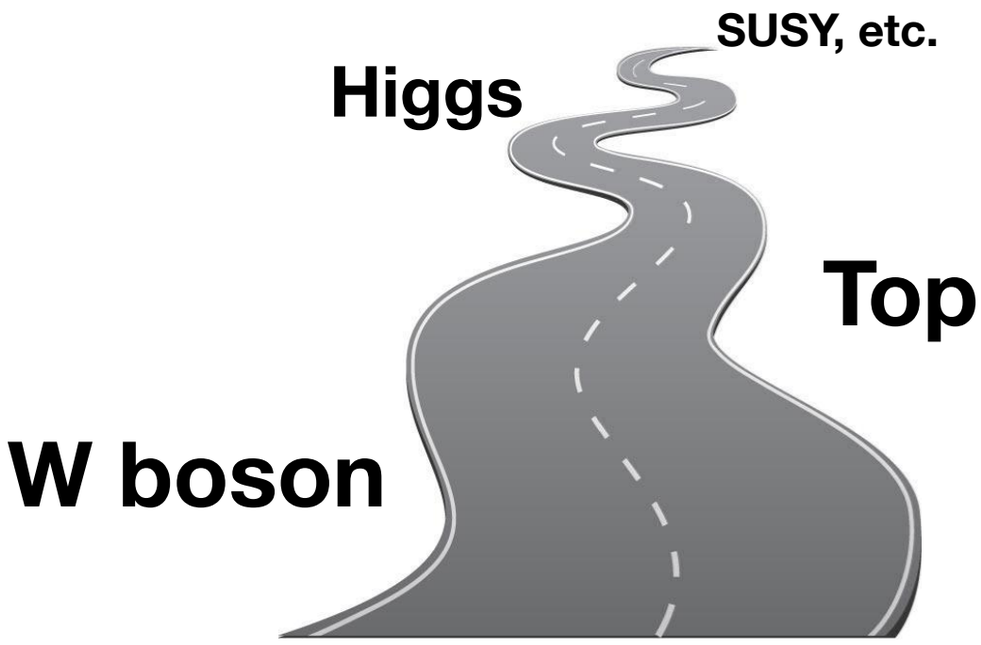
Machine Learning for BSM

Andrea Wulzer

The Challenge

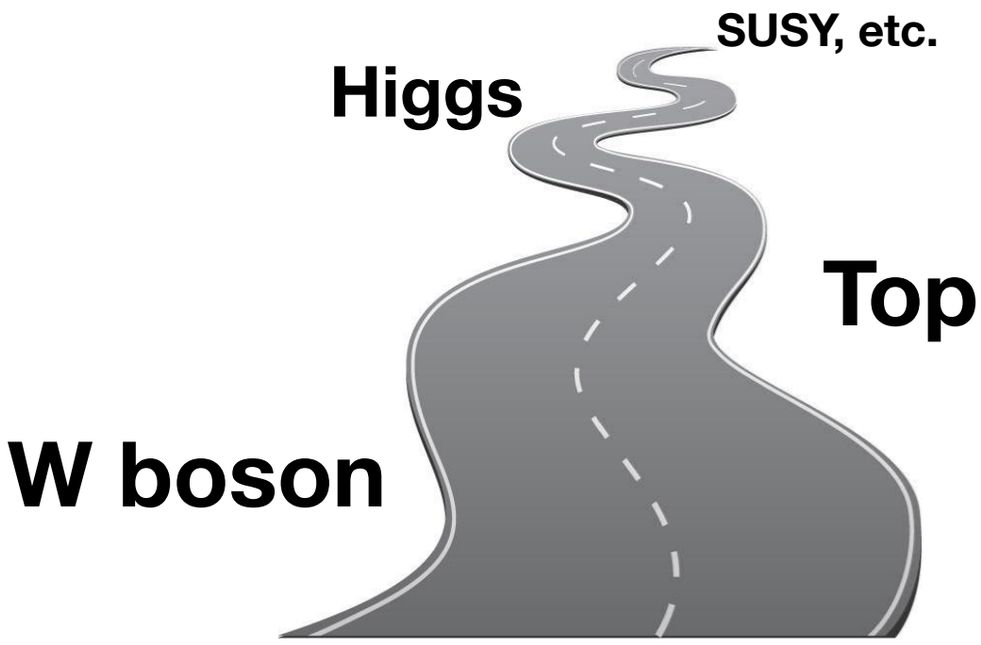
HEP yesterday



HEP today

The Challenge

HEP yesterday



HEP today

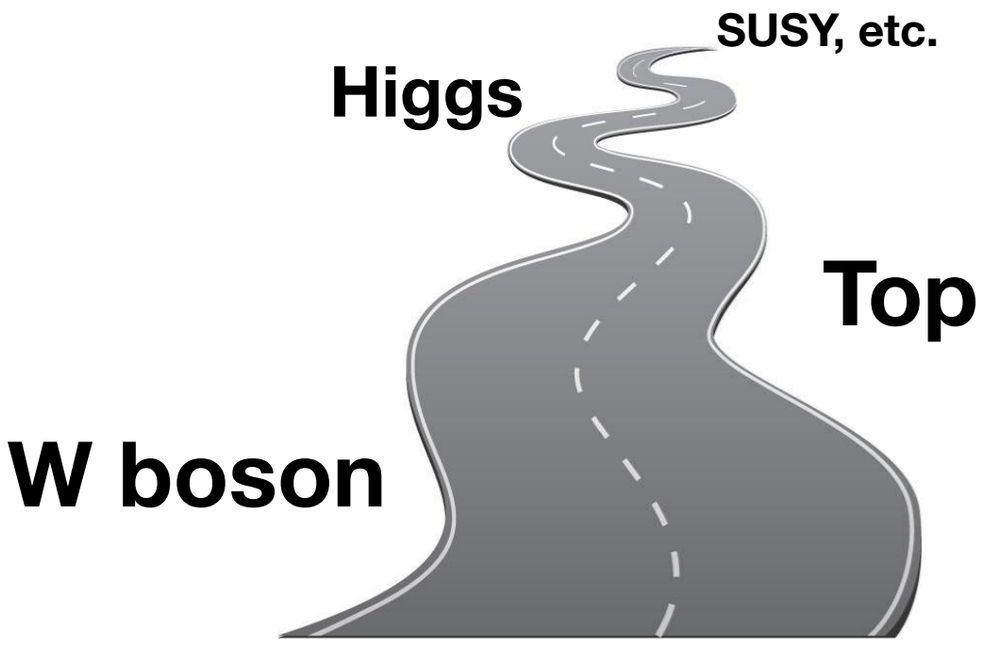
BSM physics must exist, because **SM does not predict all phenomena we observe** (e.g., DM).

But we cannot tell **what** BSM is, **if** is potentially discoverable at LHC and, if yes, **how**.

Many theoretical possibilities, **none** strongly favoured, (almost) **all** worth exploring

The Challenge

HEP yesterday



HEP today

BSM physics must exist, because **SM does not predict all phenomena we observe** (e.g., DM).

But we cannot tell **what** BSM is, **if** is potentially discoverable at LHC and, if yes, **how**.

Many theoretical possibilities, **none** strongly favoured, (almost) **all** worth exploring

Today, much less Theory Guidance

Ways Out

1) **Precision Physics**

Ways Out

1) **Precision Physics**

Search for the indirect effects of **broadly defined** BSM scenarios

SMEFT (or κ's) parametrize a **large class** of BSM models

[but not all, by far]

Ways Out

1) **Precision Physics**

Search for the indirect effects of **broadly defined** BSM scenarios

SMEFT (or κ's) parametrize a **large class** of BSM models

[but not all, by far]

2) Search for the **Unexpected**

Ways Out

1) **Precision Physics**

Search for the indirect effects of **broadly defined** BSM scenarios

SMEFT (or κ's) parametrize a **large class** of BSM models
[but not all, by far]

2) Search for the **Unexpected**

But how?

Ways Out

1) **Precision Physics**

Search for the indirect effects of **broadly defined** BSM scenarios

SMEFT (or κ's) parametrize a **large class** of BSM models
[but not all, by far]

2) Search for the **Unexpected**

But how?

Current **Model-Independent** strategies not great, but maybe …

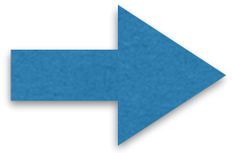
Ways Out

1) **Precision Physics**

Search for the indirect effects of **broadly defined** BSM scenarios

SMEFT (or κ's) parametrize a **large class** of BSM models

[but not all, by far]

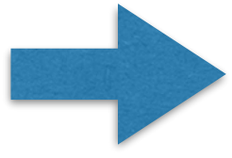


Teaching New Physics **to** a Machine

2) Search for the **Unexpected**

But how?

Current **Model-Independent** strategies not great, but maybe …



Learning New Physics **from** a Machine

Teaching New Physics to a Machine

Properly named: **Likelihood Free Inference**

[for **EFT** applications, see last recent work by K.Crammer et.al.]

Statistical Foundation: **Neyman—Pearson Lemma**

[J. Neyman and E. S. Pearson, 1933]

- Given:

  Data: $\mathcal{D} = \{x_i\},\ i = 1, \ldots, \mathcal{N}_\mathcal{D}$

  Reference Distribution: $n(x|\mathrm{R})$

  Alternative Distribution: $n(x|\mathrm{NP})$

$$n(x) = N\,P(x)$$
$$N = \int dx\, n(x)$$

- The **optimal*** test statistic variable is the likelihood ratio

$$t = 2\log\left[\frac{e^{-N(\mathrm{NP})}}{e^{-N(\mathrm{R})}}\prod_{i=1}^{\mathcal{N}_\mathcal{D}}\frac{n(x_i|\mathrm{NP})}{n(x_i|\mathrm{R})}\right]$$

Which however is **never known** in closed form.
Our predictions are Monte Carlo following R or NP dist.

* Maximum power at given size

Teaching New Physics to a Machine

Old-Fashion implementation is **Matrix-Element Method**

Use analytical phenomenological parametrization of the lik.ratio

So much hard to set up that almost never used

Modern Machine Learning version:

Train a classifier on R vs NP. It converges to:

$$C(x) = n(x|\mathrm{NP})/[n(x|\mathrm{NP}) + n(x|\mathrm{R})]$$

Invert, compute lik.ratio and use it for test

Natural application to SMEFT precision searches

Teaching New Physics to a Machine

Old-Fashion implementation is **Matrix-Element Method**

Use analytical phenomenological parametrization of the lik.ratio

So much hard to set up that almost never used

Modern Machine Learning version:

Train a classifier on R vs NP. It converges to:

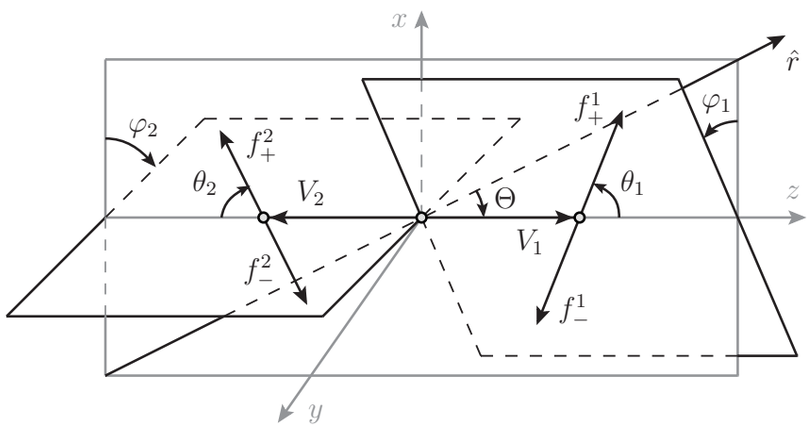
$$C(x) = n(x|\mathrm{NP})/[n(x|\mathrm{NP}) + n(x|\mathrm{R})]$$

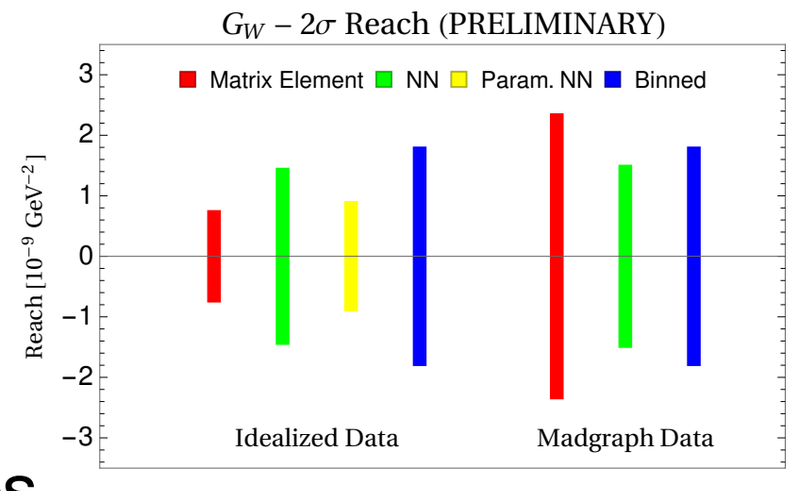
Invert, compute lik.ratio and use it for test

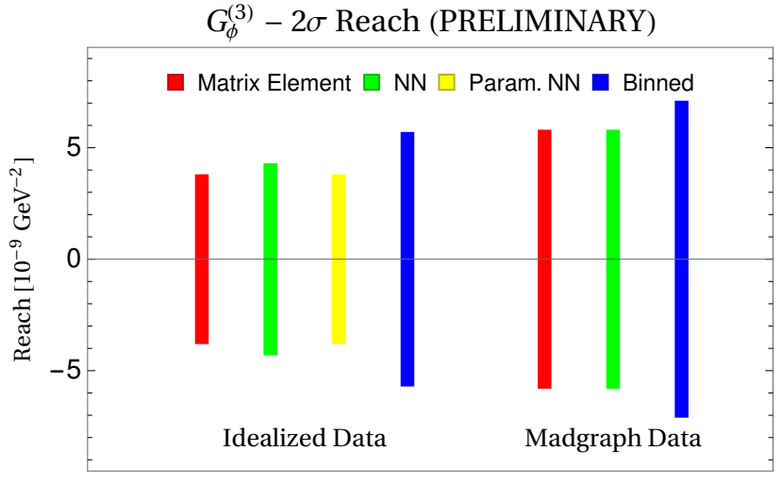
Natural application to SMEFT precision searches

Fully-leptonic WZ is an interesting case study

[Chen, Glioti, Panico, AW, in progress]

