Trustworthy Neural Simulation-Based Inference at Colliders

Parameter Estimation with Noisy Networks and Biased Simulations

Sean Benevedes
LITP Spring Symposium 2026
May 19th, 2026

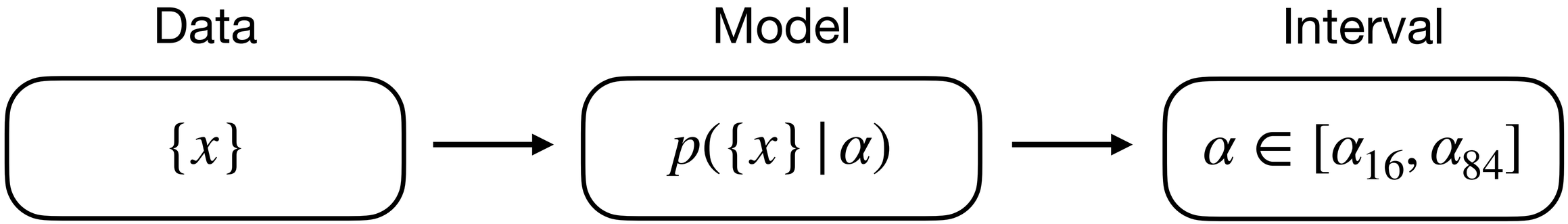
(Based on **SB**, Thaler 2506.00113,
Alvarez, **SB**, Szewc, Thaler 2604.02219)

Outline

1. Why simulation-based inference (SBI)?
2. Neural uncertainties for SBI: $w_i f_i$ ensembles (2506.00113)
3. SBI with biased simulations: template-adapted mixture models (2604.02219)

Simulation-based inference (SBI) for colliders

Ideal inference pipeline



~~Ideal inference pipeline~~

Data → Model → Interval

$\{x\}$ →  → $\alpha \in [\alpha_{16}, \alpha_{84}]$

~~Ideal inference pipeline~~

Data → Model → Interval

$\{x\}$ →  → $\alpha \in [\alpha_{16}, \alpha_{84}]$

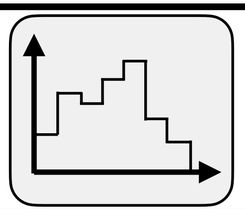
We need simulation-based inference!

(Classical) SBI

Data

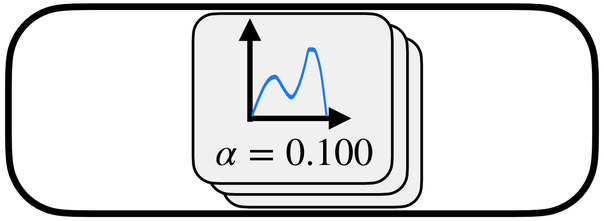
$\{x\}$

(Binned) Data



Model

(Partial) Model



Interval

$\alpha \in [\alpha_{16}, \alpha_{84}]$

(Classical) SBI

This procedure yields a tractable likelihood, but it has two serious deficiencies

1. Binning is lossy: we throw away all information about what is happening within the bins!

2. Curse of dimensionality: If you want to compare to multidifferential predictions, you need exponentially large datasets to fill your bins!

Density ratio estimation lets us do better.

(Neural) SBI

Data

$\{x\}$

Model

Interval

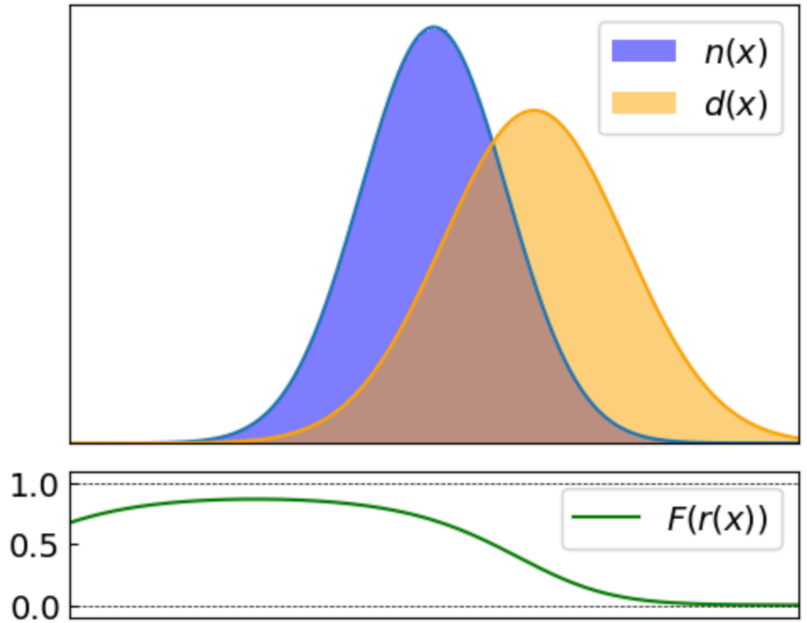
$\alpha \in [\alpha_{16}, \alpha_{84}]$

Estimated likelihood ratio

$$\frac{p(\{x\} \mid \alpha)}{p_{\text{ref}}(\{x\})}$$

How: Machine learning!

- Neyman-Pearson Lemma: The best classifier between n and d is given by any monotonic function of r
- Idea ("likelihood ratio trick"): Train a classifier on samples from n and d, solve for r (see e.g. Cranmer et al., 1506.02169)



But… neural networks only estimate the density ratio, not learn it exactly! Our statistics will be wrong unless we account for this.

Need uncertainties!

See Hermans et al. 2110.06581 for more on problems caused by neglecting these uncertainties

Also, even if we were to learn the likelihood ratio perfectly, our SBI is only as good as our simulators.

May need to perform inference with biased simulators!

As we will see, both of these problems can be attacked by augmenting our neural networks with classical, parametric statistics.

SBI with uncertainties: $w_i f_i$ ensembles

Intuition: These uncertainties manifest as differences in the trained network from training to training. Let’s try to use this!

