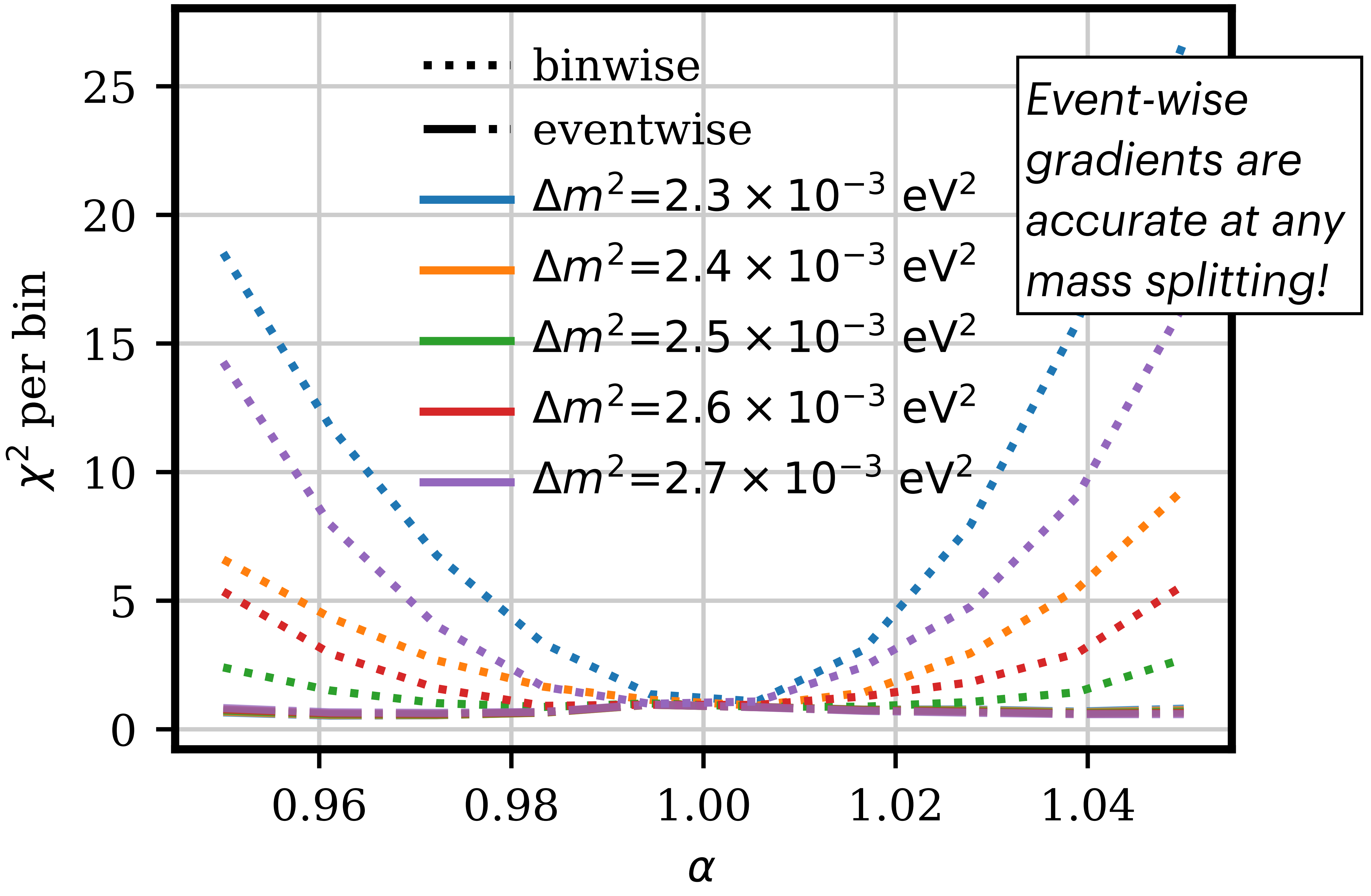


Note: The classifier was trained on the *unweighted* events!