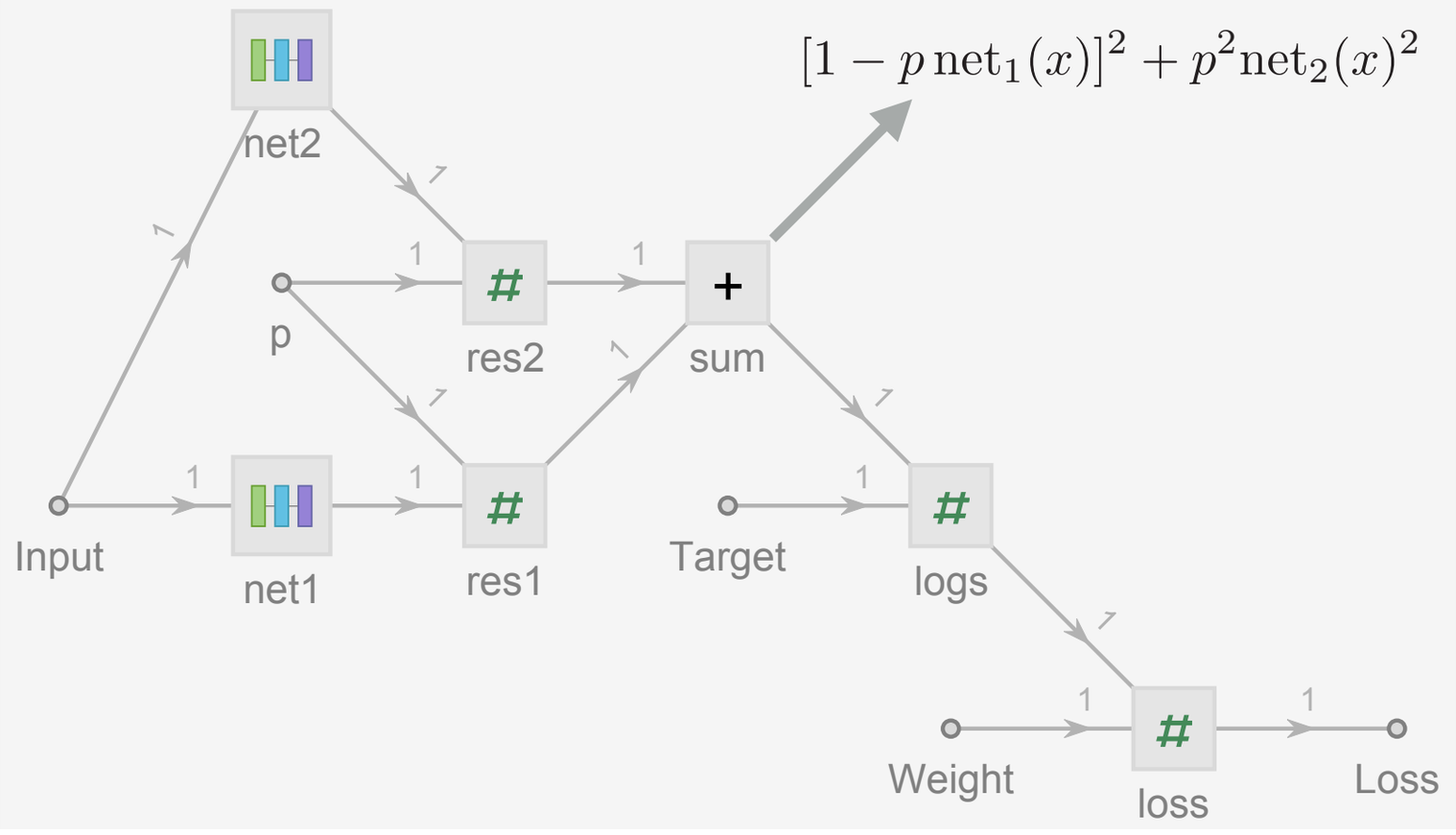






Six discriminating variables

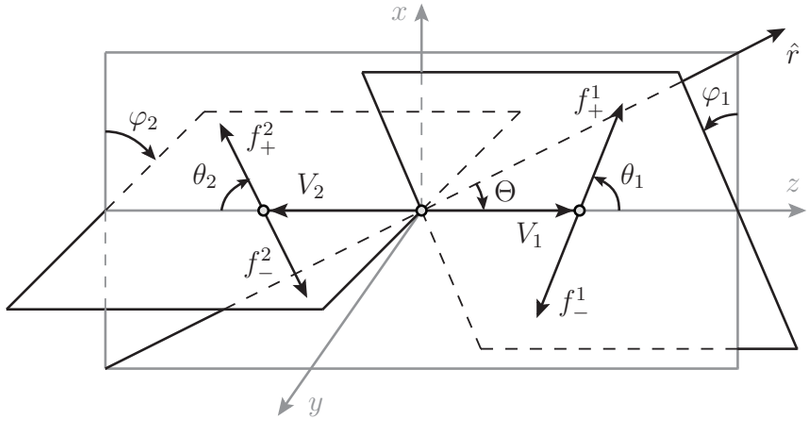
Teaching New Physics to a Machine



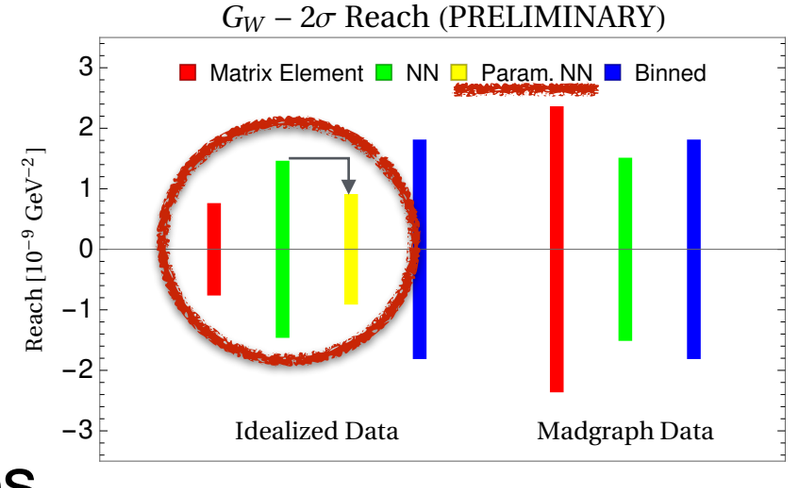
“Parametrised” NN:

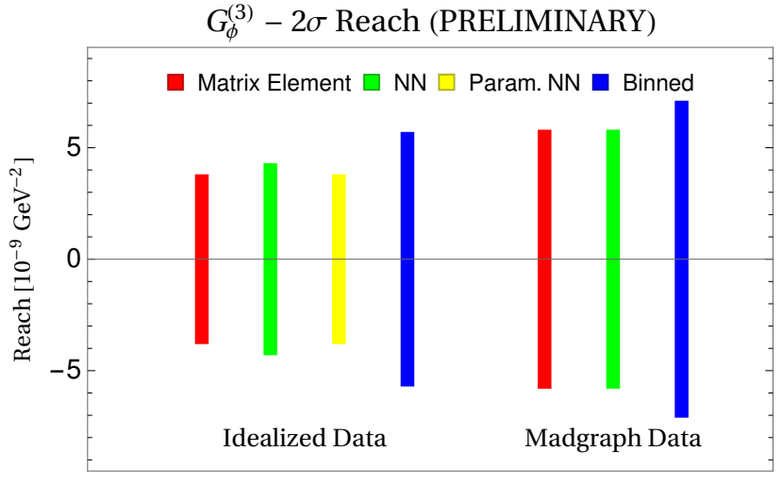
Quadratic dependence on the SMEFT Wilson Coefficients is **built-in**

[p-dependent NN also called “parametrised”.
Looking for a better name…]



Six discriminating variables





Learning New Physics from a Machine

[D'Agnolo, AW, arXiv:1806.02350]

New Physics Search:

algorithm aimed at detecting **data departures** from a given **Reference Model**

Model-Independent NP Search:

ideally sensitive to "any" NP model, rather than to specific "BSM" alternatives

Disadvantage: negative M-I searches are **not informative**

Advantage: might **discover** model we had not thought of

Important Remark: [not only to please statisticians]

Hypothesis test **unavoidably requires alternative** hypothesis, or **probability model**, to compare with

M-I physically means that the alternative distribution is not selected as the one predicted by known alternative physics model

Maximum Likelihood

[J. Neyman and E. S. Pearson, 1933]

Data: $\mathcal{D} = \{x_i\},\ i = 1, \ldots, \mathcal{N}_\mathcal{D}$

Reference Distribution: $n(x|\mathrm{R})$

Alternative Distribution: $n(x|\mathbf{w})$

depending on **parameters** (composite)

$$n(x) = N\,P(x)$$
$$N = \int dx\, n(x)$$

$$t(\mathcal{D}) = 2\,\underset{\mathbf{w}}{\mathrm{Max}} \left\{ \log \left[\frac{e^{-N(\mathbf{w})}}{e^{-N(\mathrm{R})}} \prod_{i=1}^{\mathcal{N}_\mathcal{D}} \frac{n(x_i|\mathbf{w})}{n(x_i|\mathrm{R})} \right] \right\} =$$

Fully Model-Dependent (ideal):

$$2 \log \left[\frac{e^{-N(\mathrm{NP})}}{e^{-N(\mathrm{R})}} \prod_{i=1}^{\mathcal{N}_\mathcal{D}} \frac{n(x_i|\mathrm{NP})}{n(x_i|\mathrm{R})} \right] = t_{\mathrm{id}}$$

Model-Independent:

$$-2\,\underset{\mathbf{w}}{\mathrm{Min}} \left[N(\mathbf{w}) - N(\mathrm{R}) - \sum_{i=1}^{\mathcal{N}_\mathcal{D}} f(x_i; \mathbf{w}) \right]$$

Alternative in parametrised form: $n(x|\mathbf{w}) = n(x|\mathrm{R})\, e^{f(x;\mathbf{w})}$

Maximum Likelihood

[J. Neyman and E. S. Pearson, 1933]

Data: $\mathcal{D} = \{x_i\},\ i = 1, \ldots, \mathcal{N}_\mathcal{D}$

Reference Distribution: $n(x|\mathrm{R})$

Alternative Distribution: $n(x|\mathbf{w})$

depending on **parameters** (composite)

$$n(x) = N\,P(x)$$
$$N = \int dx\, n(x)$$

$$t(\mathcal{D}) = 2\,\underset{\mathbf{w}}{\mathrm{Max}} \left\{ \log \left[\frac{e^{-N(\mathbf{w})}}{e^{-N(\mathrm{R})}} \prod_{i=1}^{\mathcal{N}_\mathcal{D}} \frac{n(x_i|\mathbf{w})}{n(x_i|\mathrm{R})} \right] \right\} =$$

Fully Model-Dependent (ideal):

$$2 \log \left[\frac{e^{-N(\mathrm{NP})}}{e^{-N(\mathrm{R})}} \prod_{i=1}^{\mathcal{N}_\mathcal{D}} \frac{n(x_i|\mathrm{NP})}{n(x_i|\mathrm{R})} \right] = t_{\mathrm{id}}$$

Model-Independent:

$$-2\,\underset{\mathbf{w}}{\mathrm{Min}} \left[N(\mathbf{w}) - N(\mathrm{R}) - \sum_{i=1}^{\mathcal{N}_\mathcal{D}} f(x_i;\mathbf{w}) \right]$$

Alternative in parametrised form: $n(x|\mathbf{w}) = n(x|\mathrm{R})\, e^{f(x;\mathbf{w})}$

If f piece-wise constant in bins: $t(\mathcal{D}) = 2 \sum_{\alpha=1}^{N_\mathrm{bin}} \left[N_\alpha(\mathrm{R}) - O_\alpha + O_\alpha \log \frac{O_\alpha}{N_\alpha(\mathrm{R})} \right]$

recover **binned histogram test !**

Maximum Likelihood

[D'Agnolo, AW, arXiv:1806.02350]

Basic idea: $f(x;\mathbf{w}) = \mathrm{NN}$

replace histograms with NN, literally!

Highly motivated attempt:

- NN "effective" unbiased function approximants
- **NNPDF** exploits this virtue since 2002
- Often introduced as **alternative to histograms** to fit distributions
- Better dimensionality scaling

$$-2 \min_{\mathbf{w}} \left[N(\mathbf{w}) - N(\mathrm{R}) - \sum_{i=1}^{N_{\mathcal{D}}} f(x_i;\mathbf{w}) \right]$$

Alternative in parametrised form: $n(x|\mathbf{w}) = n(x|\mathrm{R})\, e^{f(x;\mathbf{w})}$

If f piece-wise constant in bins: $t(\mathcal{D}) = 2 \sum_{\alpha=1}^{N_{\mathrm{bin}}} \left[N_\alpha(\mathrm{R}) - O_\alpha + O_\alpha \log \frac{O_\alpha}{N_\alpha(\mathrm{R})} \right]$

recover **binned histogram test !**

Maximum Likelihood Loss

[D'Agnolo, AW, arXiv:1806.02350]

Easy to turn the evaluation of "t" into supervised training problem:

$$t(\mathcal{D}) = -2 \operatorname*{Min}_{\mathbf{w}} \left[N(\mathbf{w}) - N(\mathrm{R}) - \sum_{i=1}^{\mathcal{N}_\mathcal{D}} f(x_i; \mathbf{w}) \right] \qquad N(\mathbf{w}) = \int dx\, n(x|\mathrm{R})\, e^{f(x;\mathbf{w})}$$

Use **Reference Sample**, distributed according to the Reference Model

$$\mathcal{R} = \{x_i\},\ i = 1, \ldots, \mathcal{N}_\mathcal{R}$$

Approximate integral as Monte Carlo sum:

$$N(\mathbf{w}) = \int dx\, n(x|\mathrm{R})\, e^{f(x;\mathbf{w})} = \frac{N(\mathrm{R})}{\mathcal{N}_\mathcal{R}} \sum_{x \in \mathcal{R}} e^{f(x;\mathbf{w})}$$

Maximum Likelihood Loss

[D'Agnolo, AW, arXiv:1806.02350]

Easy to turn the evaluation of "t" into supervised training problem:

$$t(\mathcal{D}) = -2 \operatorname*{Min}_{\mathbf{w}} \left[N(\mathbf{w}) - N(\mathrm{R}) - \sum_{i=1}^{\mathcal{N}_\mathcal{D}} f(x_i;\mathbf{w}) \right] \qquad N(\mathbf{w}) = \int dx\, n(x|\mathrm{R})\, e^{f(x;\mathbf{w})}$$

Use **Reference Sample**, distributed according to the Reference Model

$$\mathcal{R} = \{x_i\},\ i = 1, \ldots, \mathcal{N}_\mathcal{R}$$

Approximate integral as Monte Carlo sum:

$$N(\mathbf{w}) = \int dx\, n(x|\mathrm{R})\, e^{f(x;\mathbf{w})} = \frac{N(\mathrm{R})}{\mathcal{N}_\mathcal{R}} \sum_{x \in \mathcal{R}} e^{f(x;\mathbf{w})}$$

Get **t = -2 * minimal loss.** The trained **net is distribution log ratio**

$$t(\mathcal{D}) = -2 \operatorname*{Min}_{\{\mathbf{w}\}} \left[\frac{N(\mathrm{R})}{\mathcal{N}_\mathcal{R}} \sum_{x \in \mathcal{R}} (e^{f(x;\mathbf{w})} - 1) - \sum_{x \in \mathcal{D}} f(x;\mathbf{w}) \right] \equiv -2 \operatorname*{Min}_{\{\mathbf{w}\}} L[f(\,\cdot\,, \mathbf{w})]$$

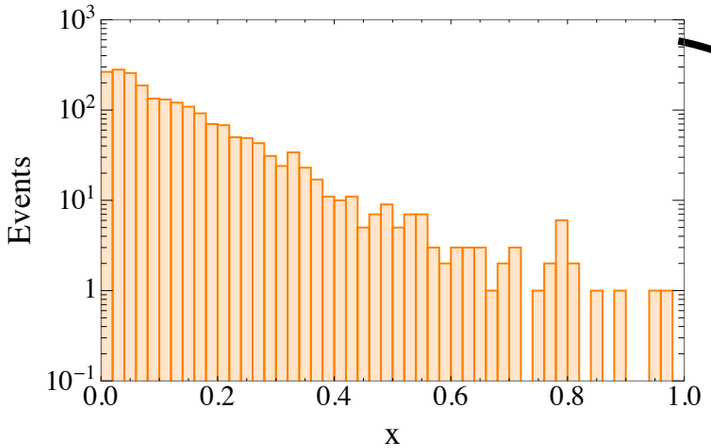
$$L[f] = \sum_{(x,y)} \left[(1-y) \frac{N(\mathrm{R})}{\mathcal{N}_\mathcal{R}} (e^{f(x)} - 1) - y\, f(x) \right]$$

The Algorithm

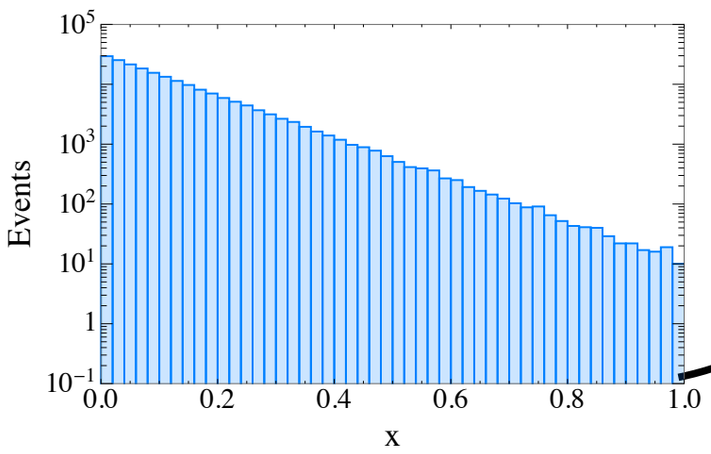
[D'Agnolo, AW, arXiv:1806.02350]