$w_i f_i$ ensembles

$$\log r(x \mid \{w_i\}) = \sum_i w_i f_i(x)$$

where w_i are scalar weights and $f_i(x)$ are frozen networks

Pros	Cons
• Fully frequentist, no prior dependence • Intuitive: just like nuisance parameters! • Computationally efficient; no bootstrapping	• Only formally valid if model is well-specified

$w_i f_i$ ensembles

$$\log r(x \mid \{w_i\}) = \sum_i w_i f_i(x)$$

where w_i are scalar weights and $f_i(x)$ are frozen networks

Pros	Cons
• Fully frequentist, no prior dependence • Intuitive: just like nuisance parameters! • Computationally efficient; no bootstrapping	• Only formally valid if model is well-specified Potentially a big problem!

Model specification

- Definition: A model $p(x \mid \{w\})$ of some data x given a set of parameters $\{w\}$ is *well-specified* if there exists some $\{w^*\}$ such that $p(x \mid \{w^*\}) = p_{\text{data}}(x)$
- This is a very strong assumption! $w_i f_i$ ensembles, and all the rest of statistical inference, would be hopeless if our models needed to be exactly well-specified to work in practice (all models are wrong!)
- Empirical question: Are $w_i f_i$ ensembles well-specified enough? (Yes).

NB: Standard practice is to train *one* network and assume that the resulting model is well-specified, or to train an ensemble and simply average the outputs.

$w_i f_i$ ensembles: the recipe

Ingredients

- M randomly initialized networks
- A dataset D_{train} with N_{train} samples each from n and d
- Another* dataset D_{fit} with N_{fit} samples each from n and d
- (Optional) Dataset(s) for downstream applications

1. Train the f_i on D_{train}
2. Fit $\hat{w}_i$ with D_{fit} using the MLC loss (D'Agnolo et al. 1806.02350)
3. Compute $C_{ij} \equiv \text{Cov}(\hat{w}_i, \hat{w}_j)$ with analytic formulae
4. Propagate uncertainties for downstream tasks

1. Training the f_i

Objective: Train f_i such that there is some set of w_i for which $w_i f_i$ is a good model of $\log r$... that leaves a lot of freedom!

1. Bootstrap

Training data

Ensemble

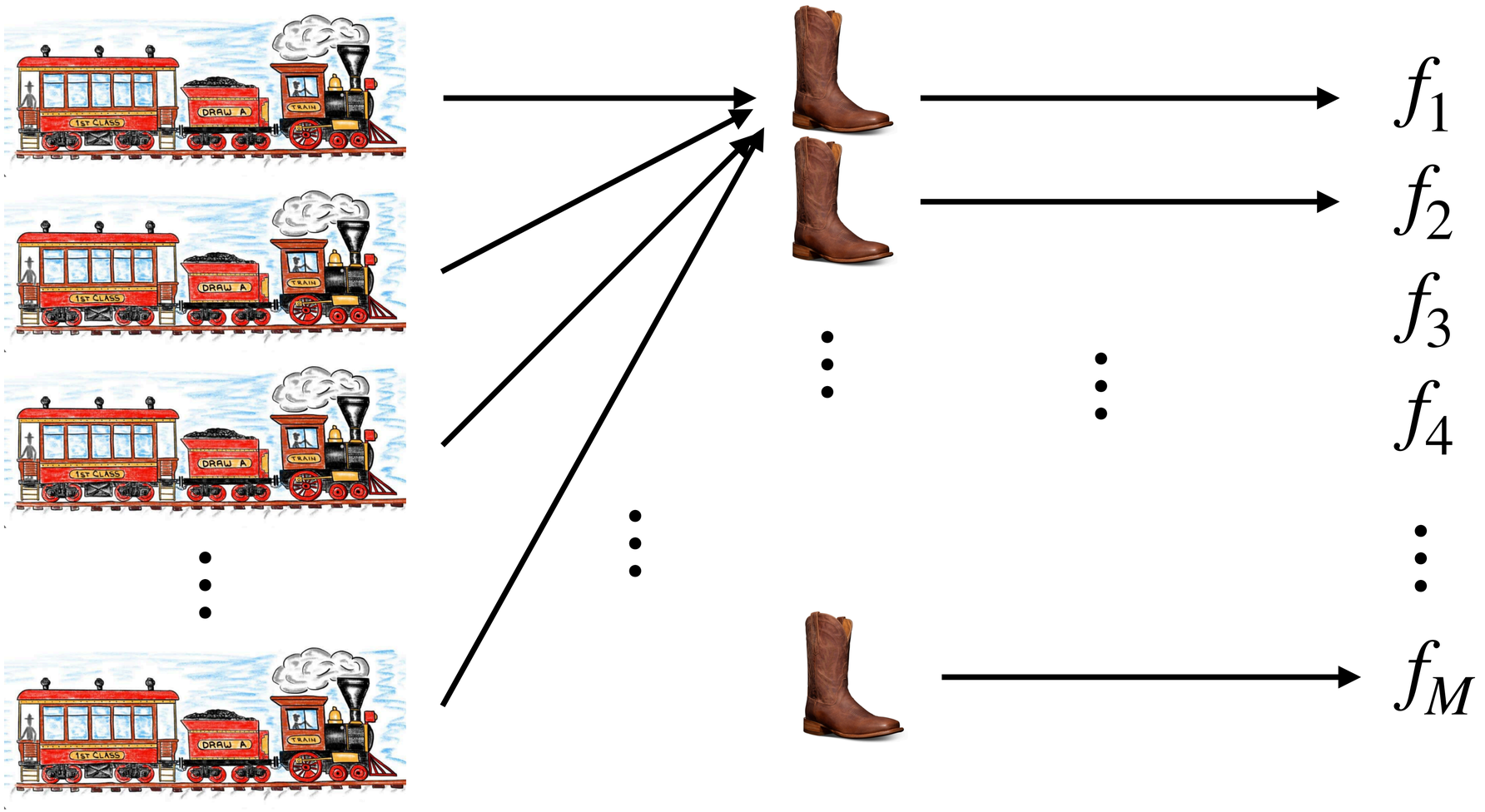
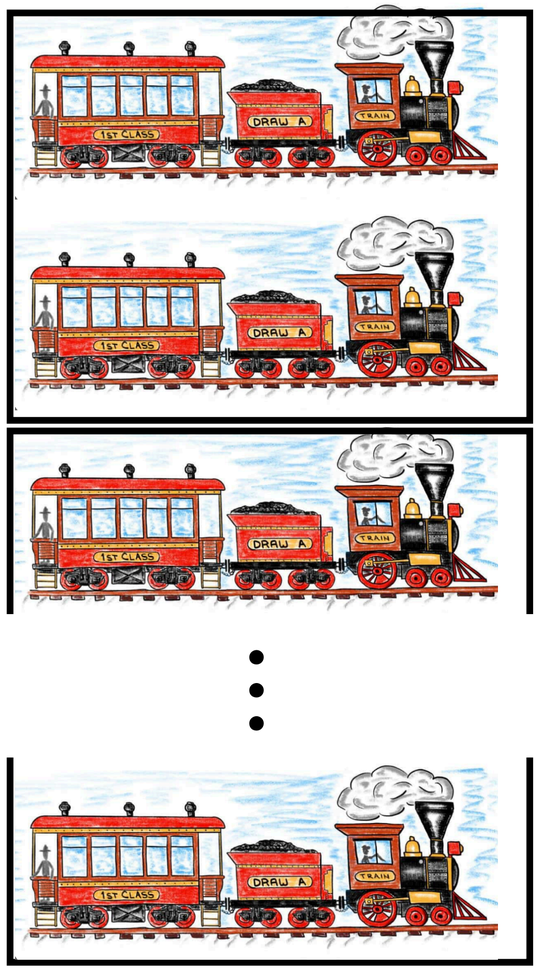