Performance on Toy MC

Gradients make Sense and Produce Accurate Predictions





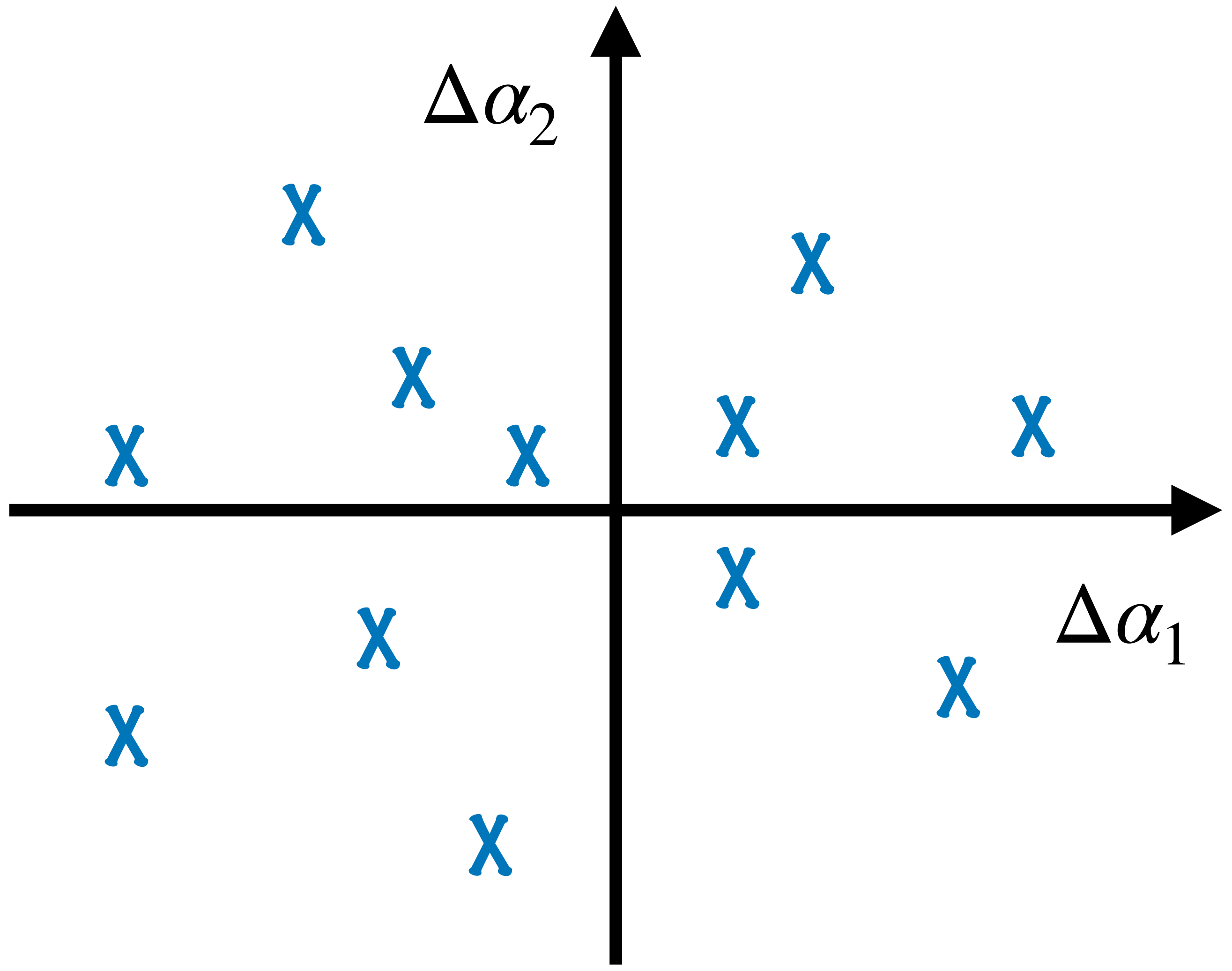
SUMMARY OF THE PROCEDURE

How you can apply this in your analysis

Summary of the procedure