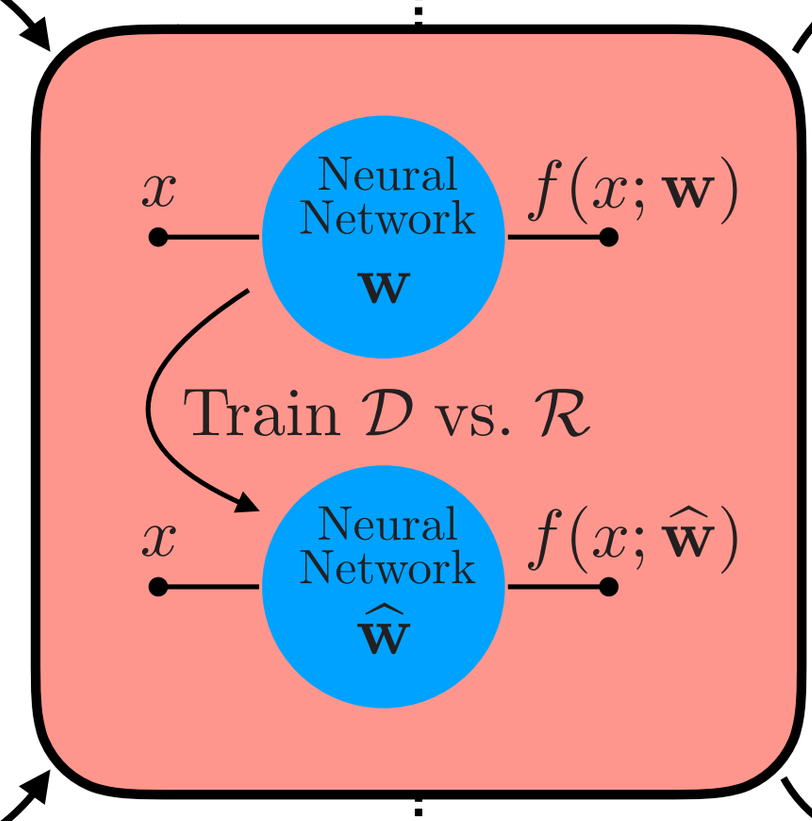
INPUT

Data sample $\mathcal{D}$



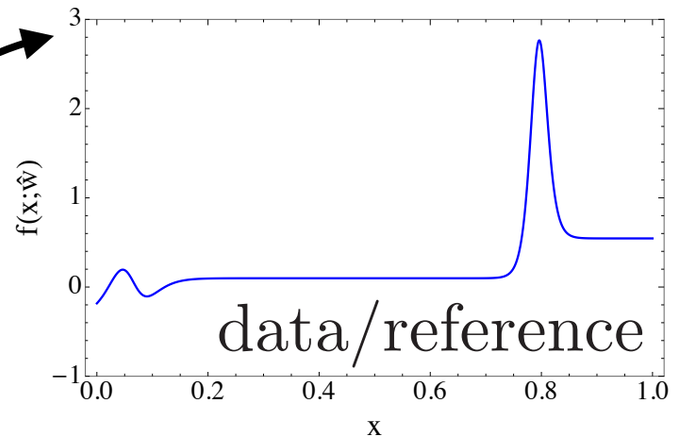
Reference sample $\mathcal{R}$





OUTPUT

Dist. log ratio



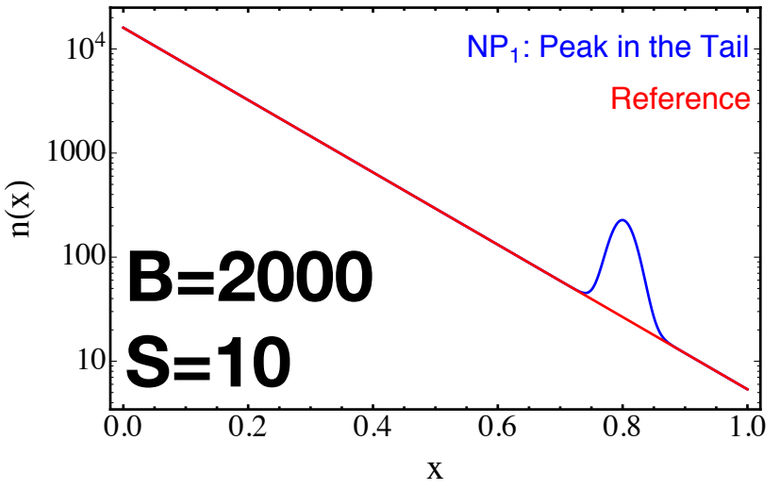
$$f(x;\widehat{\mathbf{w}}) \simeq \log\left[\frac{n(x|\mathrm{T})}{n(x|\mathrm{R})}\right]$$

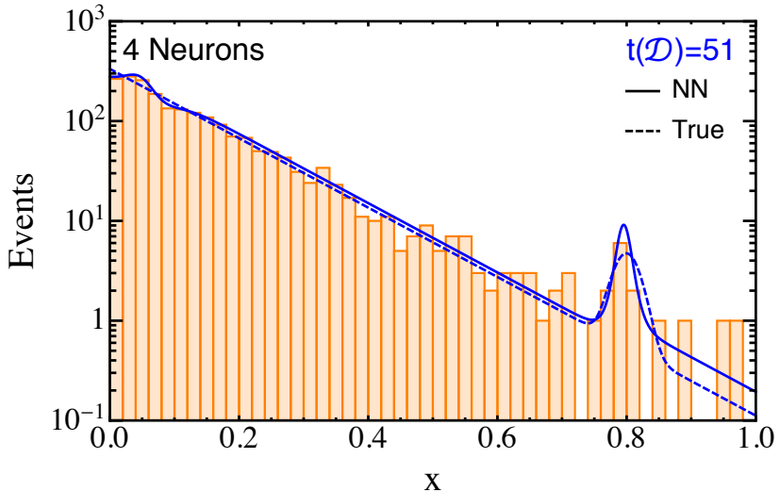
Test statistic t computed on the data sample $\mathcal{D}$

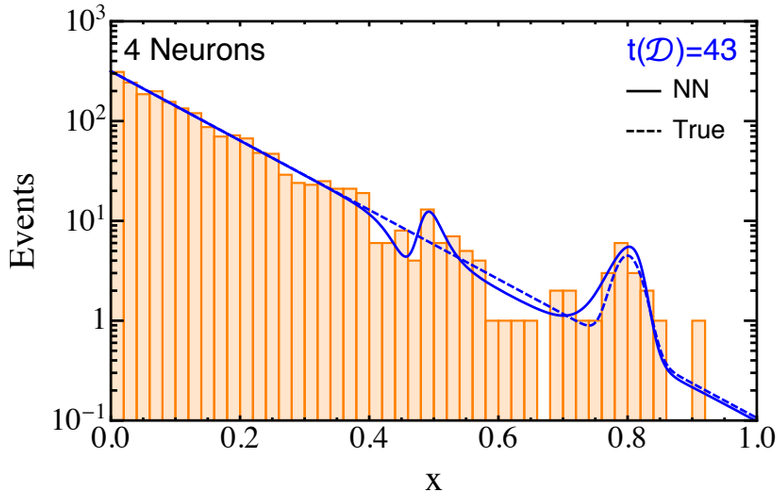
$$t(\mathcal{D}) = -2 \operatorname*{Min}_{\{\mathbf{w}\}} L[f]$$

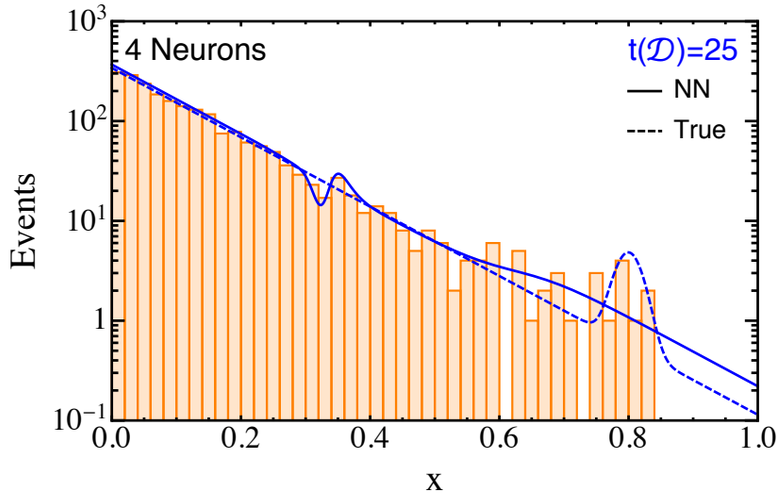
Illustrating Performances

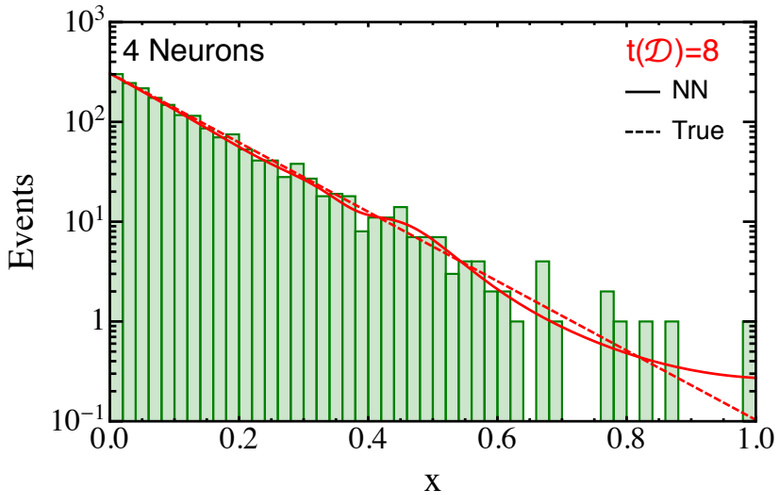
[D'Agnolo, AW, arXiv:1806.02350]

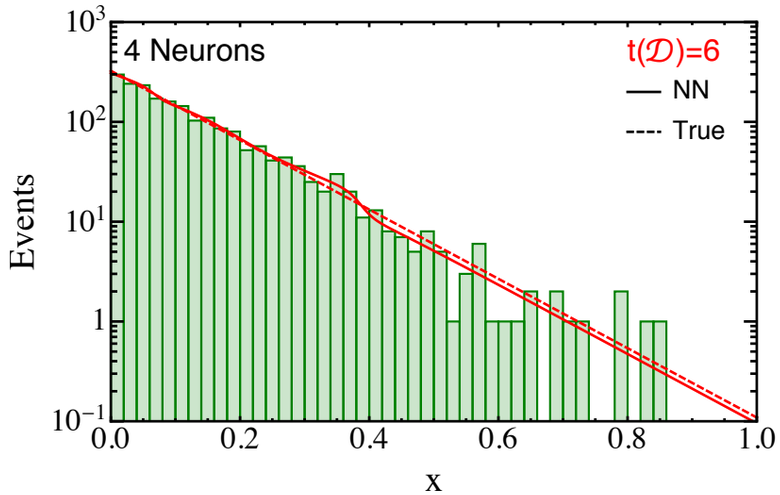


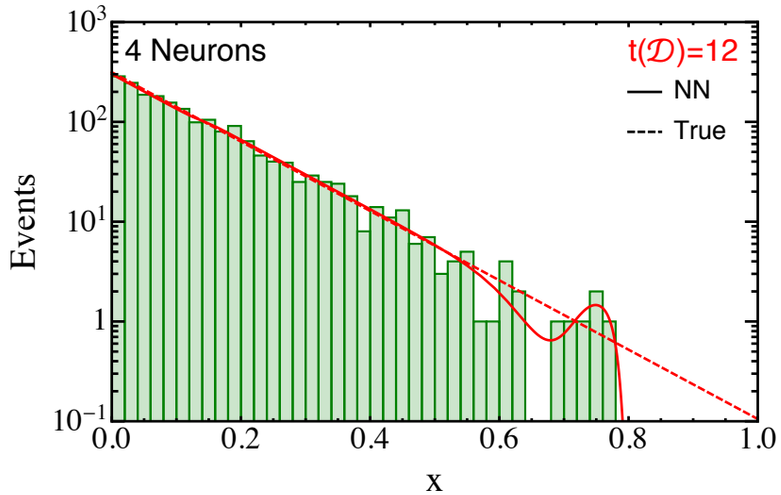






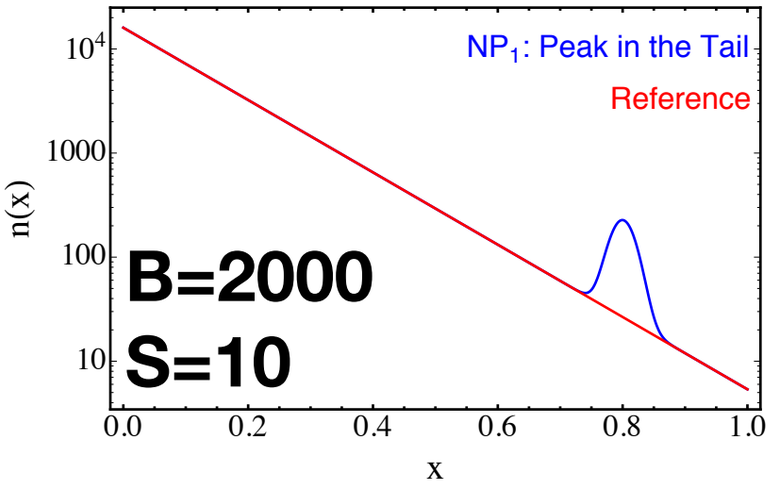


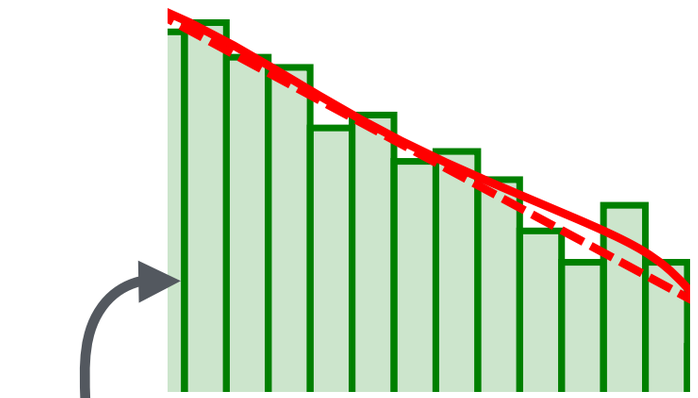




Illustrating Performances

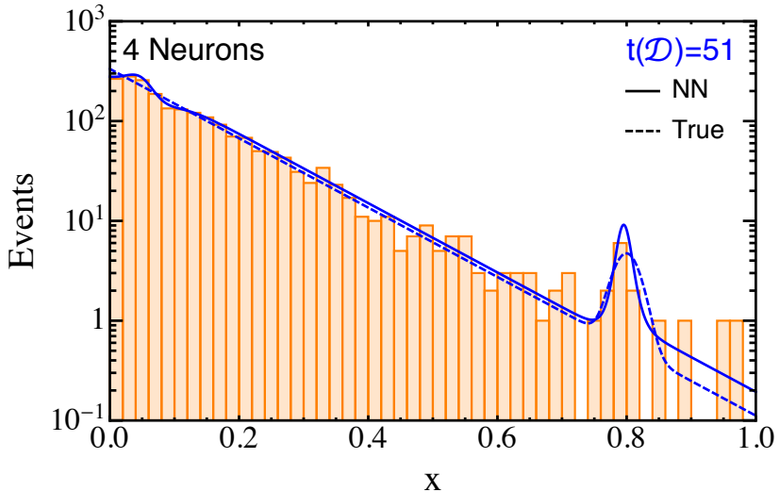
[D'Agnolo, AW, arXiv:1806.02350]