Image credit: https://texasbootcompany.com/products/tecovas-mens-the-doc-scotch-goat-boots
Image credit: https://letsdrawthat.com/how-to-draw-a-train/

1. Partition

Training data

Ensemble



f_1

f_2

f_3

f_4

$\vdots$

f_M

Image credit: https://letsdrawthat.com/how-to-draw-a-train/

2/3. Fitting the w_i

Now, find the $\hat{w}_i$ minimizing the symmetrized MLC loss:

$$\mathscr{L}(w) = -\left(\sum_{x_\alpha \sim n}^{N_{fit}} [w_i f_i(x_\alpha) - (e^{-w_i f_i(x_\alpha)} - 1)] + \sum_{x_\alpha \sim d}^{N_{fit}} [-w_i f_i(x_\alpha) - (e^{w_i f_i(x_\alpha)} - 1)] \right)$$

Estimators of this form are called *M-estimators* (Huber 1964) in the statistics literature, and their covariances are known!

$$\boxed{C_{ij} = V^{-1}_{ik} U_{kl} V^{-1}_{lj}} \left(\text{where } U_{ij} \equiv \text{Cov}\left(\frac{\partial \mathscr{L}}{\partial w_i}, \frac{\partial \mathscr{L}}{\partial w_j} \right), V_{ij} \equiv \frac{\partial^2 \mathscr{L}}{\partial w_i \partial w_j} \right)$$

4. Uncertainty propagation

- As an example of a downstream application, consider parameter estimation

(Recall, given r: $\hat{\theta} = \text{argmax}_\theta \sum_\alpha^{N_{test}} \log \frac{p(x_\alpha|\theta)}{p_{\text{ref}}(x_\alpha)} = \text{argmax}_\theta \sum_\alpha^{N_{test}} \log r(x_\alpha|\theta)$)

- Gong-Samaniego theorem (Gong and Samaniego 1981): With a consistent estimator of the $\hat{w}_i$, and therefore of $\log r$ (under the well-specified assumption), the estimator for θ that we get by just plugging $\hat{r}$ into the MLE is asymptotically consistent with known variance!

$$\sigma^2_{\hat{\theta}} = \sigma^2_{\hat{\theta},\text{MLE}} \left(1 + \sigma^2_{\hat{\theta},\text{MLE}} A_i C_{ij} A_j\right), A_i \equiv \frac{\partial^2 \mathscr{L}}{\partial w_i \partial \theta}$$

Physics case study: Quark/gluon likelihood

- Now, $n \sim$ Pythia 8.226 quark jets, $d \sim$ Pythia 8.226 gluon jets
  - We use the dataset included in the EnergyFlow package (you can find the jets here: https://zenodo.org/records/3164691)
- Since I don't know the exact answer for the likelihood ratio, we only examine coverage on the mixture fraction task