How you can apply this in your analysis

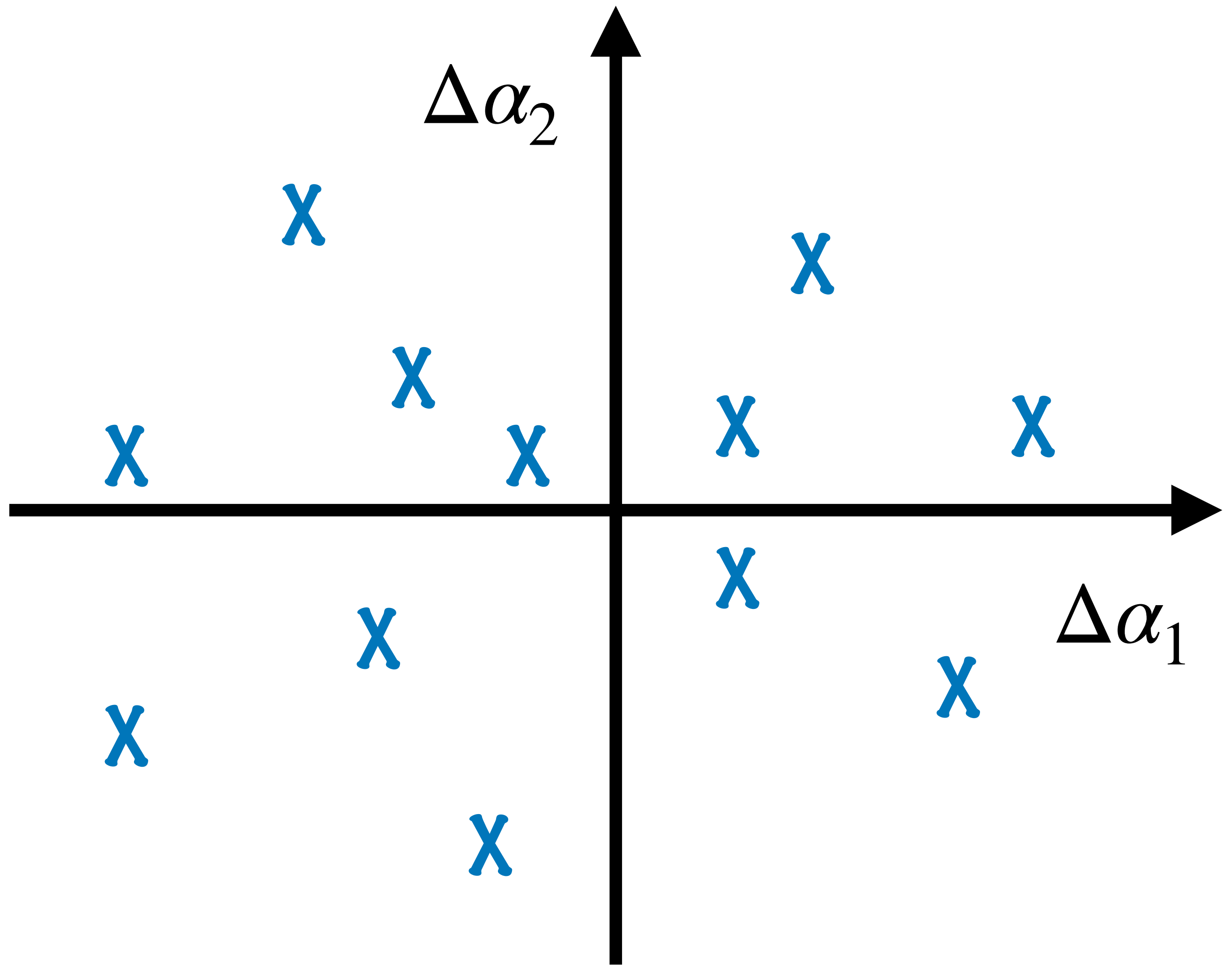
1. Generate MC sets at various realizations of the detector
    - Systematic sets may be arranged arbitrarily!