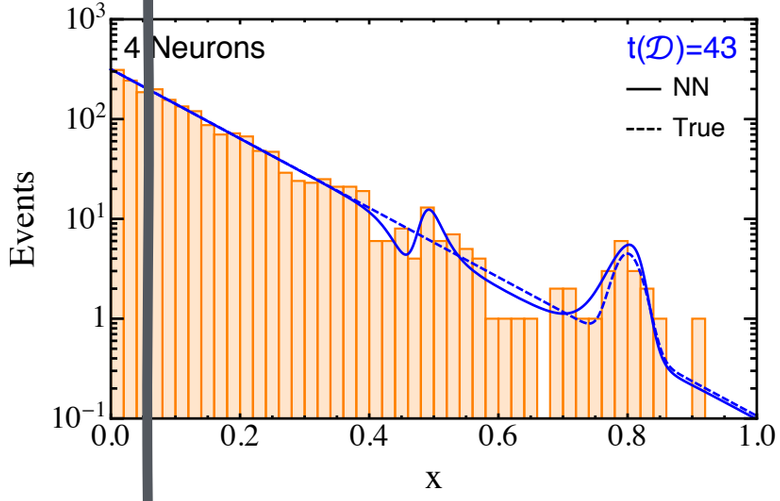


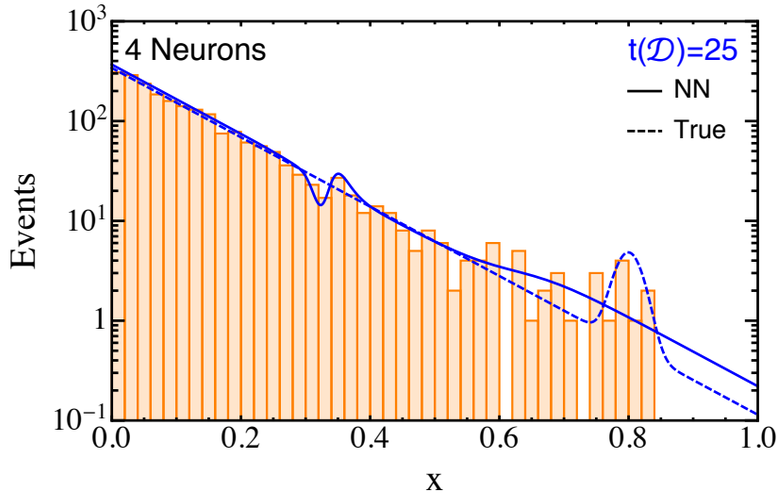


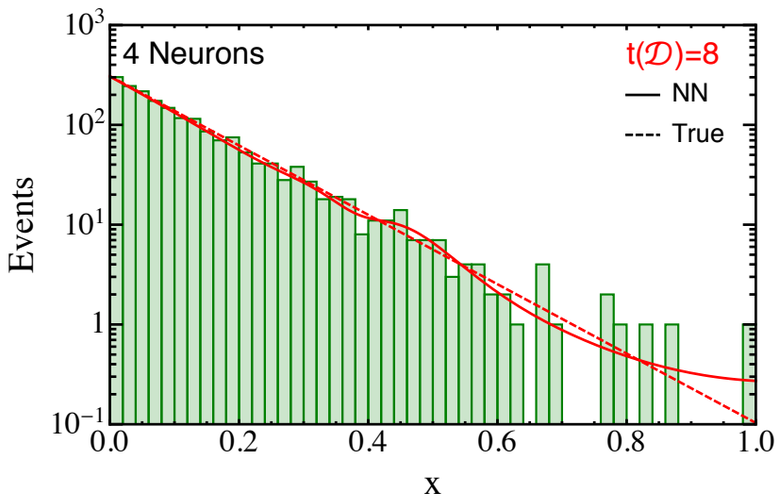
Bins: Non-discrepant data fluctuations wash out reach

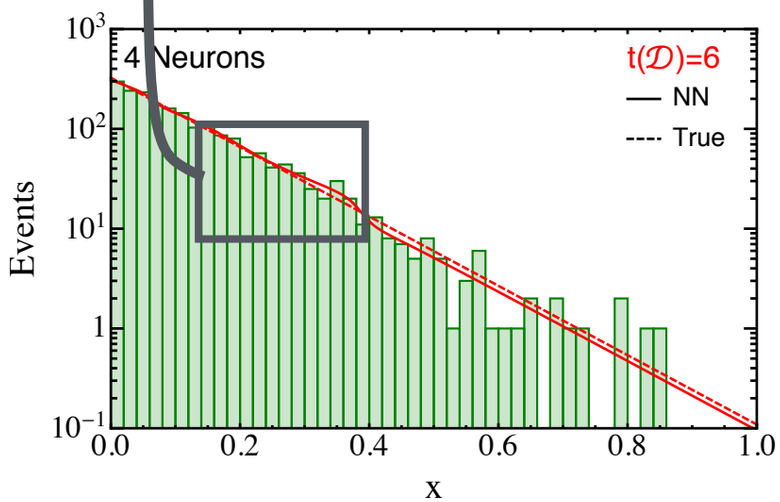
NN: Smooth curve. Can handle non-discrepant data

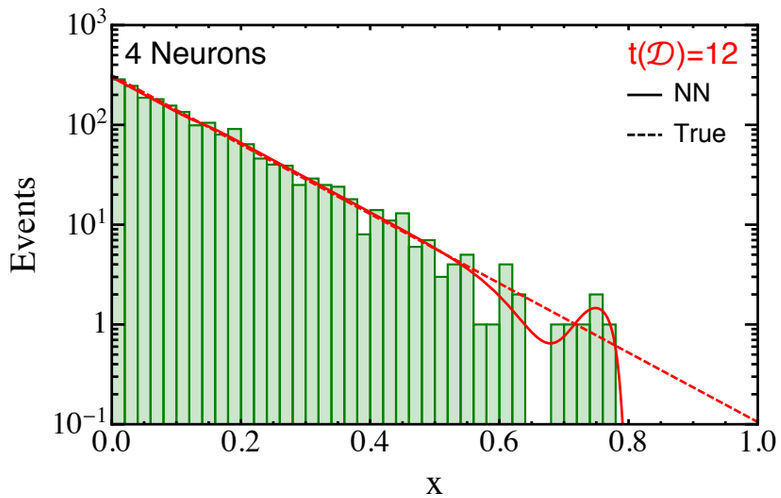










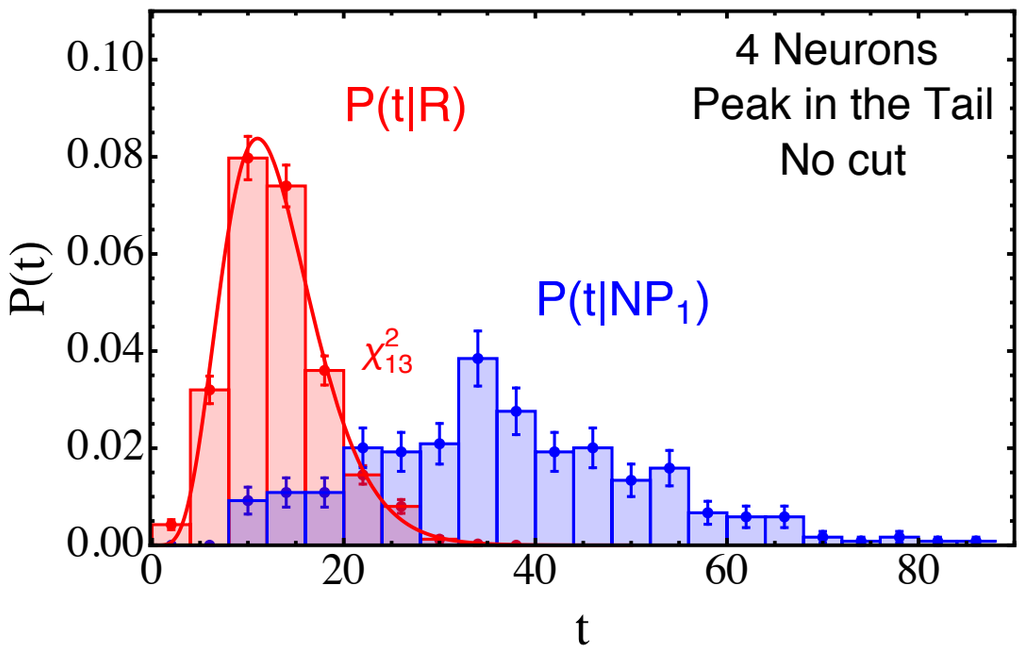


Quantifying Performances

[D'Agnolo, AW, arXiv:1806.02350]

Our Z-score vs **Ideal** Z-score

Run over R and NP toys [repeat train.]
Compute NP p-value distribution

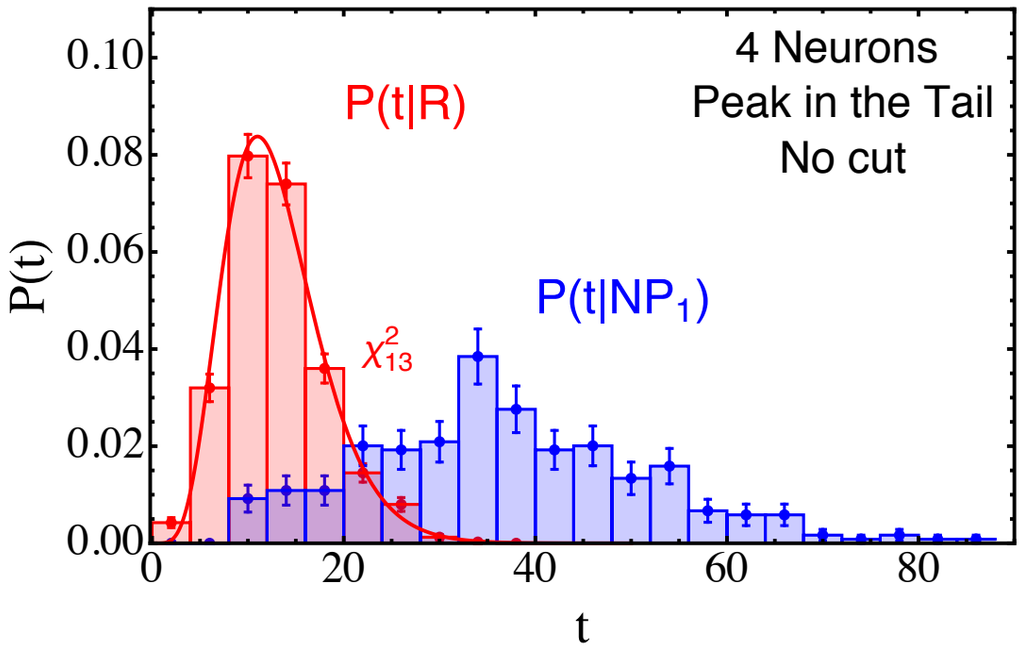


Quantifying Performances

[D'Agnolo, AW, arXiv:1806.02350]

Our Z-score vs **Ideal** Z-score

Run over R and NP toys [repeat train.]
Compute NP p-value distribution



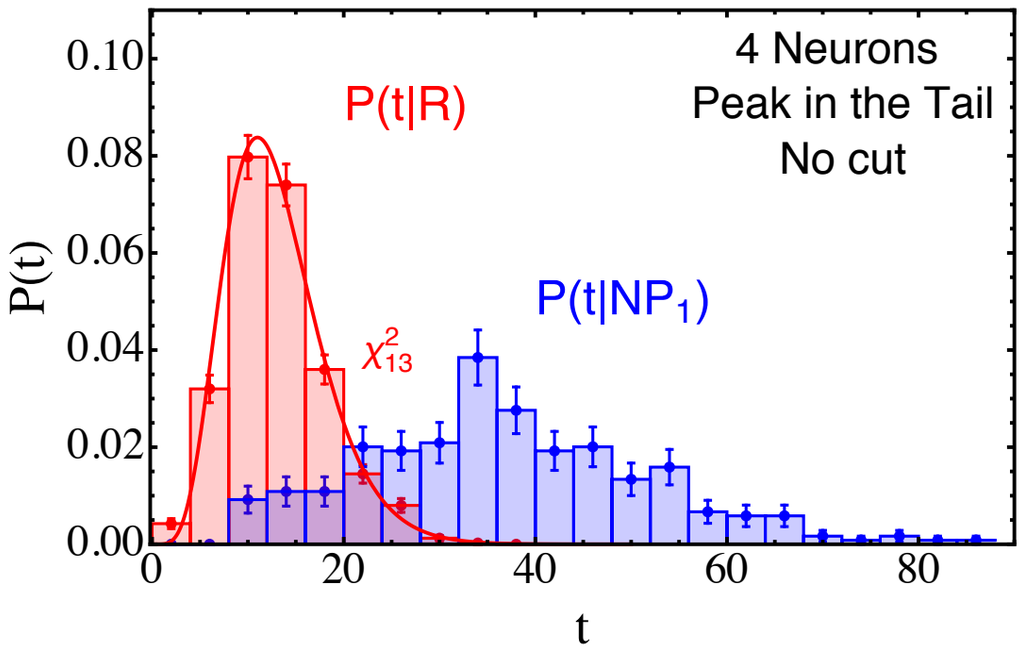
How difficult is to see "NP"?

Quantifying Performances

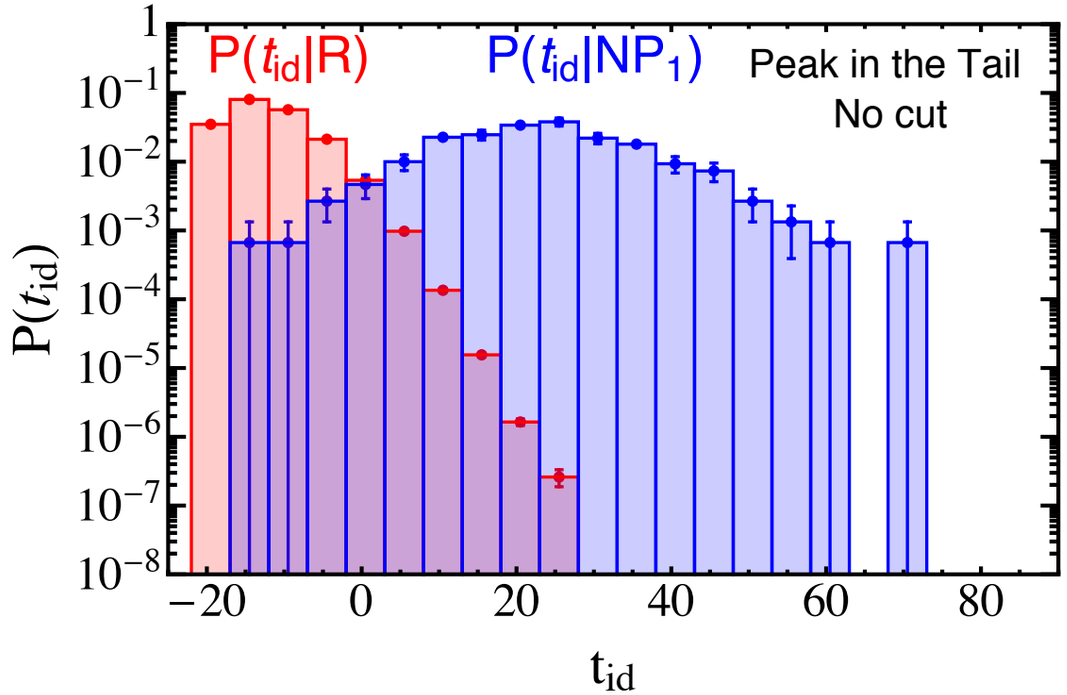
[D'Agnolo, AW, arXiv:1806.02350]

Our Z-score vs **Ideal** Z-score

Run over R and NP toys [repeat train.]
Compute NP p-value distribution



How difficult is to see "NP"?
Compute ideal p-value [Ney.-Pea.]