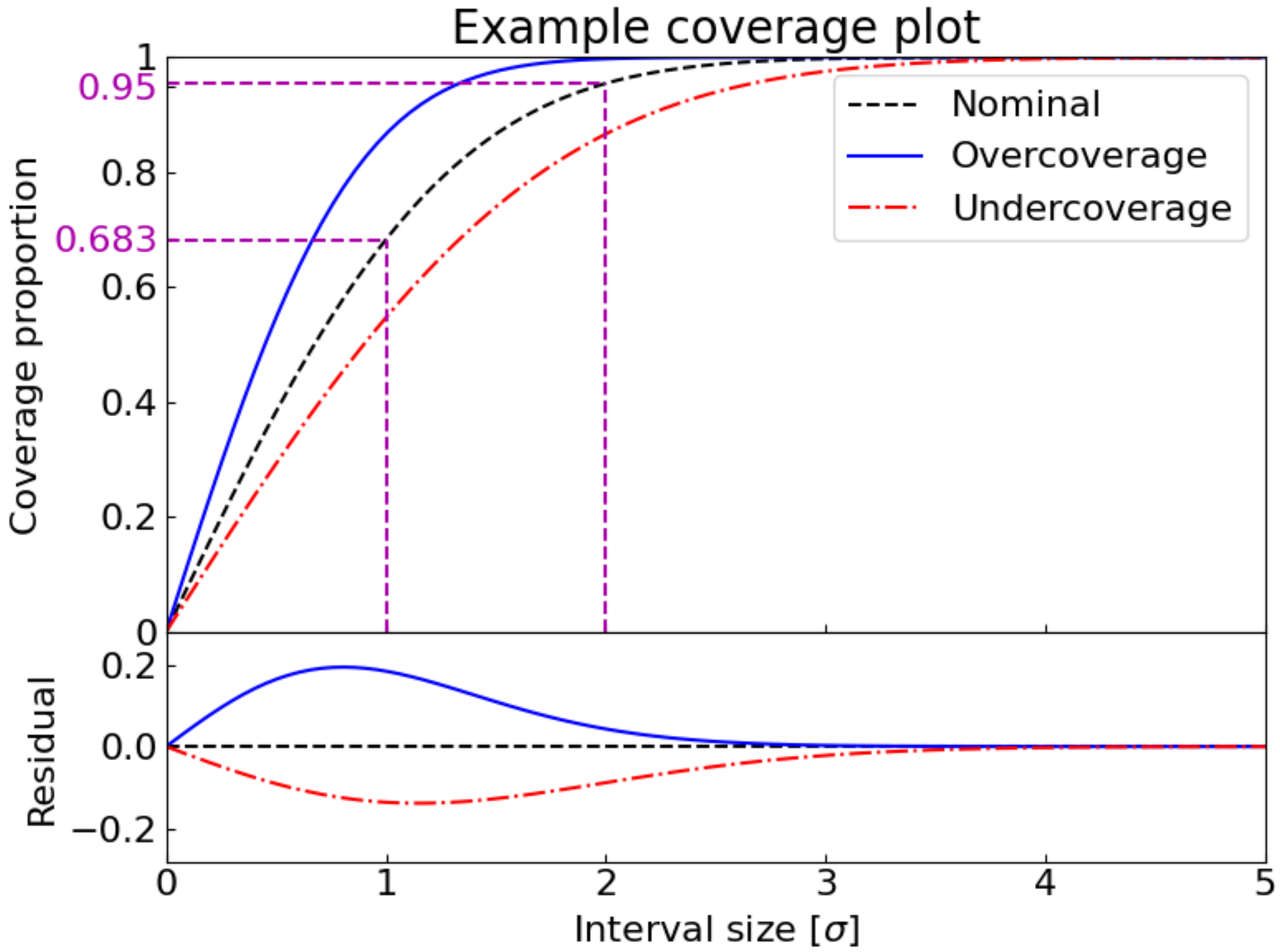
Technical detail: We construct an IRC-safe ensemble with energy-flow networks (Komiske et al. 1810.05165), so the objective function is actually the projection of $\log r$ to an IRC-safe observable

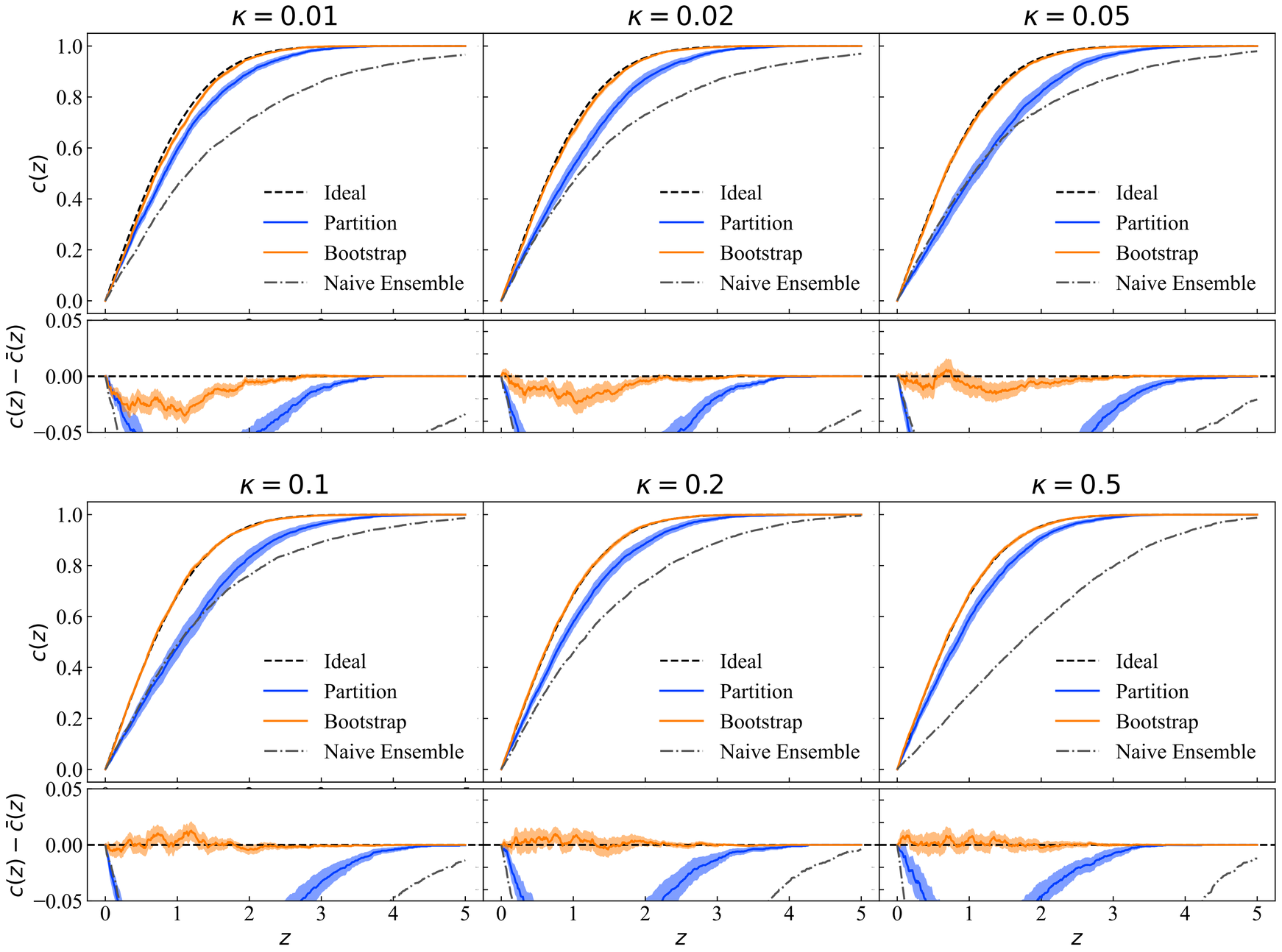
Toy SBI task: Mixture fraction estimation