Summary of the Procedure

How you can apply this in your analysis

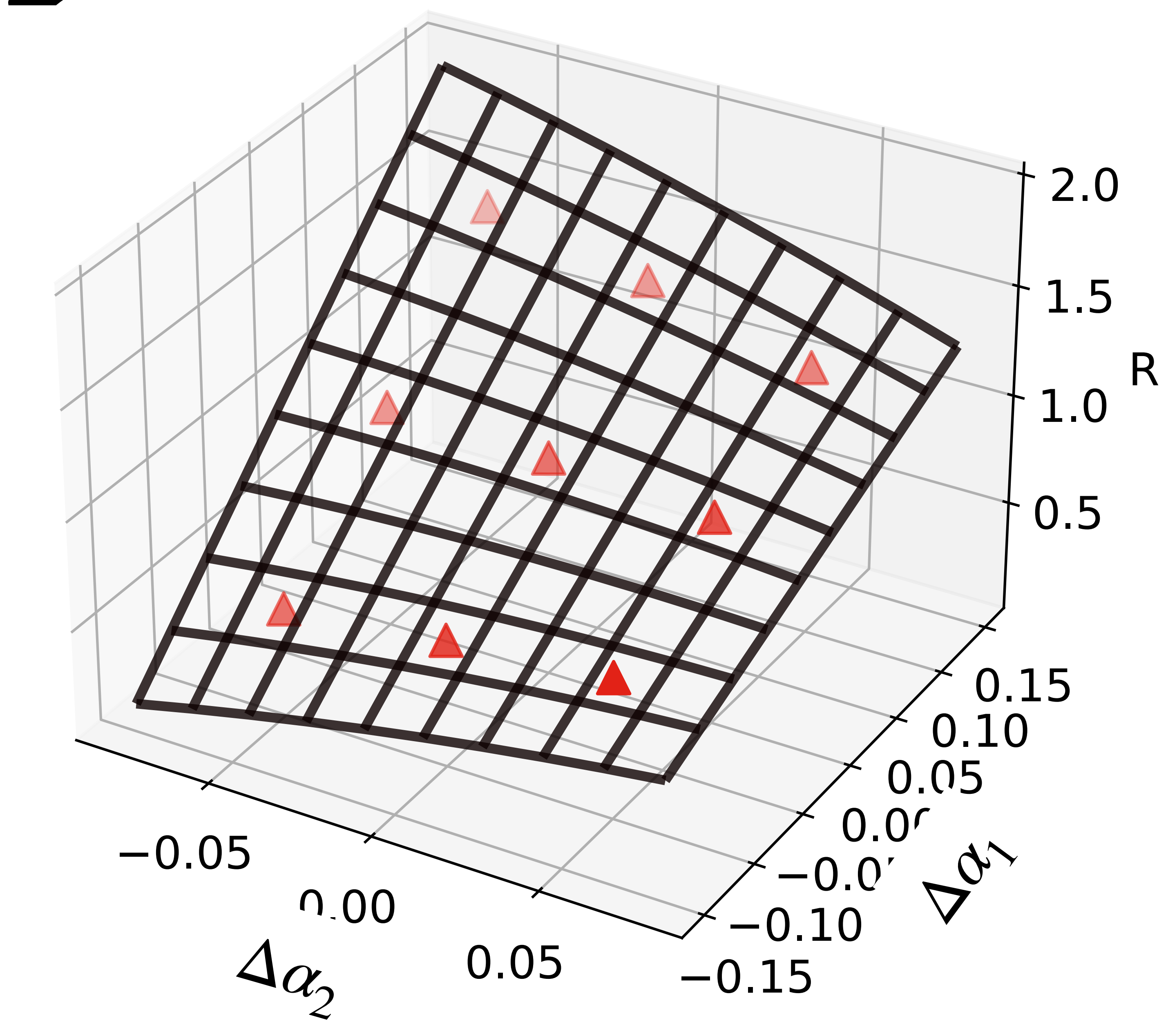
1. Generate MC sets at various realizations of the detector
   - Systematic sets may be arranged arbitrarily!
2. Fit the classifier
   - Any classifier giving *calibrated* posteriors may be used