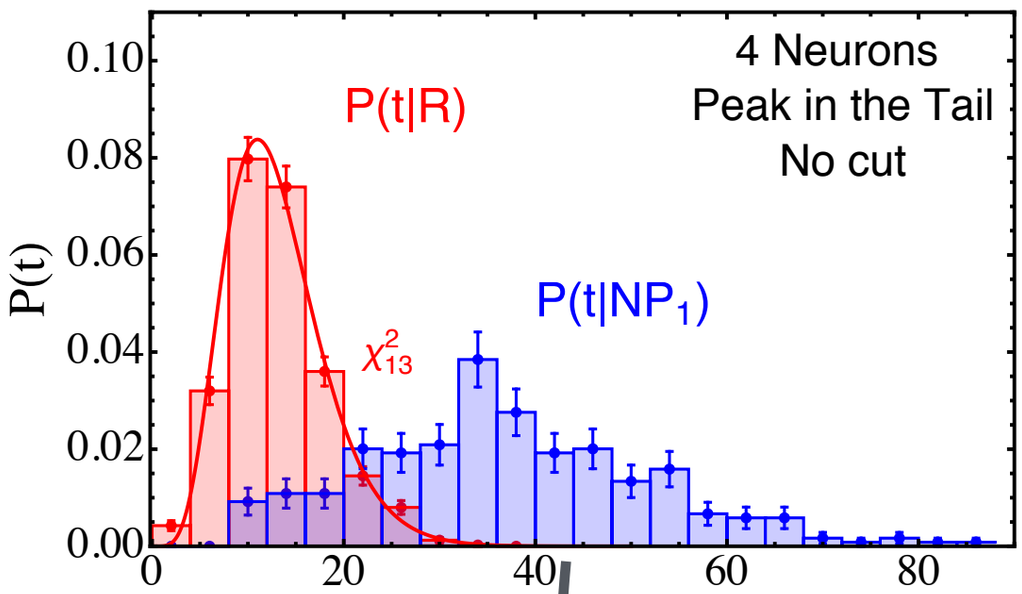


Quantifying Performances

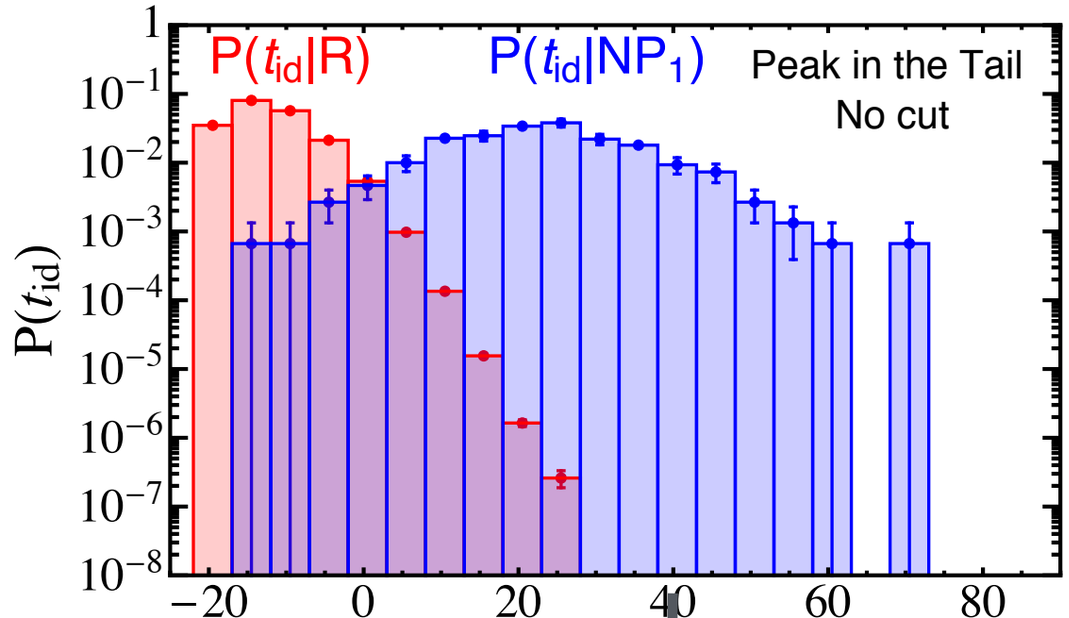
[D'Agnolo, AW, arXiv:1806.02350]

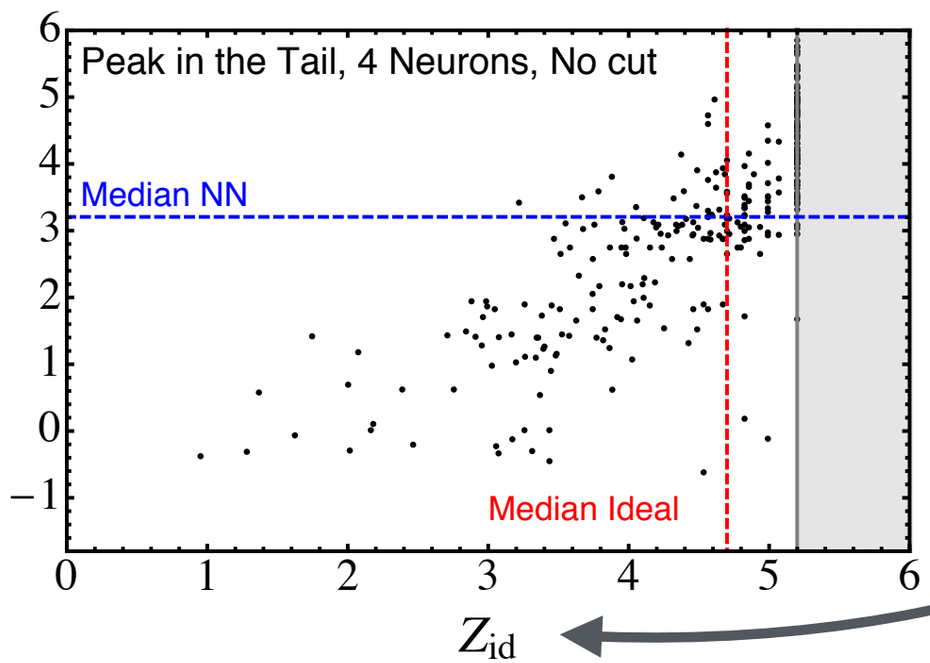
Our Z-score vs **Ideal** Z-score

Run over R and NP toys [repeat train.]
Compute NP p-value distribution



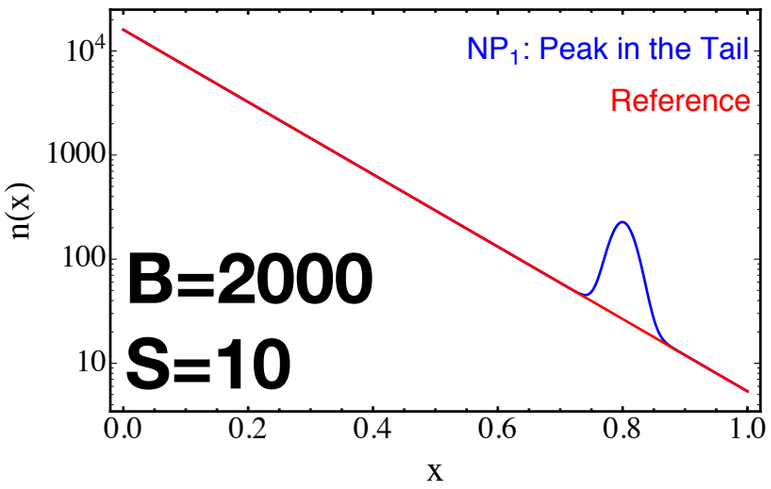
How difficult is to see "NP"?
Compute ideal p-value [Ney.-Pea.]

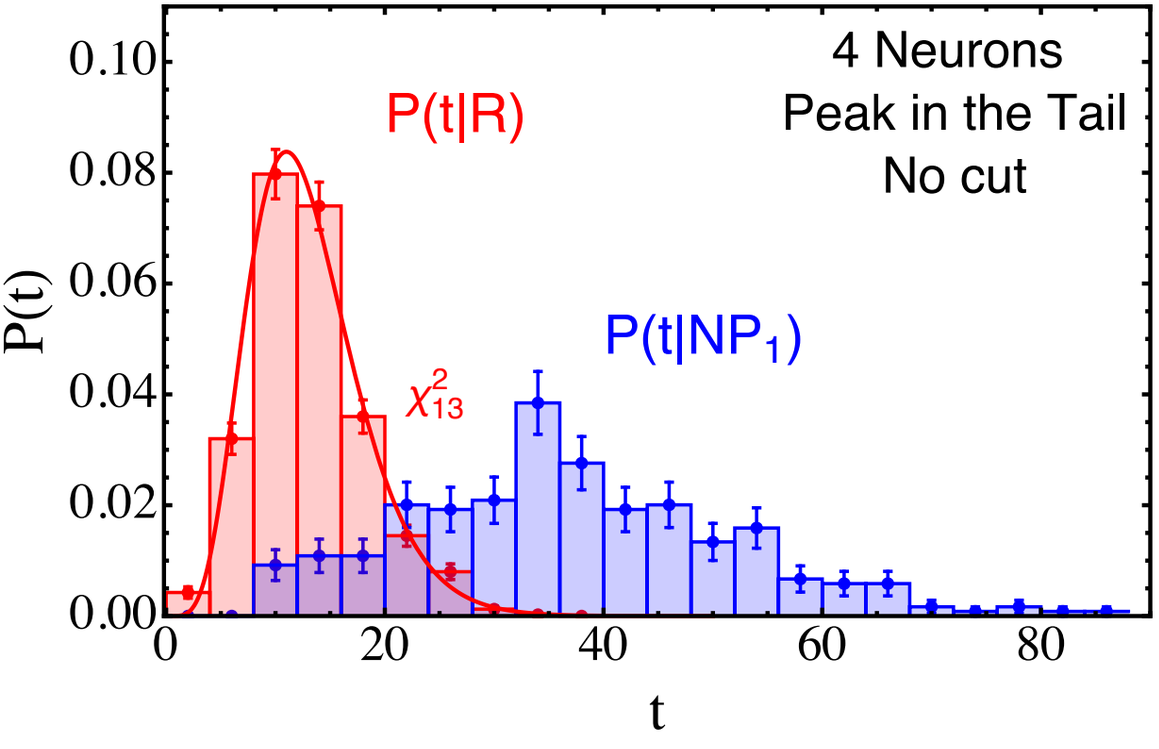


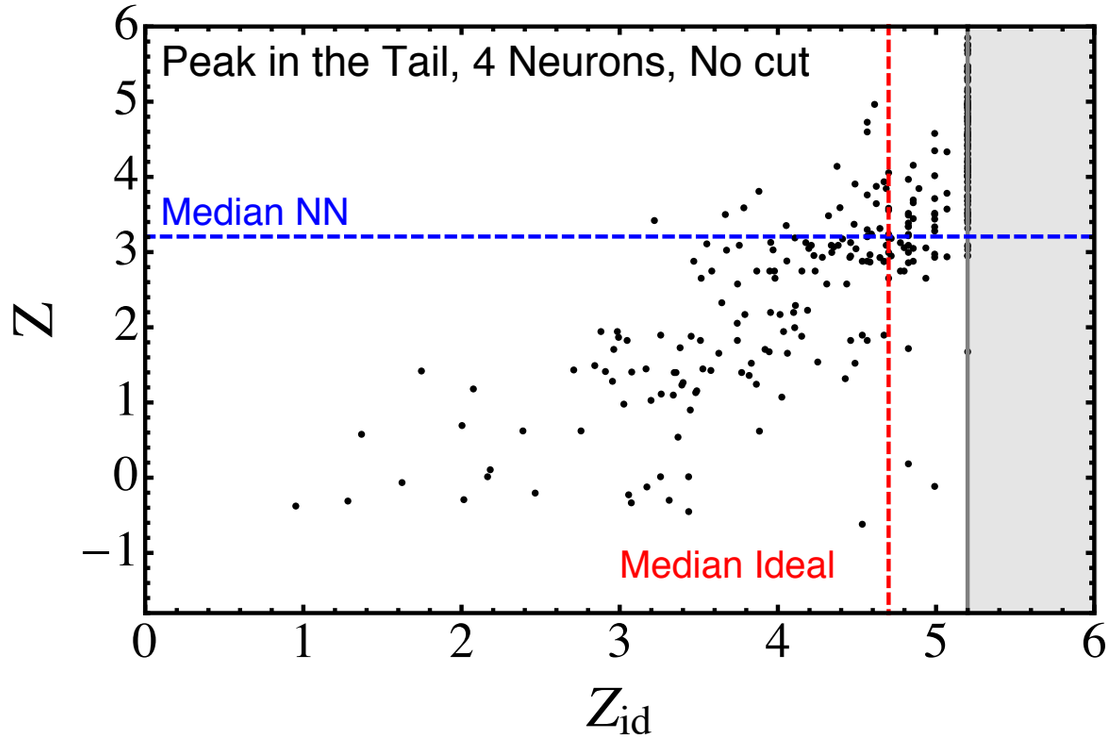


Quantifying Performances: NP_1

[D'Agnolo, AW, arXiv:1806.02350]