Estimate the mixture fraction κ given a set of samples D drawn i.i.d. from $p(x\,|\,\kappa) = \kappa n(x) + (1-\kappa)d(x)$

$r(x)$ suffices to estimate κ with maximum likelihood estimation:

$$\hat{\kappa} = \text{argmax}_\kappa \sum_\alpha \log \frac{p(x_\alpha\,|\,\kappa)}{d(x_\alpha)} = \text{argmax}_\kappa \sum_\alpha \log(\kappa r(x_\alpha) + (1-\kappa))$$


Example coverage plot
Coverage proportion
1
0.95
0.8
0.683
0.6
0.4
0.2
0
Nominal
Overcoverage
Undercoverage
Residual
0.2
0.0
−0.2
0
1
2
3
4
5
Interval size [σ]



$N_{\text{train}} = 10{,}000$

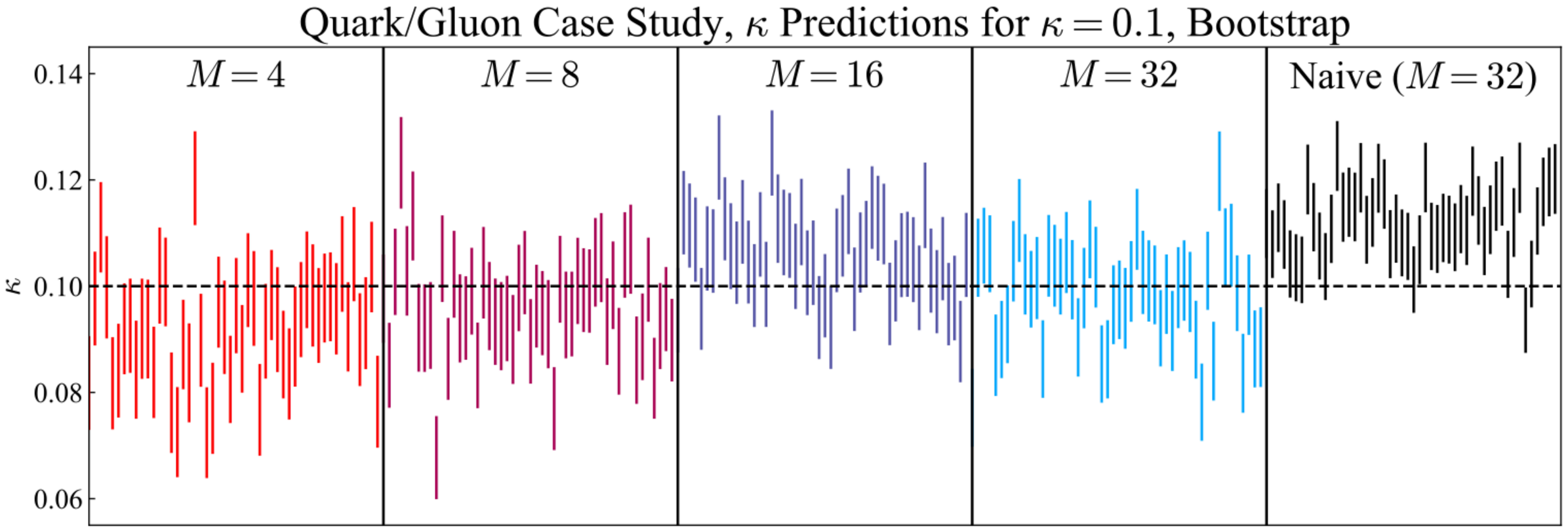
$N_{\text{fit}} = 10{,}000$

$N_{\text{trainings}} = 10$

$N_{\text{samp}} = 300$

$M = 32$

Not just uncertainties



$w_i f_i$ ensembles

$$\log r(x \mid \{w_i\}) = \sum_i w_i f_i(x)$$

where w_i are scalar weights and $f_i(x)$ are frozen networks

Pros	Cons
• Fully frequentist, no prior dependence • Intuitive: just like nuisance parameters! • Computationally efficient; no bootstrapping	• Only formally valid if model is well-specified Seems fine in practice!

SBI with biased simulators: Template-adapted mixture models

Problem statement

- We have some target data which is a mixture of signal and background $s_{\text{target}}(x)$ and $b_{\text{target}}(x)$; want to estimate signal fraction κ
- We don't have the functions $s_{\text{target}}(x)$ and $b_{\text{target}}(x)$, but we have simulated signal and background samples

$$p_{\text{target}}(x) = \kappa_{\text{target}}\, s_{\text{target}}(x) + (1-\kappa_{\text{target}})\, b_{\text{target}}(x)$$

→Compare to mixture fraction task from previous section

What if our simulations are poor?

- Despite heroic efforts, we know that in some circumstances our models are untrustworthy
    - Limited perturbative precision
    - Details of hadronization
- If our simulations are poor, inference for κ is generically biased!

$$s_{\text{sim}}(x) \neq s_{\text{target}}(x), \quad b_{\text{sim}}(x) \neq b_{\text{target}}(x)$$

Assumption that lets us make progress: Our signal and background simulations are misspecified individually, but we have several of each!