Summary of the Procedure

How you can apply this in your analysis

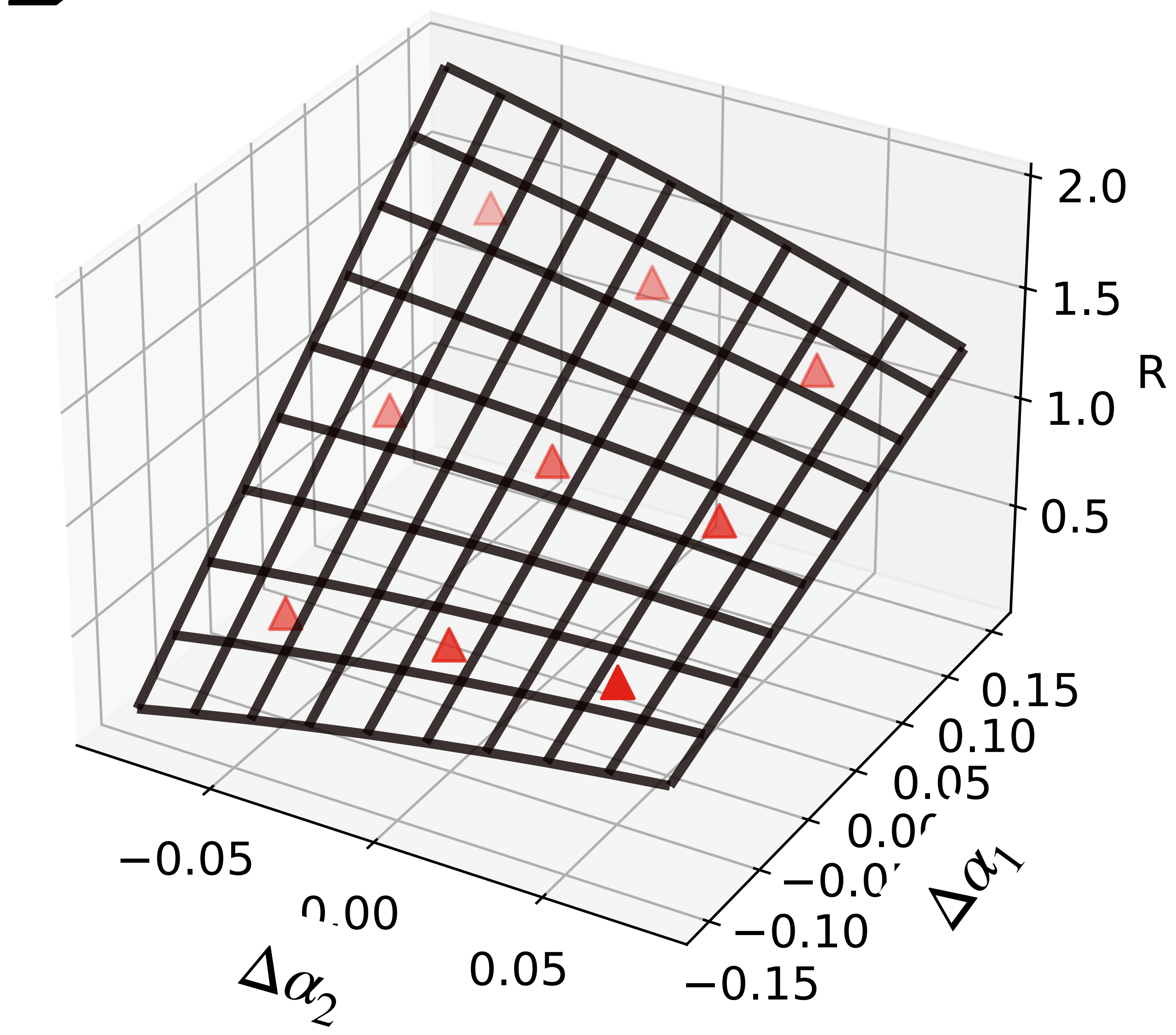
1. Generate MC sets at various realizations of the detector
   - Systematic sets may be arranged arbitrarily!
2. Fit the classifier
   - Any classifier giving *calibrated* posteriors may be used
3. Fit gradients for all nominal MC events
   - Polynomial features of parameters may be used



SUMMARY OF THE PROCEDURE

How you can apply this in your analysis

1. Generate MC sets at various realizations of the detector
   - Systematic sets may be arranged arbitrarily!
2. Fit the classifier
   - Any classifier giving *calibrated* posteriors may be used
3. Fit gradients for all nominal MC events
   - Polynomial features of parameters may be used

