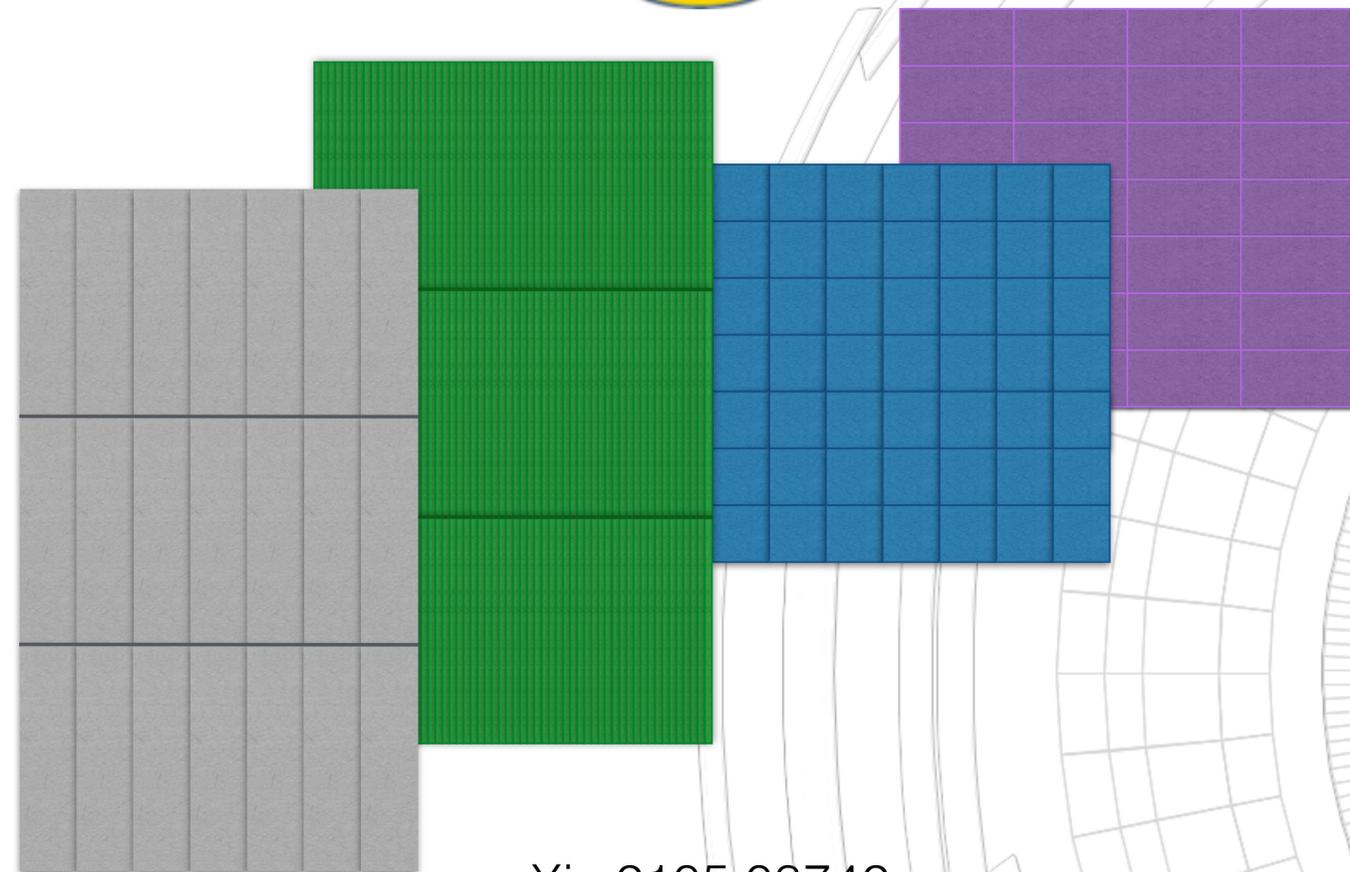


Uncertainty Aware Learning For High Energy Physics

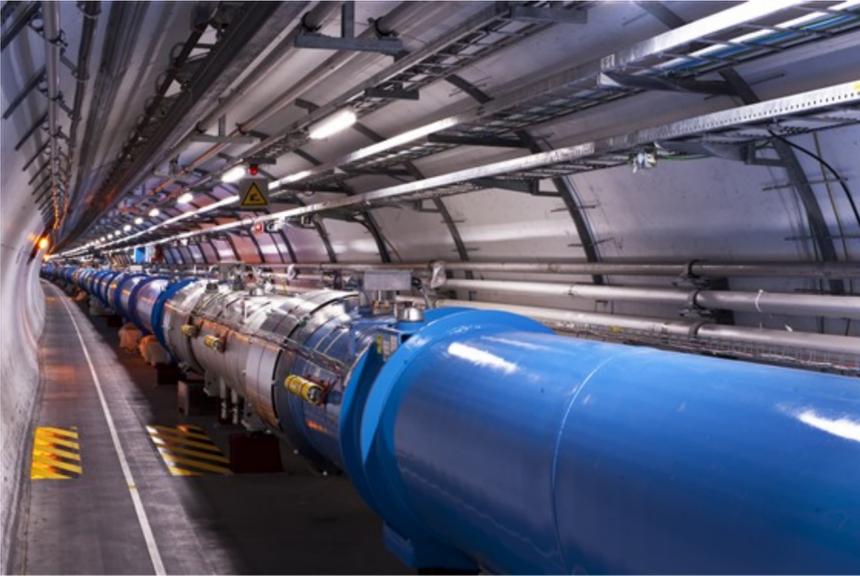
Aishik Ghosh, Benjamin Nachman, Daniel Whiteson

ML4Jets
08 July 2021