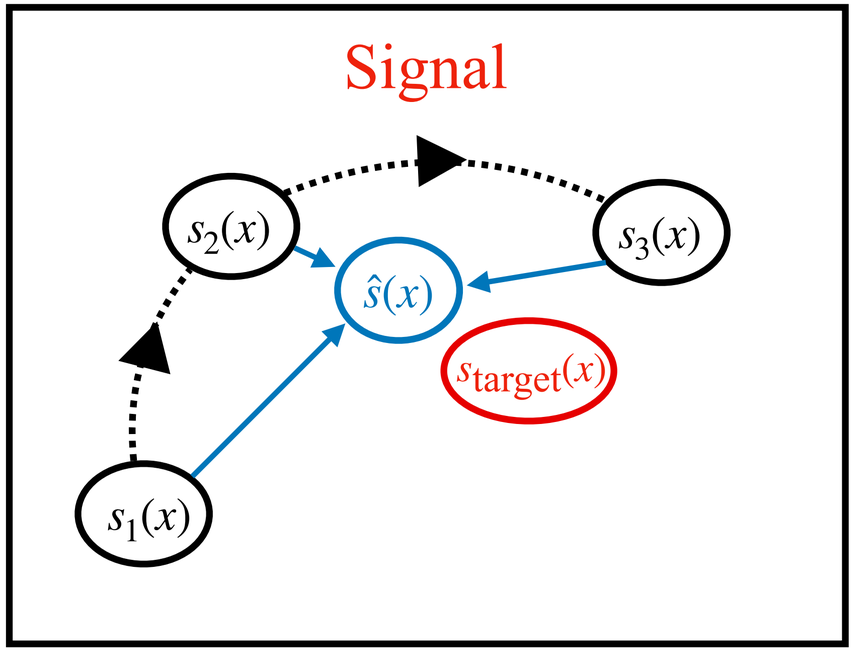
(E.g. different event generators, different showers, or even just different nuisance parameter settings)

Previous work

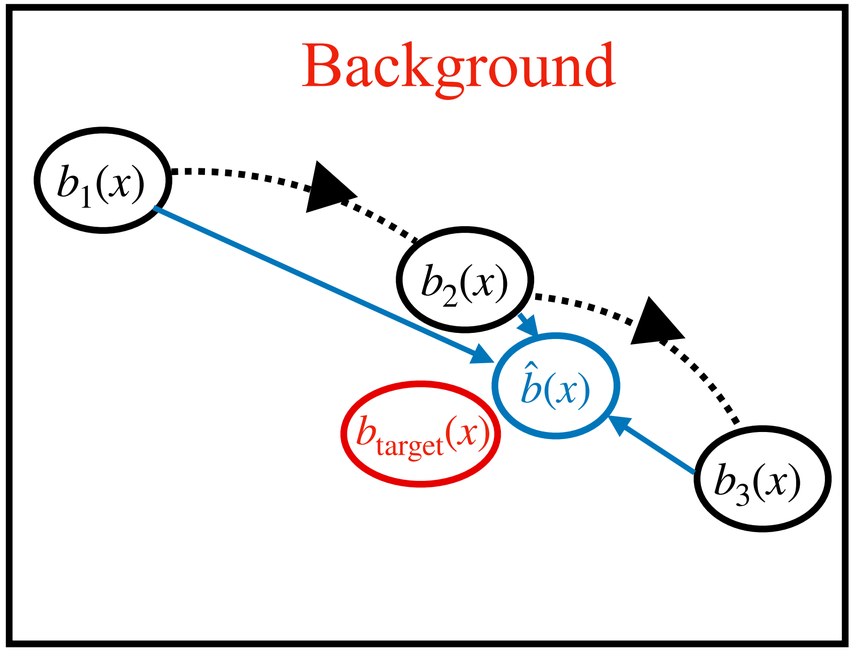
- Idea of combining simulations to model data better than any individual simulation not new, e.g. template morphing (Read 1999) and moment morphing (Baak, Gadatsch, Harrington, Verkerke 1410.7388)
  - These assume simulations are generated via different values of a set of nuisance parameters
  - They then interpolate in nuisance parameter space → only expect success if there is some value of nuisance parameters for which your simulation is correct

Template-Adapted Mixture Models (TAMMs)

$$p_{\text{target}}(x) = \kappa_{\text{target}}$$



$$+ (1 - \kappa_{\text{target}})$$



$$\text{Target Dataset} \xrightarrow{\text{infer}} \hat{\kappa}\,\hat{s}(x) + (1 - \hat{\kappa})\,\hat{b}(x)$$

TAMM Details

How do we put this into practice? Need to make some choices

1. Feature representation: Unbinned or binned
2. Model components $\mathfrak{s}_k$ and $\mathfrak{b}_k$: raw simulation or preprocessed
3. Model form: Exponential or linear
4. Statistical framework: Frequentist or Bayesian

Exponential: $$s_{\exp}(x) = c_s e^{w_k \ln \mathfrak{s}_k(x)}, \quad b_{\exp}(x) = c_b e^{v_k \ln \mathfrak{b}_k(x)}$$

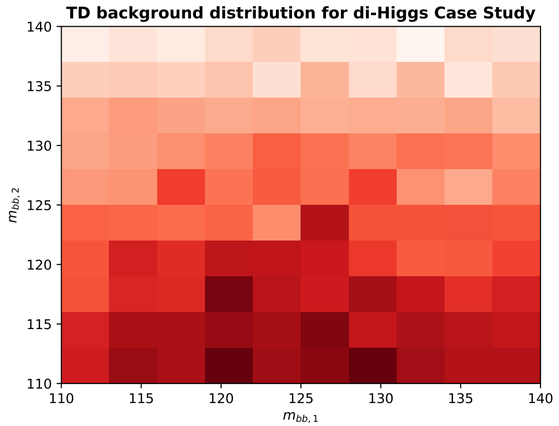
Linear: $$s_{\text{lin}}(x) = w_k \mathfrak{s}_k(x), \quad b_{\text{lin}}(x) = v_k \mathfrak{b}_k(x)$$