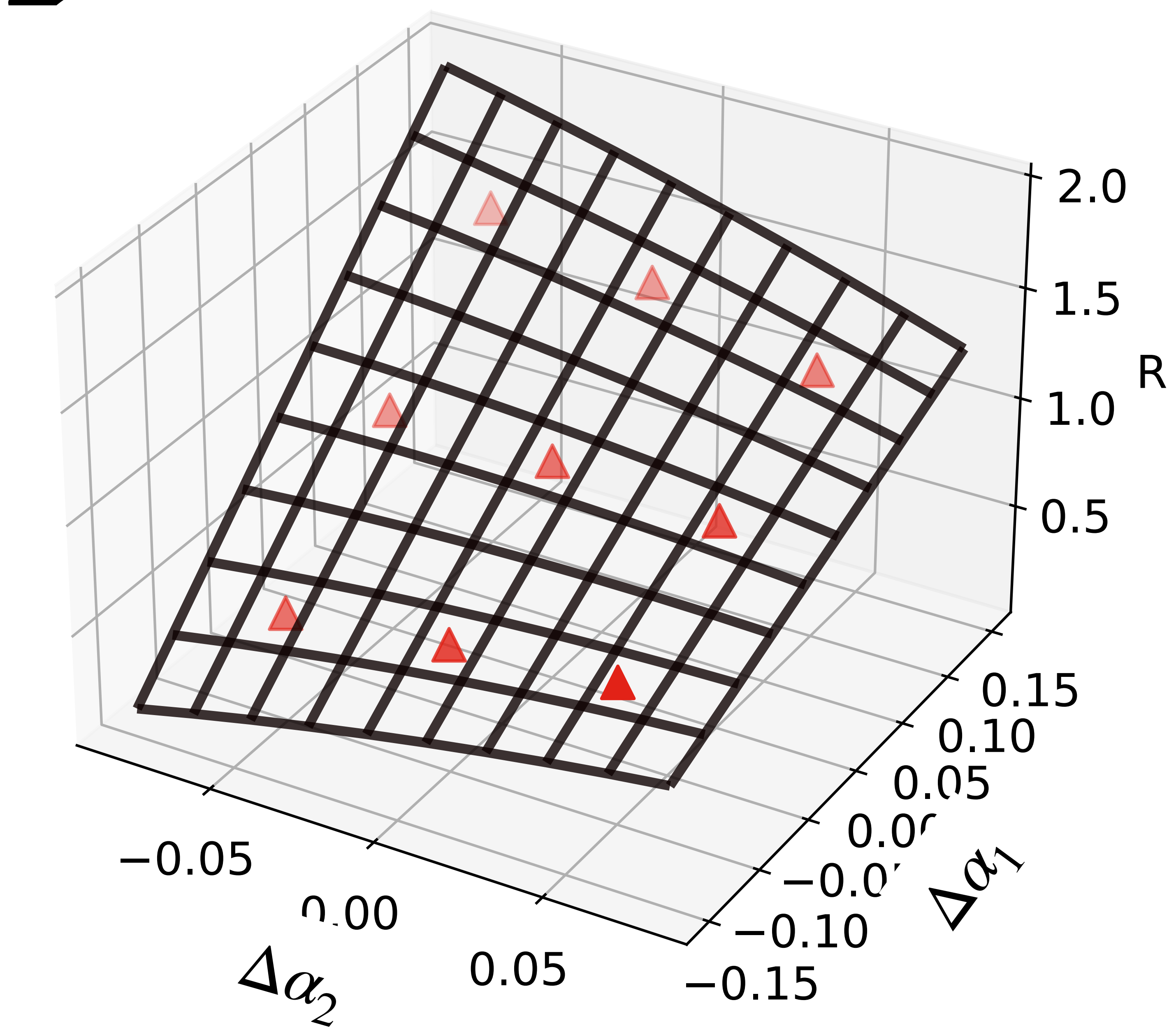


4. Weight your MC by $w_j = \exp\left(\sum_n g_{jn}(\alpha_n - \alpha_{n,\,\mathrm{nom}})\right)$ to get expectation values for any detector realization!

Summary of the Procedure

How you can apply this in your analysis

1. Generate MC sets at various realizations of the detector
   - Systematic sets may be arranged arbitrarily!
2. Fit the classifier
   - Any classifier giving *calibrated* posteriors may be used
3. Fit gradients for all nominal MC events
   - Polynomial features of parameters may be used



4. Weight your MC by $w_j = \exp\left(\sum_n g_{jn}(\alpha_n - \alpha_{n,\,\mathrm{nom}})\right)$ to get expectation values for any detector realization!

Analyzers need only the last step!

Benefits of the Method

Final Takeaways

Benefits of the Method

Final Takeaways

- ✓ Learn event-wise gradients to re-weight MC events to any detector configuration *decoupled* from physics

BENEFITS OF THE METHOD

Final Takeaways

- ✓ Learn event-wise gradients to re-weight MC events to any detector configuration *decoupled* from physics
- ✓ Change your binning*! Change your Physics! Gradients stay valid!

**Re-binning or selection changes may only use variables that are included in the classifier training*

BENEFITS OF THE METHOD

Final Takeaways

- ✓ Learn event-wise gradients to re-weight MC events to any detector configuration *decoupled* from physics
- ✓ Change your binning*! Change your Physics! Gradients stay valid!
- ✓ Allow anyone to use detector effects by adding gradients to MC data-release

**Re-binning or selection changes may only use variables that are included in the classifier training*

Benefits of the Method

Final Takeaways

- ✓ Learn event-wise gradients to re-weight MC events to any detector configuration *decoupled* from physics
- ✓ Change your binning*! Change your Physics! Gradients stay valid!
- ✓ Allow anyone to use detector effects by adding gradients to MC data-release
- ✓ No assumption of linearity of detector effects or Gaussian error distribution