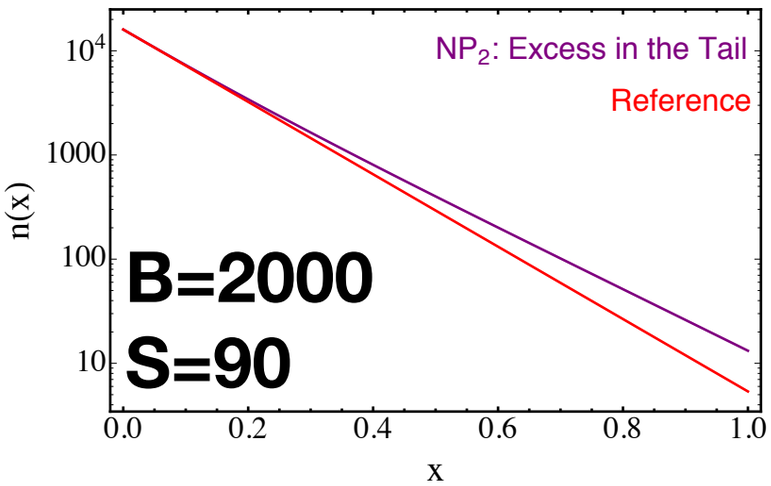


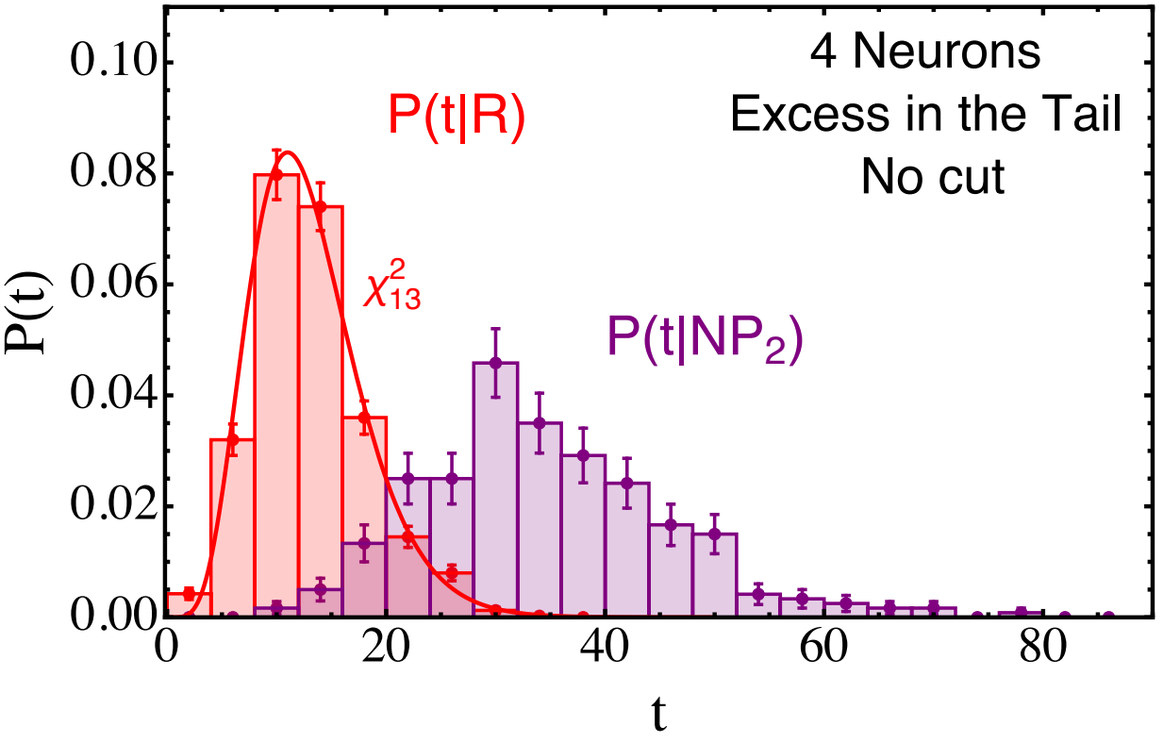


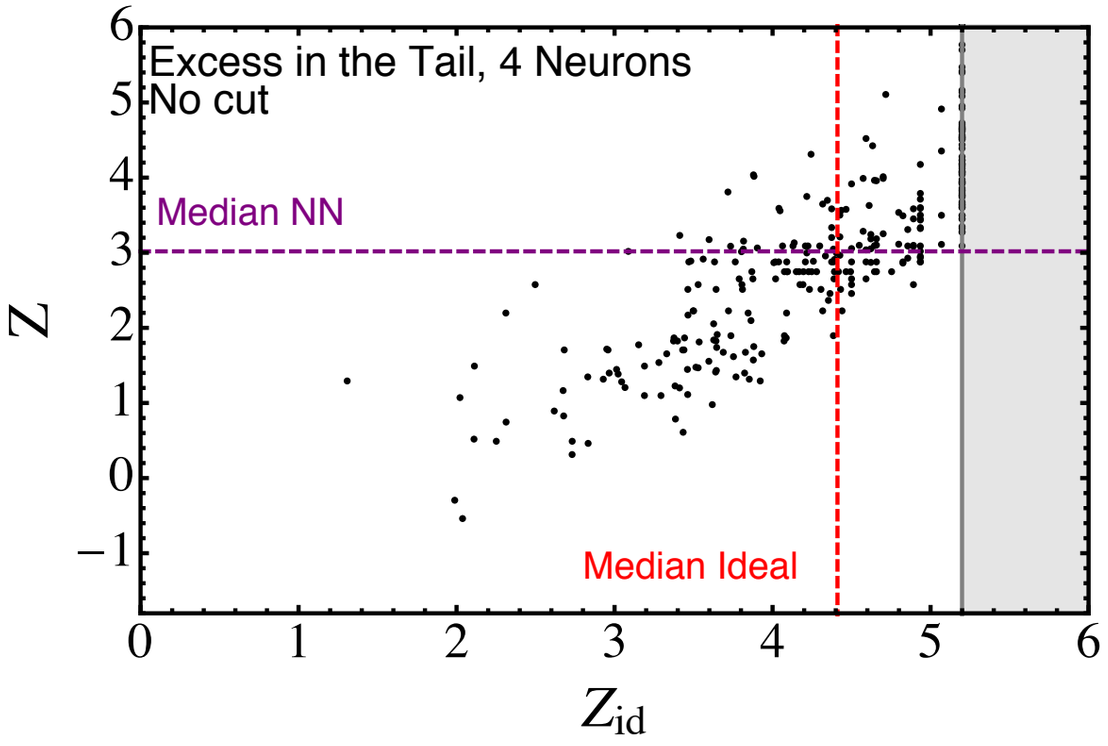


Quantifying Performances: NP_2

[D'Agnolo, AW, arXiv:1806.02350]

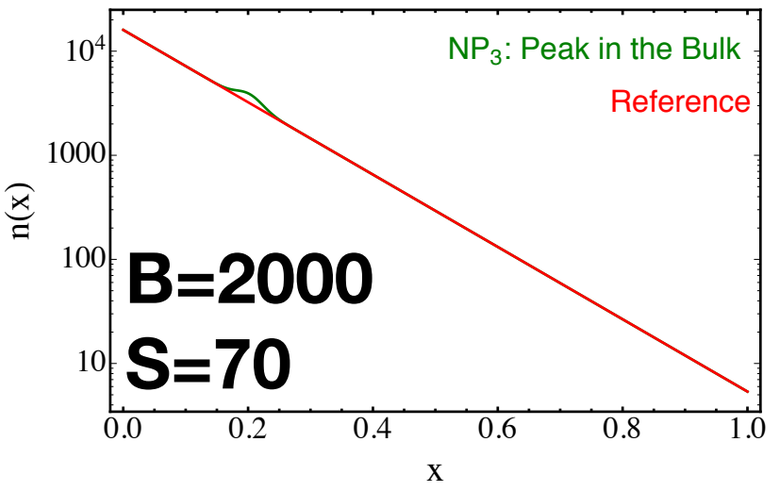


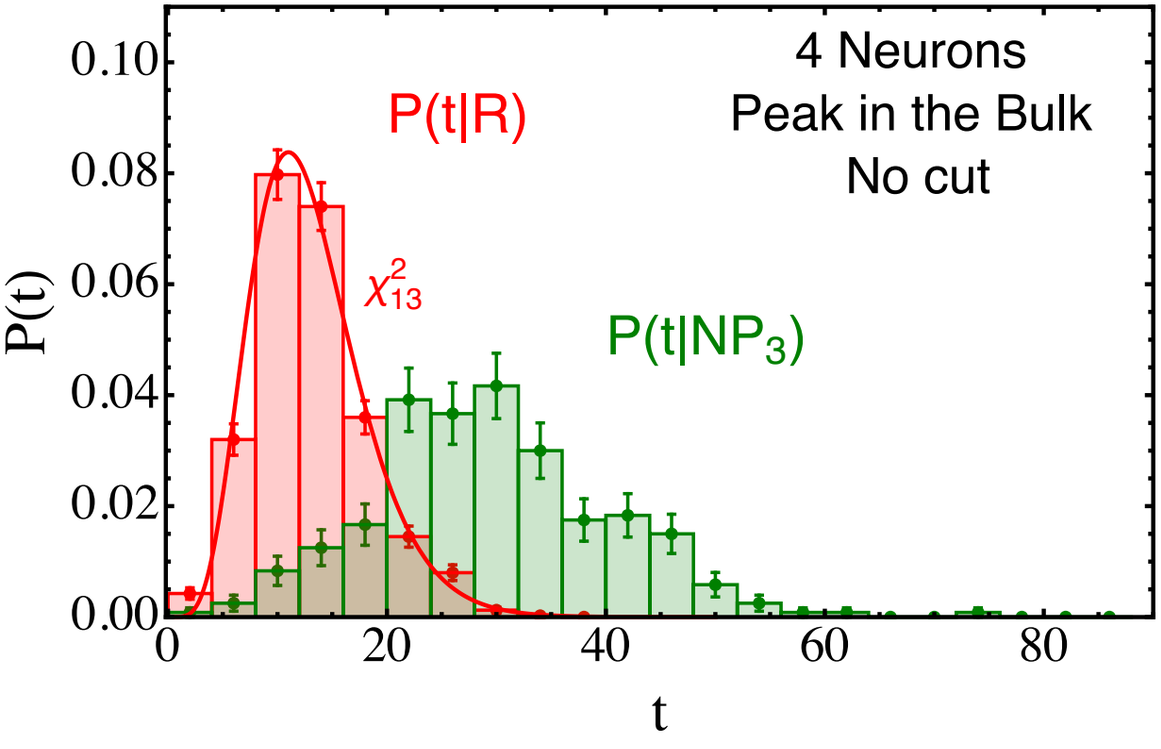


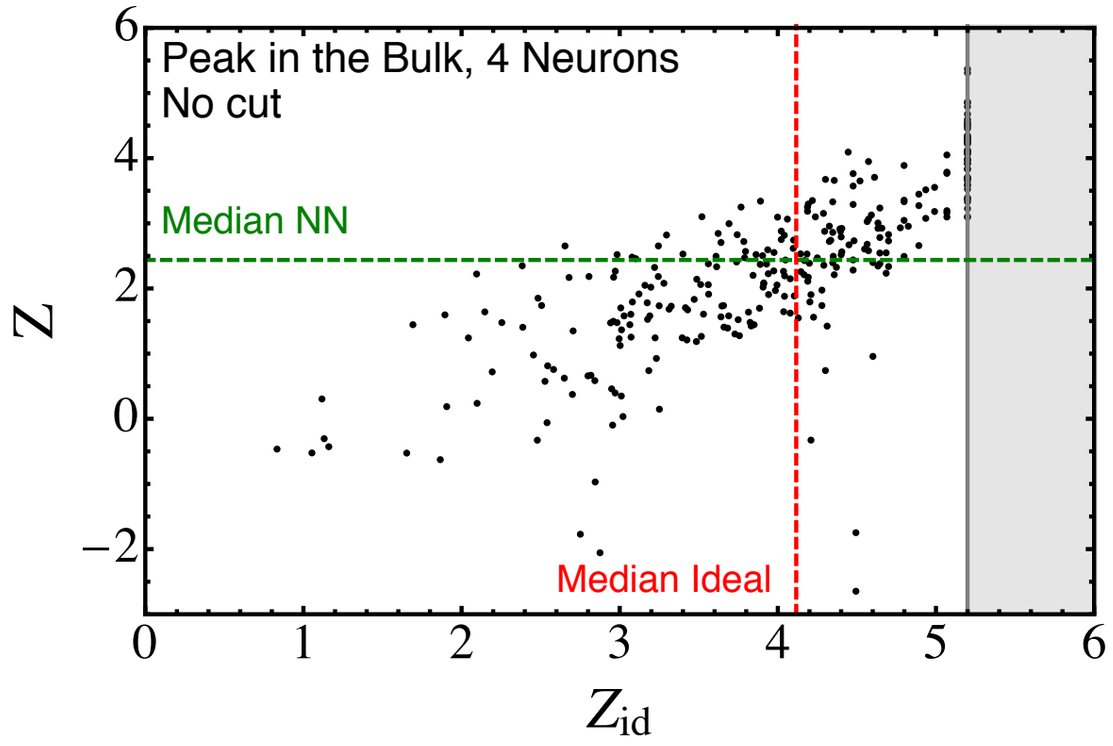


Quantifying Performances: NP_3

[D'Agnolo, AW, arXiv:1806.02350]

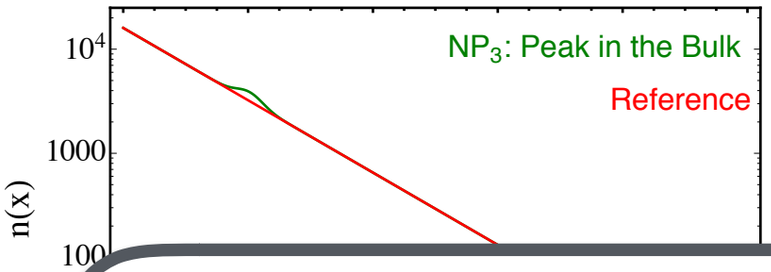






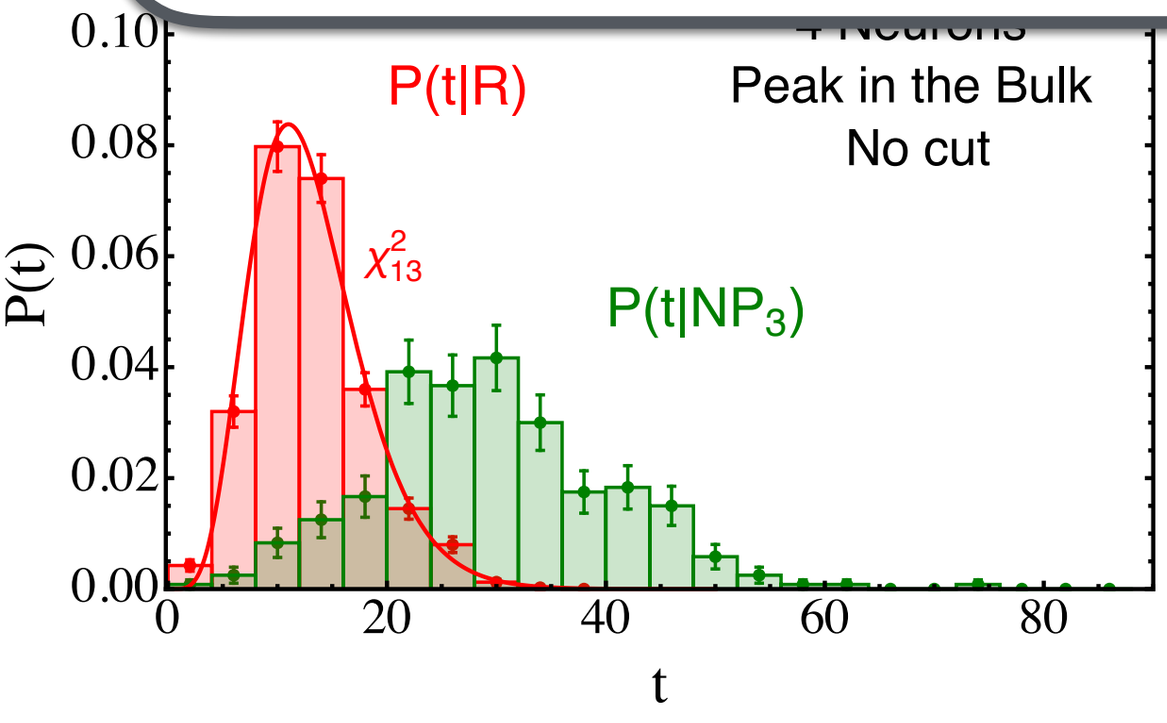
Quantifying Performances: NP_3

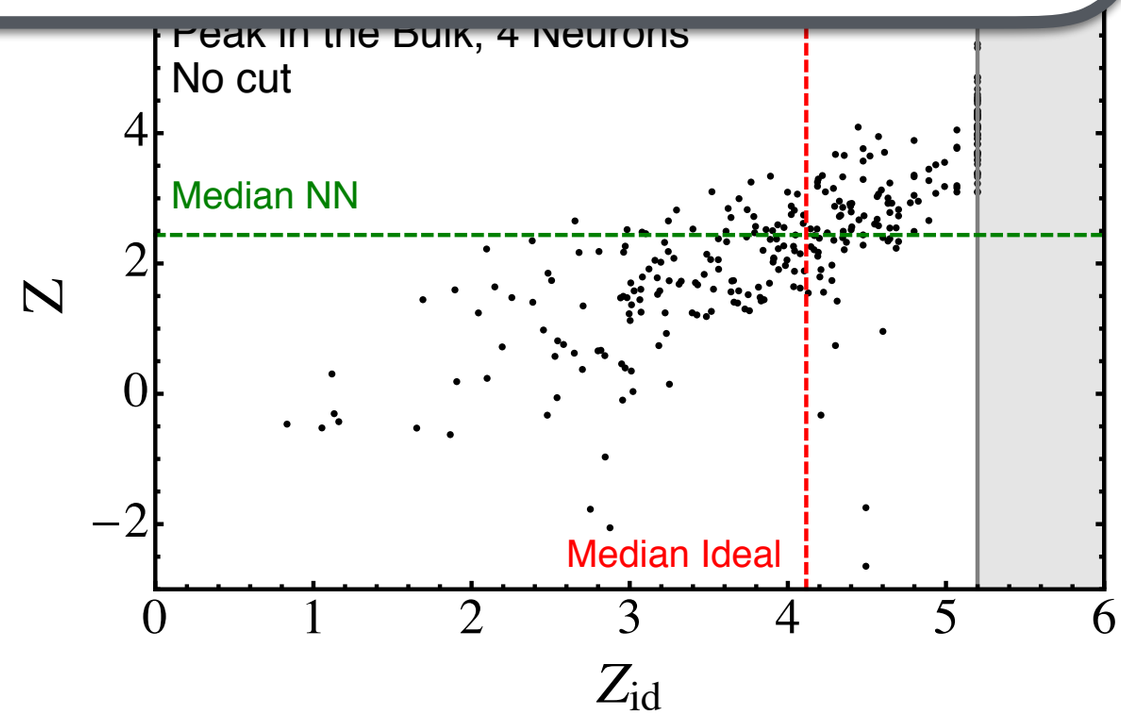
[D'Agnolo, AW, arXiv:1806.02350]



Very Model-Independent !!