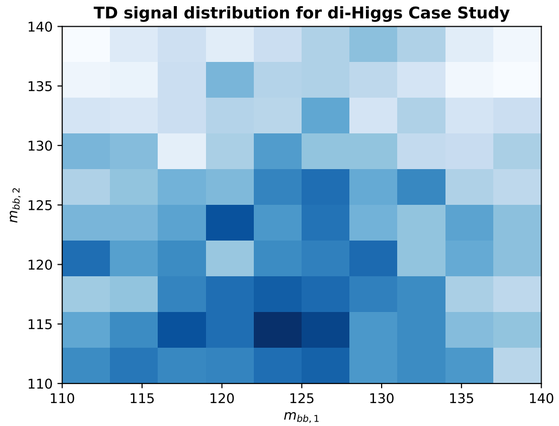
TAMM Details

	Bayesian Topic Modeling	Frequentist Neural Estimation
Representation	Binned	Unbinned
Components	Topic models	Raw simulation densities
Model	Linear	Exponential
Inference	Bayesian	Frequentist

Physics case study: simulated di-Higgs

- We take the signal to be di-Higgs to four b-jets, with nonresonant QCD background. Observed data consists of 5k signal and 50k background events
  - We generate our events in MadGraph, shower them in Pythia, and perform fast detector simulation in Delphes
- Our features are the two dijet masses obtained after pairing the b-jets which minimize squared distance from the Higgs boson mass

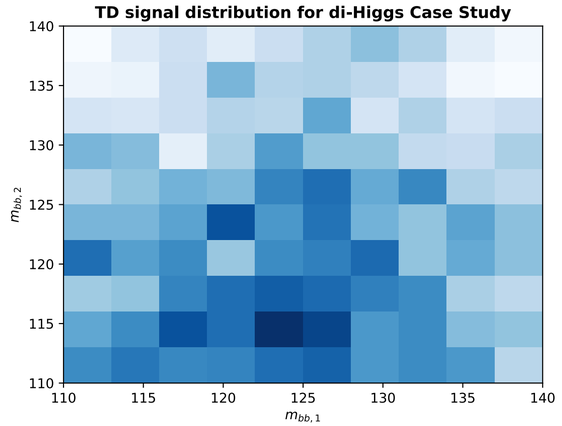




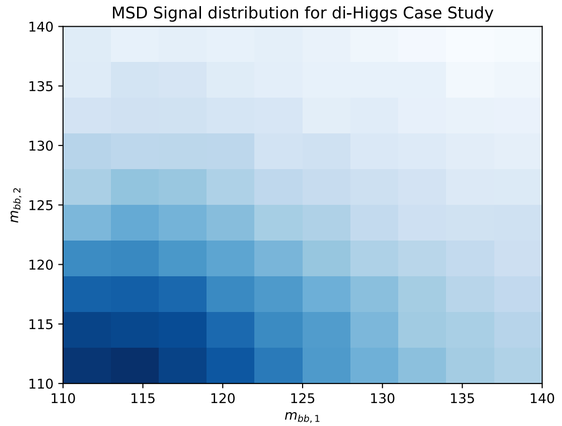
Physics case study: simulated di-Higgs

- We produce misspecified simulations (both of background and signal) by modifying the parameters of the Jet Energy Scale function in the CMS card of Delphes
- We perform a coverage study on the inferred signal fraction $\hat{\kappa}$
- All jets are available on Zenodo!