**Re-binning or selection changes may only use variables that are included in the classifier training*

BENEFITS OF THE METHOD

Final Takeaways

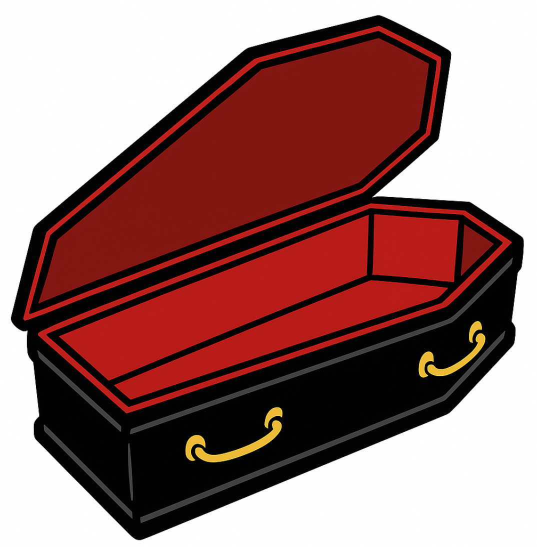
- ✓ Learn event-wise gradients to re-weight MC events to any detector configuration *decoupled* from physics
- ✓ Change your binning*! Change your Physics! Gradients stay valid!
- ✓ Allow anyone to use detector effects by adding gradients to MC data-release
- ✓ No assumption of linearity of detector effects or Gaussian error distribution
- ✓ Can model any number of detector effects

**Re-binning or selection changes may only use variables that are included in the classifier training*

BENEFITS OF THE METHOD

Final Takeaways

- ✓ Learn event-wise gradients to re-weight MC events to any detector configuration *decoupled* from physics
- ✓ Change your binning*! Change your Physics! Gradients stay valid!
- ✓ Allow anyone to use detector effects by adding gradients to MC data-release
- ✓ No assumption of linearity of detector effects or Gaussian error distribution
- ✓ Can model any number of detector effects



Happy Halloween!

**Re-binning or selection changes may only use variables that are included in the classifier training*

Backup

Power of the Method

How general is this, really?

How many systematic variations can be used?

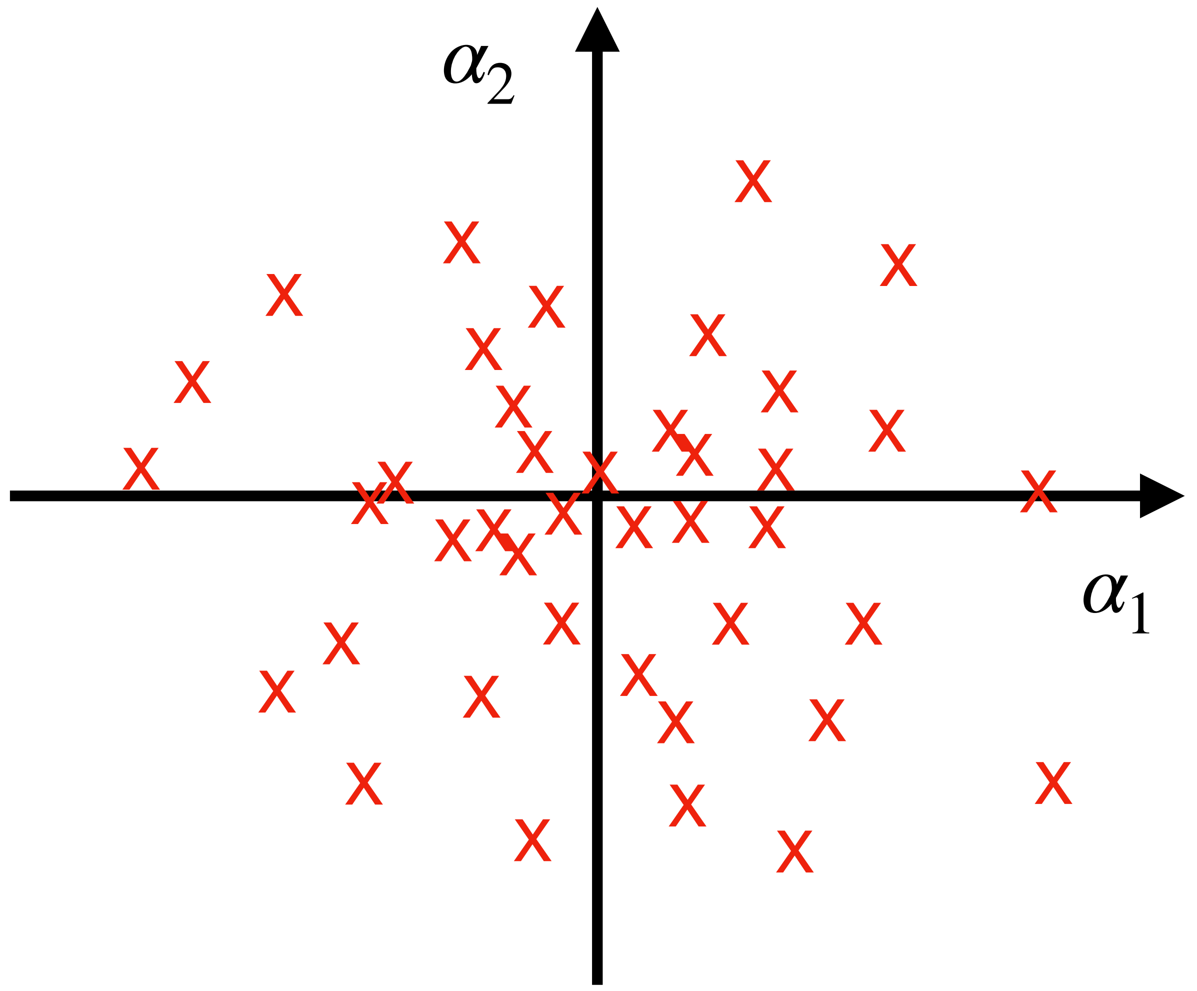
➡ Any number of MC sets, at any location!