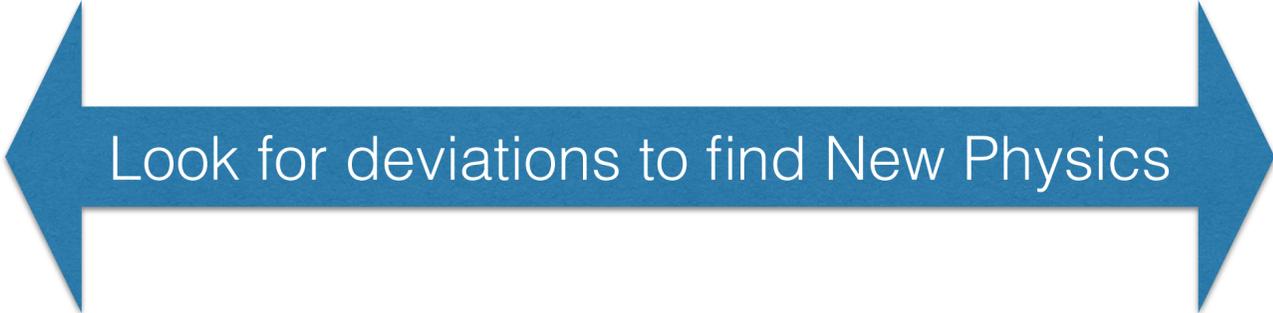


[arXiv:2105.08742](https://arxiv.org/abs/2105.08742)

Simulation-Based Inference



Real Data from LHC



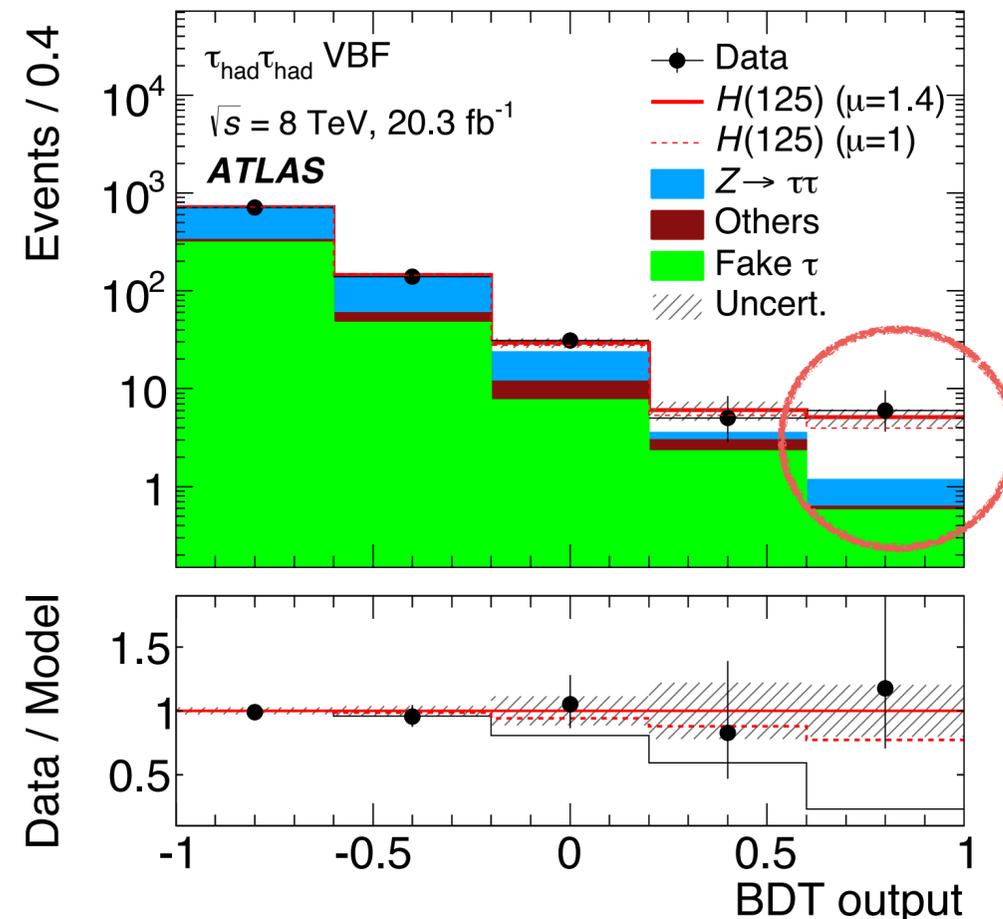
Simulation using Standard Model of Particle Physics