[whatever this means …]





A Realistic Problem

[Grosso, D'Agnolo, Pierini, AW, Zanetti in progress]

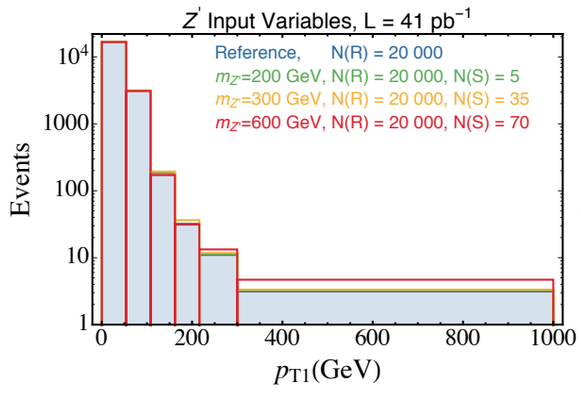
$\mu^+\mu^-$ final state with acceptance cuts [Z-peak in], 5 features

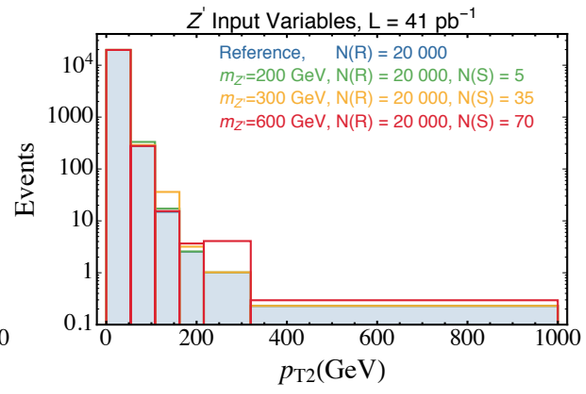
Several tests with N(R) ranging from 10^4 to 10^5. *

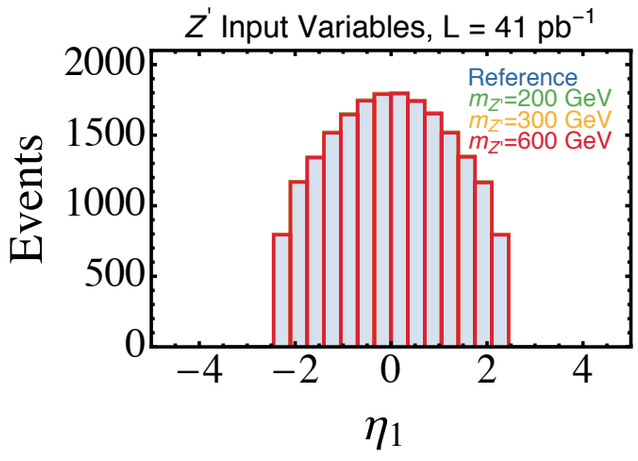
Resonant signal: Z'@200/300/600 GeV.

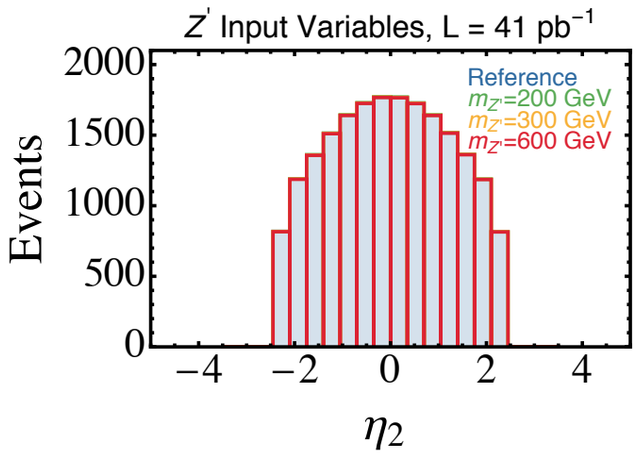
S from 10 to 100

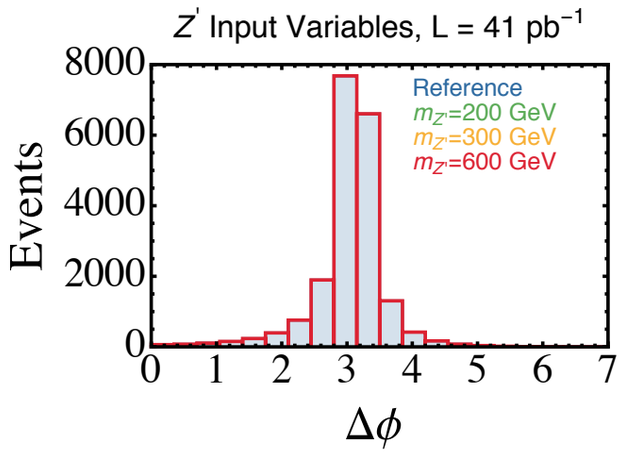
EFT signal: $C_{2W} \sim 1/\text{TeV}^2$











Benchmark: deep 5-5-5-5-1, 3 10^5 epochs

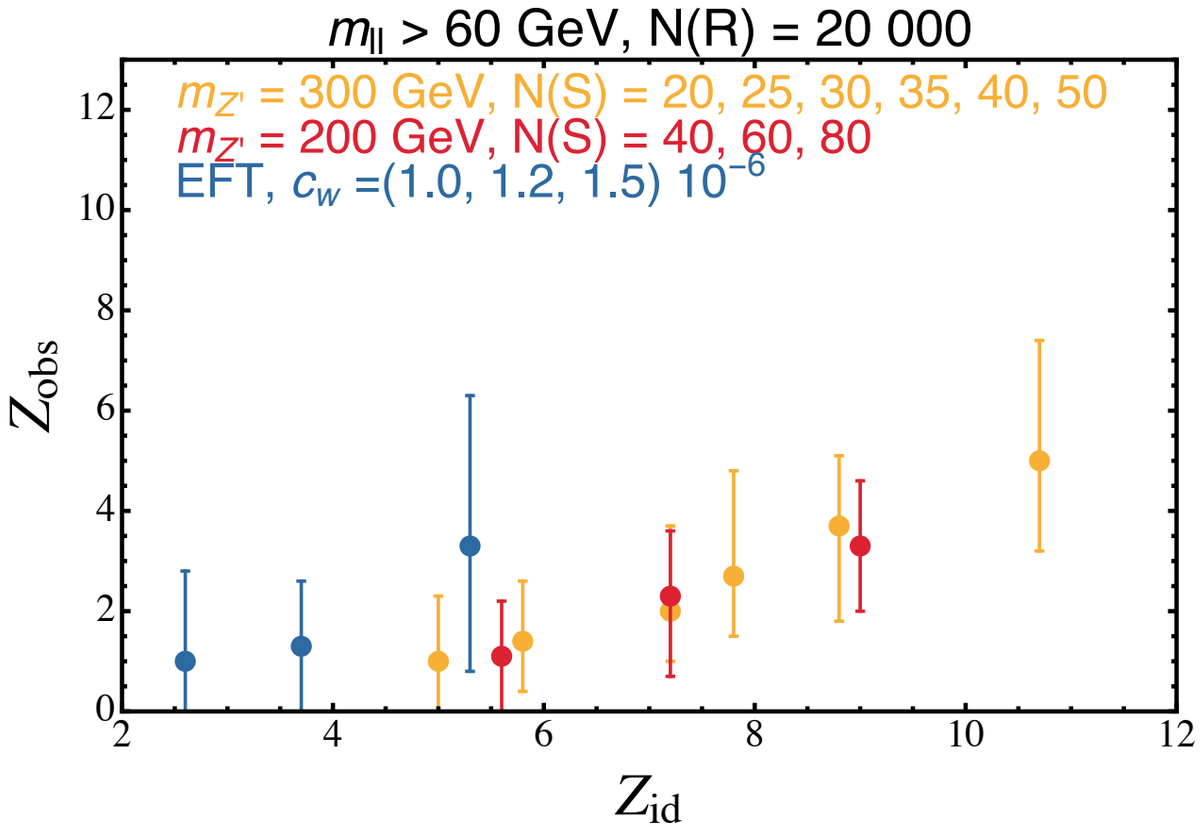
* LHC will produce around 1.5 10^8 !

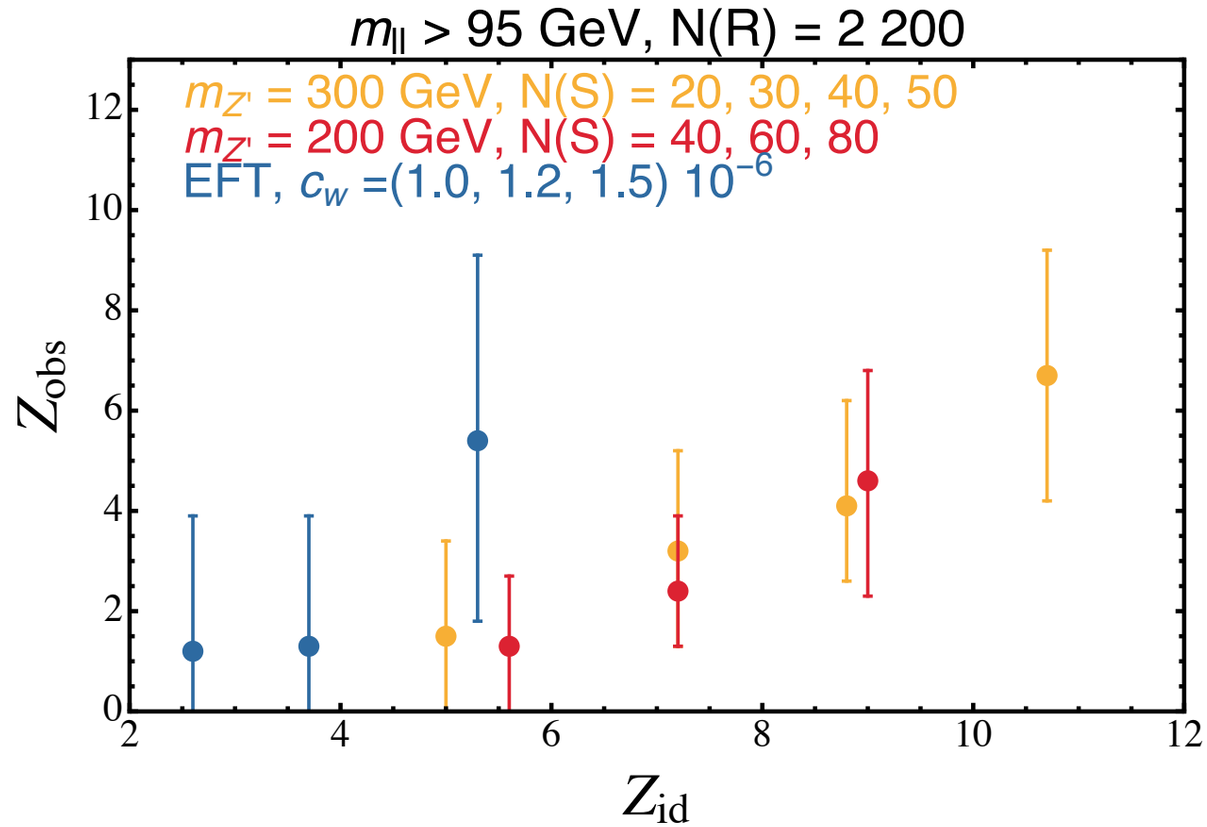
A Realistic Problem

[Grosso, D'Agnolo, Pierini, AW, Zanetti in progress]

$\mu^+\mu^-$ final state with acceptance cuts [Z-peak in], 5 features

Several tests with N(R) ranging from 10^4 to 10^5. *

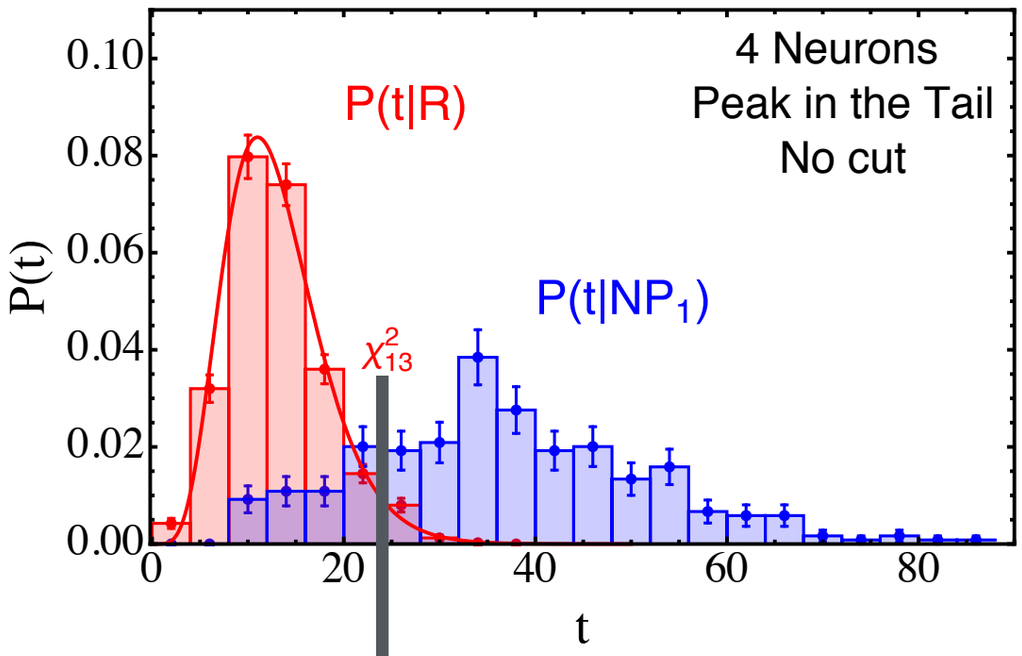




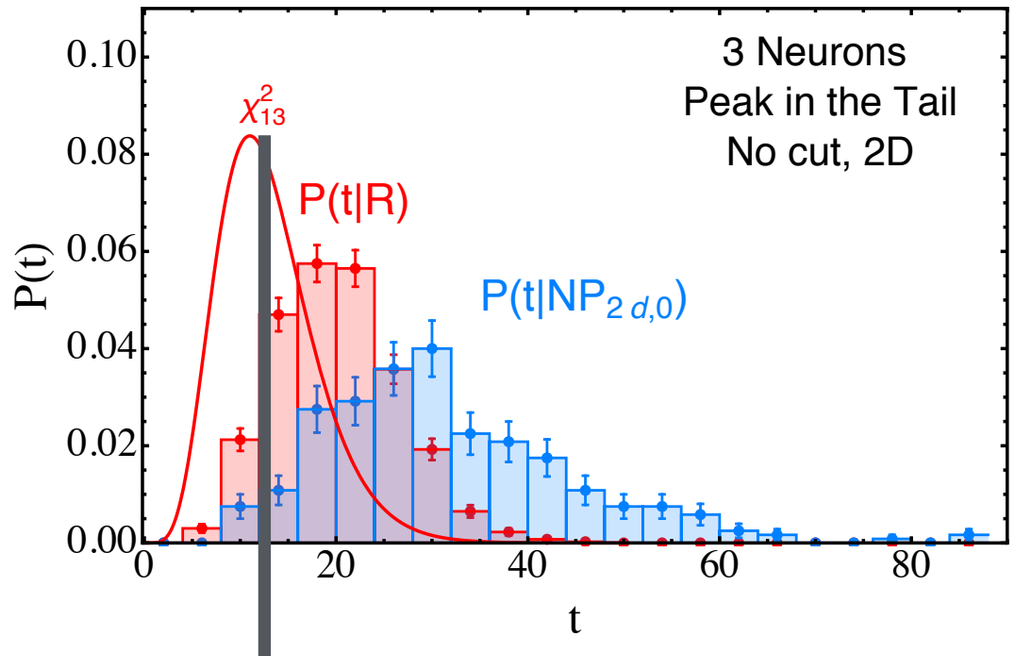
Good performances also in 5 features problem.