➡ Even continuous variation of detector parameters is possible!*

How many true/reco parameters should I include per event?

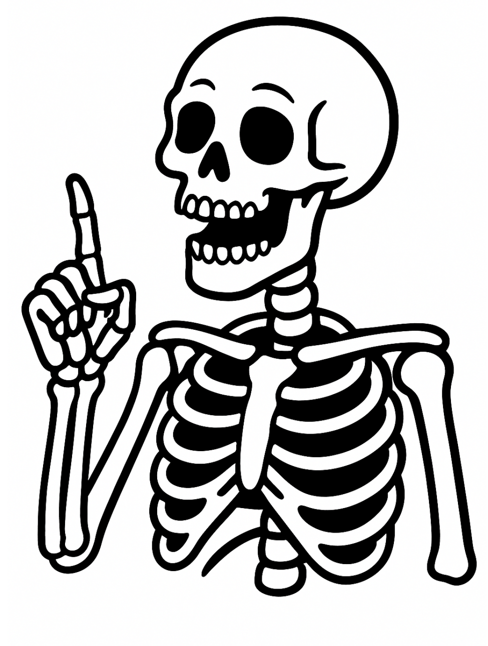
➡ Lower limit: at least those variables used in the analysis binning and those that are used by the physics model *must* be included

➡ Upper limit: as many as your classifier can handle!



$$P(\alpha_k \,|\, \mathbf{x}, \mathbf{y})$$

The inputs to the classifier are the concatenated true and reco variables



*Explaining how this works goes beyond the scope of this talk, talk to me if you're interested

Goal of this Work

Decoupling Detector Response Weight from Physics Parameters

Basic intuition: Detector response should not depend on the physics model!

- ➡ Detector reacts to *final state of each particle*, doesn't know about flux, oscillations, etc.
- ➡ Detector properties determine *relationship between true and reconstructed variables*
- ➡ If we knew $P(\mathbf{y} \mid \mathbf{x}, \alpha)P(\mathrm{acc} \mid \mathbf{x}, \alpha)$, we should be able to get the correct expectation value at any setting of α *independently* from θ

$$\hat{\mu}_i(\theta, \alpha) = \sum_j I(\mathbf{y}_j \in \mathrm{bin}\ i) \boxed{\frac{P(\mathbf{y}_j \mid \mathbf{x}_j, \alpha)P(\mathrm{acc} \mid \mathbf{x}_j, \alpha)}{P(\mathbf{y}_j \mid \mathbf{x}_j, \alpha_{\mathrm{nom}})P(\mathrm{acc} \mid \mathbf{x}_j, \alpha_{\mathrm{nom}})}} w(\mathbf{x}, \theta)$$

Event weight independent of θ

MC Event Weighting

How we get an expectation value in each bin

Full expression for expectation in each bin i:

$$\mu_i(\theta) = \int_{\mathbf{y}\in \text{bin } i} \mathrm{d}\mathbf{y} \int \mathrm{d}\mathbf{x} P(\mathbf{y} \,|\, \mathbf{x}, \alpha) P(\text{acc} \,|\, \mathbf{x}, \alpha) \frac{\Phi(\mathbf{x} \,|\, \theta)}{\Phi_{\text{sim}}(\mathbf{x})}$$

- **Flux, cross-sections, oscillations**:
  - Estimate bin count by weighting events:

$$\hat{\mu}_i(\theta) = \sum_j I(\mathbf{y}_j \in \text{bin } i) \frac{\Phi(\mathbf{x}_j \,|\, \theta)}{\Phi_{\text{sim}}(\mathbf{x}_j)}$$

- **Uncertainties of detector properties**:
  - How can we get $P(\text{acc} \,|\, \mathbf{x}, \alpha) P(\mathbf{y} \,|\, \mathbf{x}, \alpha)$?