Typical use of ML

ML classifier trained for :

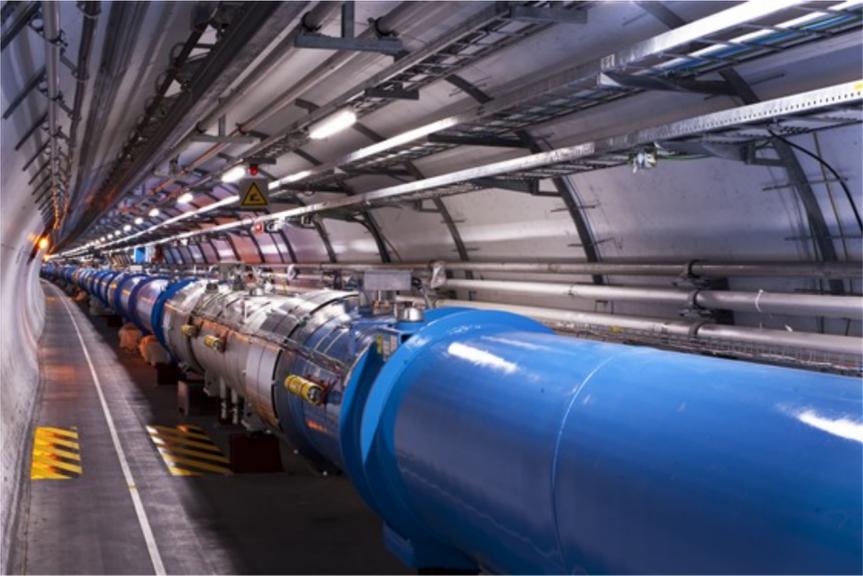
- Increase sensitivity to “signal” (example Higgs Bosons) events which give information about the theory parameters
- Reject “background” events

Output of classifier used as an “optimal” observable to measure theory parameters using a maximum likelihood fit over several bins of histogram



Compare various simulations to data to find best fit

Simulation-Based Inference - Systematic Uncertainties



Real Data from LHC



Simulation using Standard Model of Particle Physics

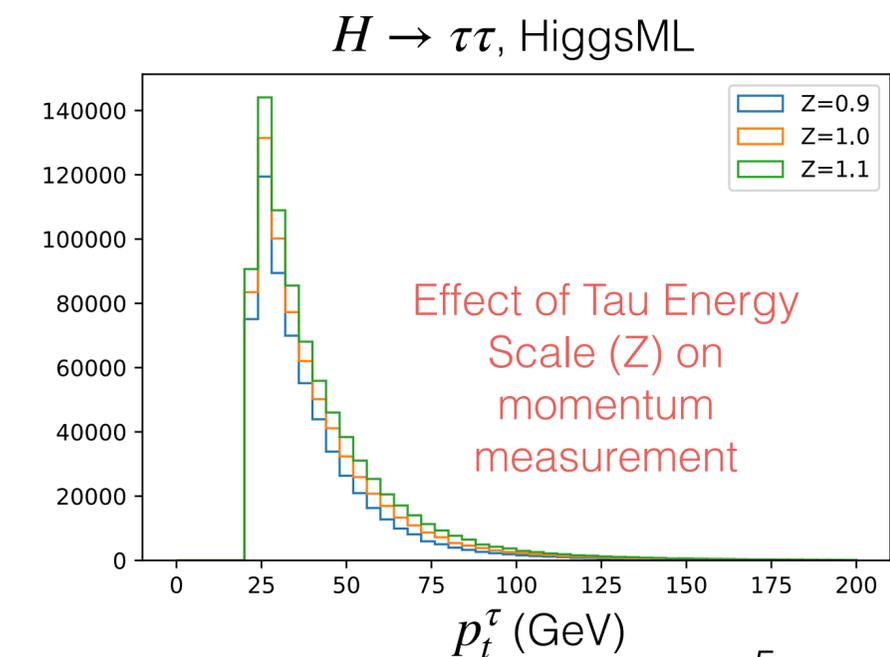
But simulation isn't perfect, **known sources of differences between simulation and data...**