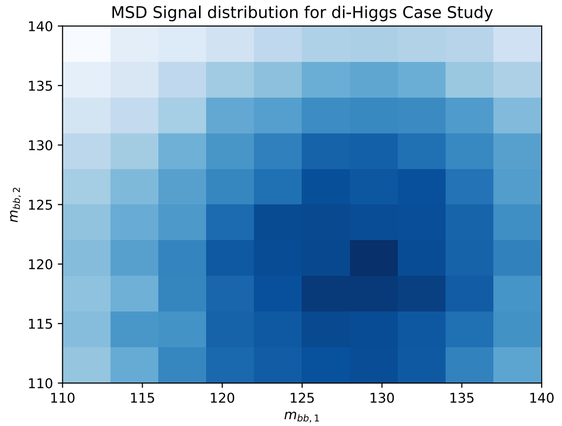
True Signal



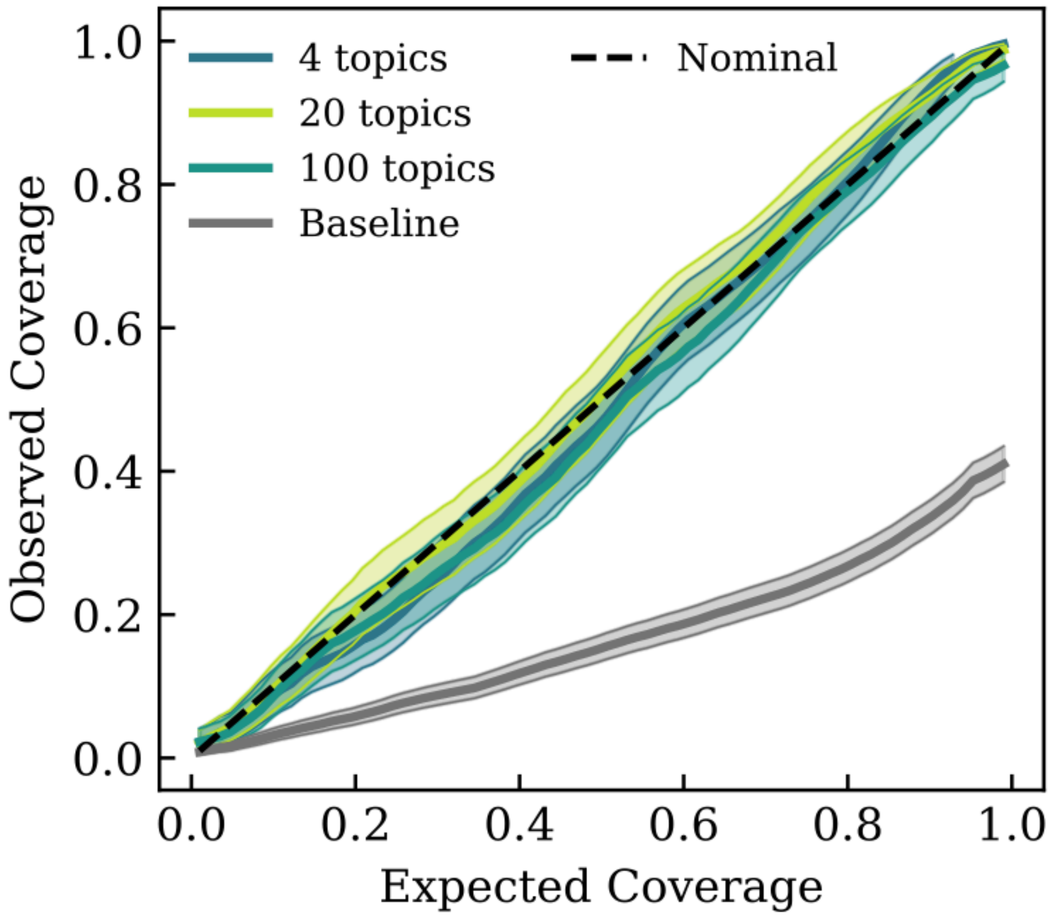
Bad Sim 1



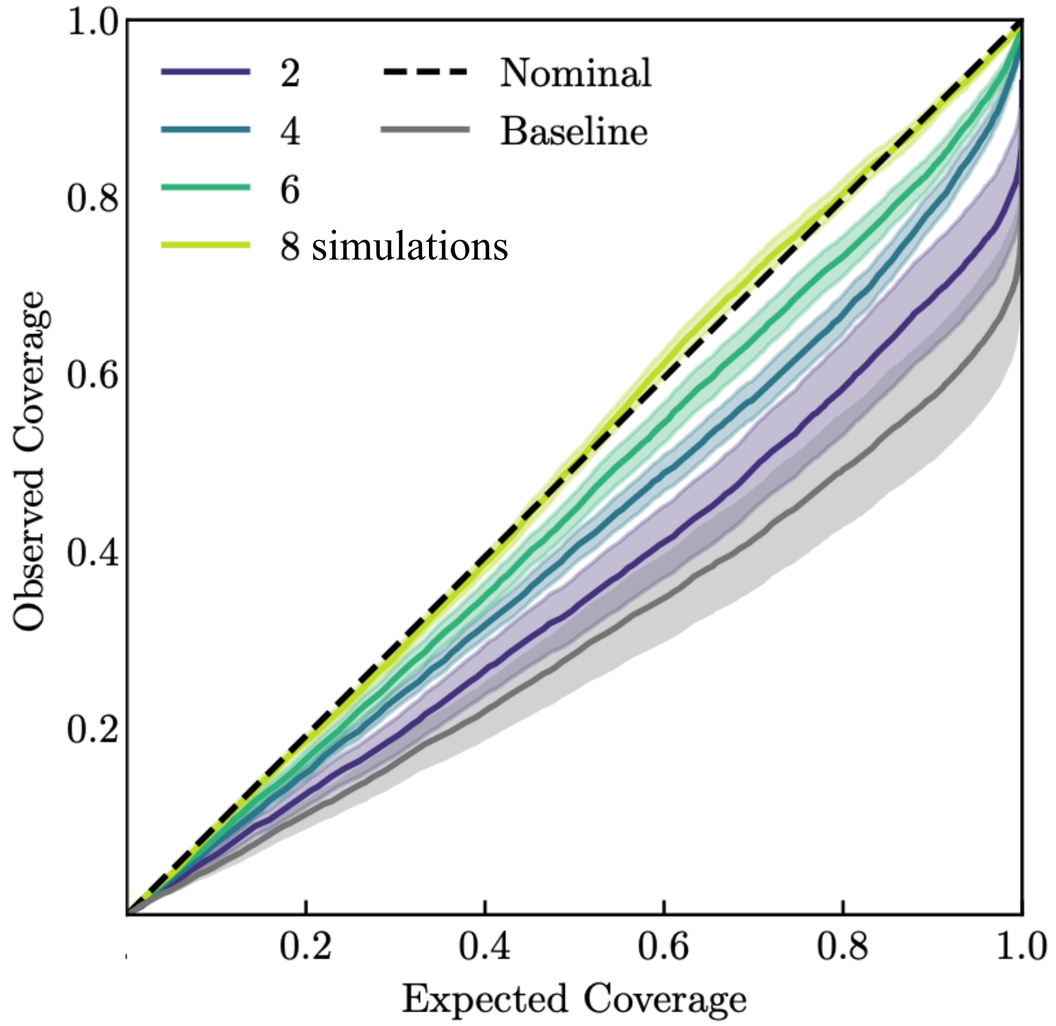
Bad Sim 2



Bayesian Topic Modeling



Frequentist Neural Estimation



TAMM Takeaways

- TAMMs achieve nominal coverage on semi-realistic signal fraction extraction with misspecified simulations!
- Bayesian Topic Modeling and Frequentist Neural Estimation are both effective, and are complementary
  - Bayesian Topic Modeling performs well with low-dimensional feature spaces, and can harness information from a large number of misspecified simulations
  - Frequentist Neural Estimation should, in principle, scale well to high-dimensional feature spaces, and achieves nominal coverage using a small number of misspecified simulations
- All jets are available on Zenodo, and try our code on GitHub!

Summary

- Two of the primary obstacles to performing trustworthy neural SBI are uncertainties and biased simulation
- $w_i f_i$ ensembles and TAMMs address these problems using classical statistics on parametrized combinations of nonparametric models
- From what we've seen, parametrized ensembles are (surprisingly!) flexible enough to achieve trustworthy coverage properties

The future for robust SBI is bright!