Weight Clipping Selection

[Grosso, D'Agnolo, Pierini, AW, Zanetti in progress]



Agreement with χ^2 dof=#NNpar.
Expected for ML in As.Lim.



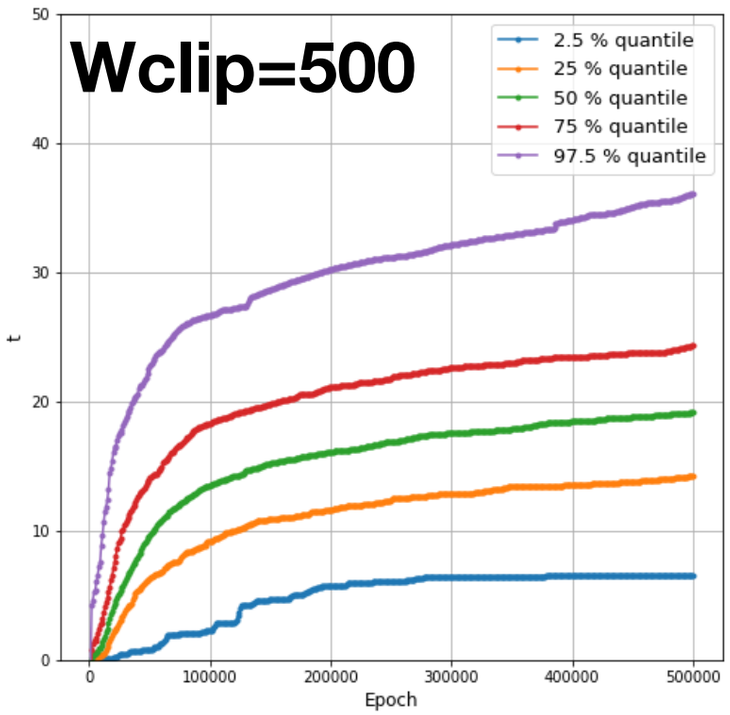
Agreement not
automatically there

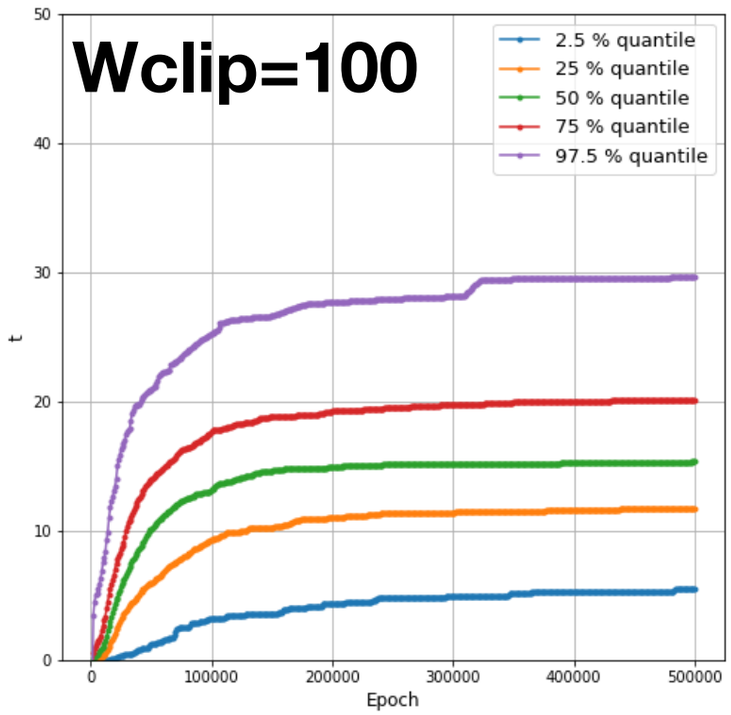
Probably due to low-statistic features being overfitted.

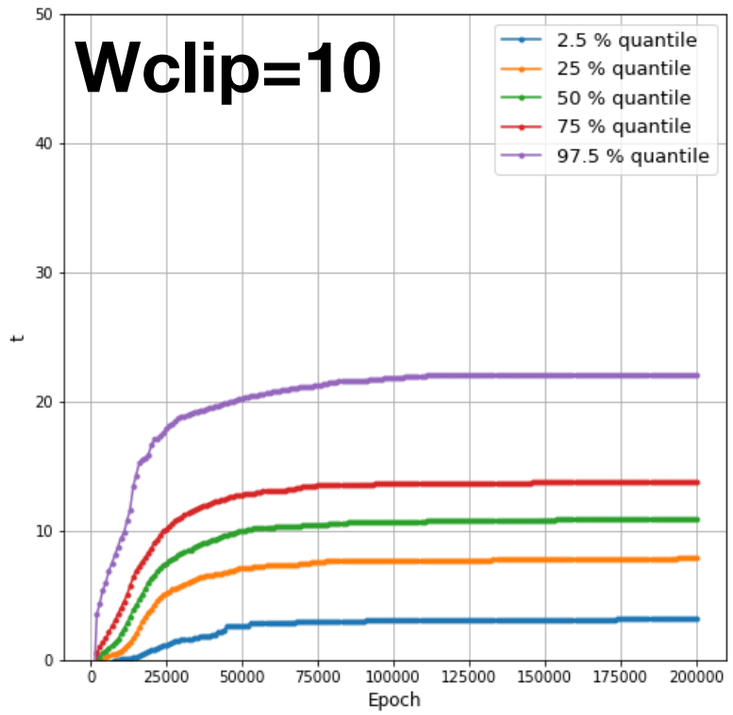
Compatibility with χ^2 used to set Weight Clipping.

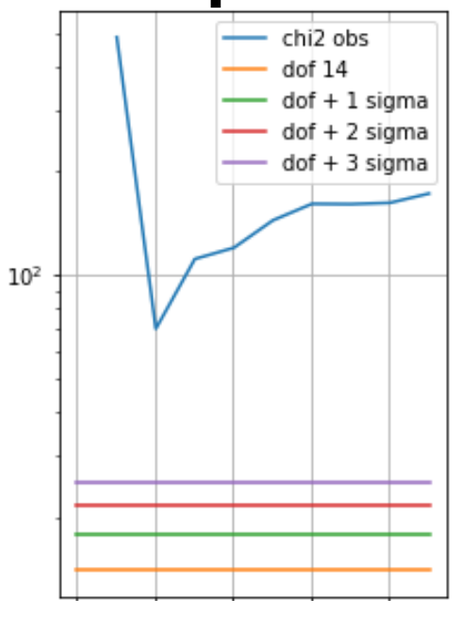
Weight Clipping Selection

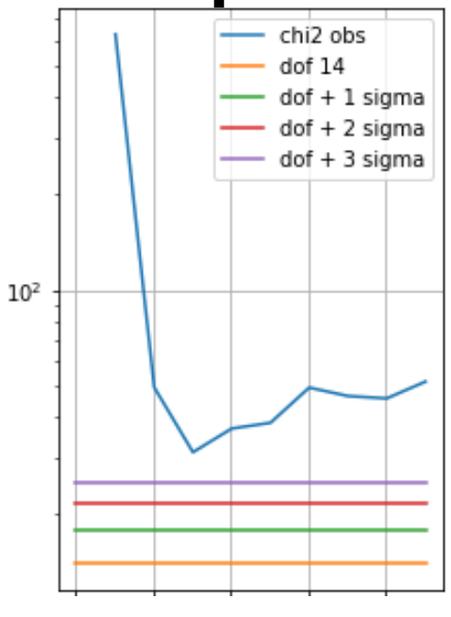
[Grosso, D'Agnolo, Pierini, AW, Zanetti in progress]

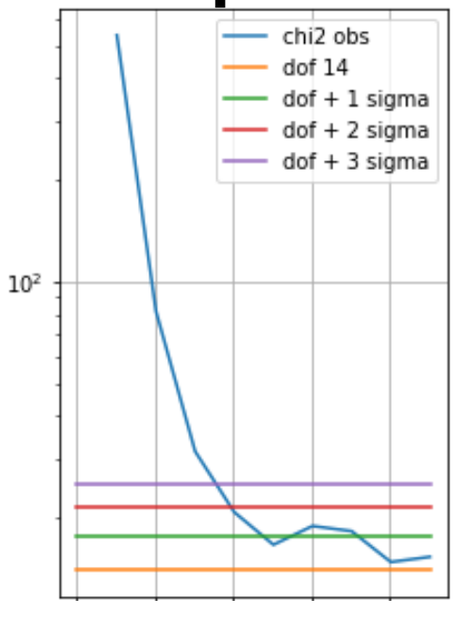


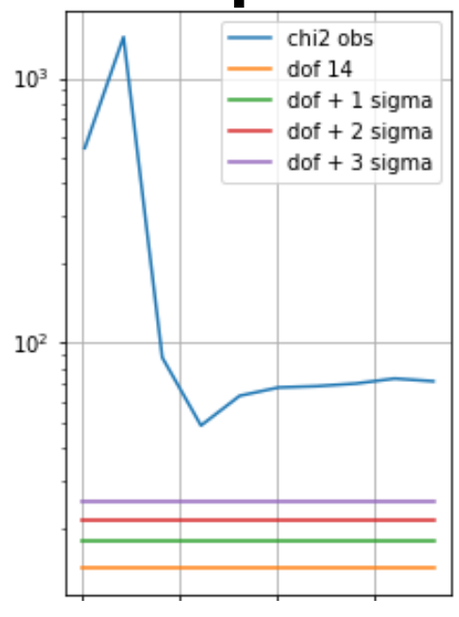












Other Approaches

CWoLa Hunting: **[Collins, Howe, Nachman: arXiv:1805.02664]**

Data/Reference regions selected by mass-window (like BumpHunter)

NN learns Data/Reference distribution ratio of additional variables

Ratio provides additional discriminant and improves BumpHunter reach

Novelty Detection: **[Hajer et al.: arXiv:1807.10261; Pierini et al., in progress]**

Slightly different: we don't necessarily care of "rare" SM events

Non-QCD jets: **[Aguilar–Saavedra et al.: arXiv:1709.01087,Heimel et al.: arXiv:1808.08979]**

NN for Dimensionality Reduction: **[Weisser, Williams: arXiv:1612.07186]**

Train Data/Reference to get classifier, use it for univariate χ^2 or KS test

Gaussian Mixture pdf: **[Kuusela et al.: arXiv:1112.3329]**

Use Gaussian Mixture pdf estimate for Data and for Reference

Nearest-Neighbours pdf: **[De Simone, Jacques: arXiv:1807.06038]**

Use Nearest-Neighbours pdf estimate for Data and for Reference

Few More Comments

Model-Independent search algorithms also good for:

- Comparison between different Monte Carlo Generators
- Data Validation

Few More Comments

Model-Independent search algorithms also good for:

- Comparison between different Monte Carlo Generators
- Data Validation

When and if these techniques make it to real analyses,
I suspect we will find plenty of wrong Monte Carlos …

Few More Comments

Model-Independent search algorithms also good for:

- Comparison between different Monte Carlo Generators
- Data Validation

When and if these techniques make it to real analyses,
I suspect we will find plenty of wrong Monte Carlos …

But maybe we will find New Physics as well !!

Few More Comments

Model-Independent search algorithms also good for:

- Comparison between different Monte Carlo Generators
- Data Validation

When and if these techniques make it to real analyses,
I suspect we will find plenty of wrong Monte Carlos …

But maybe we will find New Physics as well !!

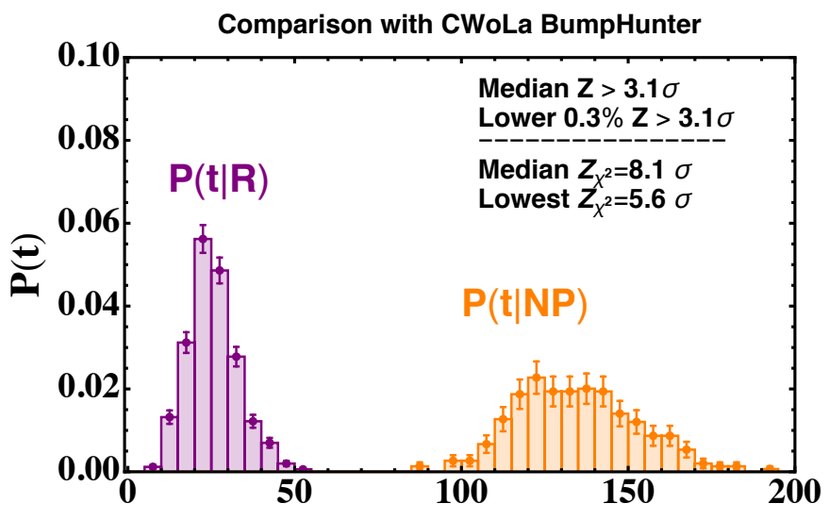
Thank You

Backup

Some Comparison

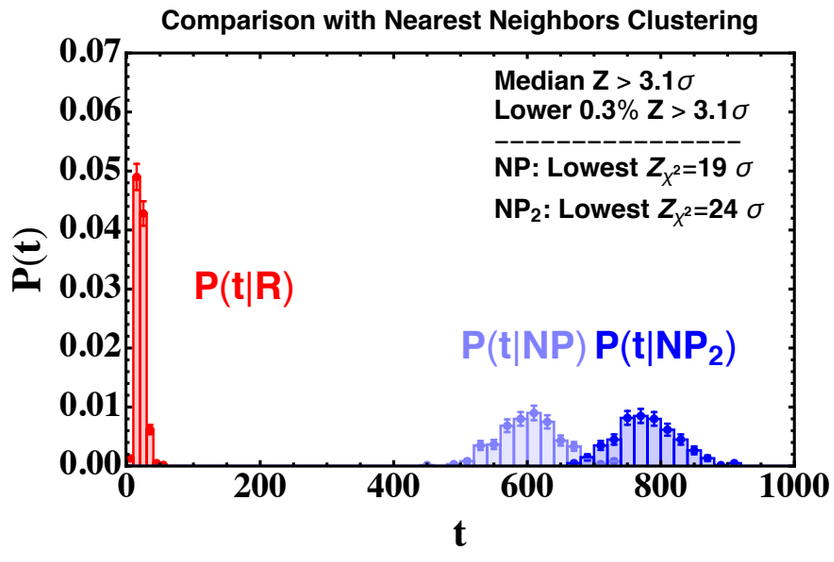
CWoLa Hunting: [Collins, Howe, Nachman: arXiv:1805.02664]

CWoLa is not model-independent.
In spite of using model-specific information, does not perform better



Nearest-Neighbours pdf: [De Simone, Jacques: arXiv:1807.06038]

No comparison



Systematic Uncertainties

Crucial observation:

$$\lambda_p(\theta = \theta_{\mathrm{R}} = 0) = \frac{L(0, \widehat{\widehat{\nu}}(0))}{L(\widehat{\theta}, \widehat{\nu})} \quad = \quad \frac{\underset{\nu}{\mathrm{Max}}[L(\mathrm{R}, \nu)]}{\underset{\mathbf{w}}{\mathrm{Max}}[L(\mathbf{w})]}$$

Unlike in model-dependent searches, no need to distinguish “parameters of interest θ” from “nuisance ν” in profile lik. denominator.

Recipe is train NN against best-fit R distribution

But we did not yet try in practice.