Systematic Uncertainties

- Statistical uncertainties shrink as you take more data, systematic uncertainties usually don't
 - Imagine a metal ruler calibrated at room temperature but used at near 0 K
 - HEP example: Jet Energy Scale
 - Systematics may dominate over statistical uncertainties when a lot of data is available
- We can measure known unknown in the “control region”
 - Final measurement includes a fit on “parameters of interest” as well as these unknown “nuisance parameters”
 - The auxiliary measurement used as a prior on systematic (Z) or fit done simultaneously



Image: <https://www.shutterstock.com/image-photo/measuring-stick-snow-ruler-shows-amount-1896983614>

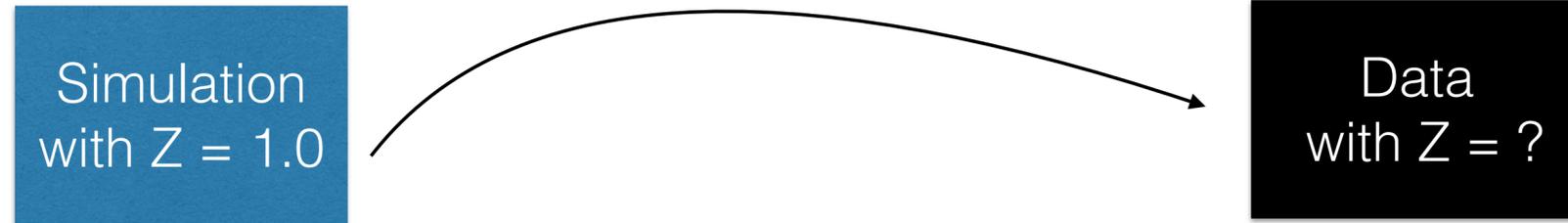


Nominal Classifier and Data Augmentation

- Baseline solution has been to train a classifier on nominal data ($Z=1$) and just account for uncertainties in measurement – which may be large. Full profile likelihood or shift Z and look at impact.

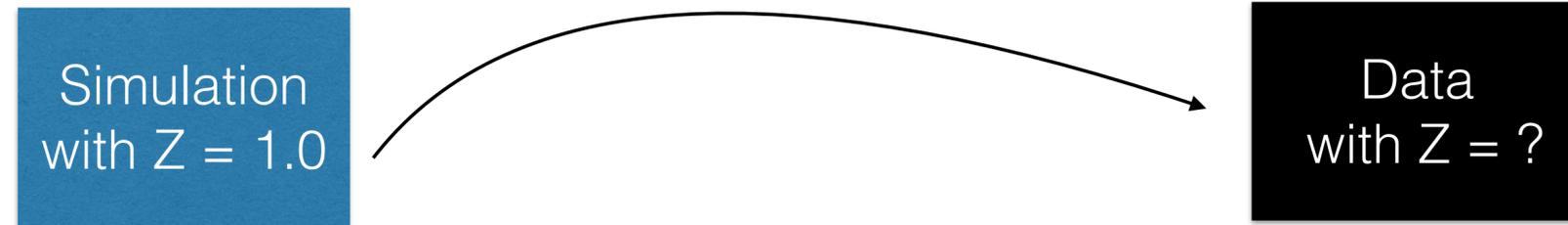
Nominal Classifier and Data Augmentation

- Baseline solution has been to train a classifier on nominal data ($Z=1$) and just account for uncertainties in measurement – which may be large. Full profile likelihood or shift Z and look at impact.



Nominal Classifier and Data Augmentation

- Baseline solution has been to train a classifier on nominal data ($Z=1$) and just account for uncertainties in measurement – which may be large. Full profile likelihood or shift Z and look at impact.

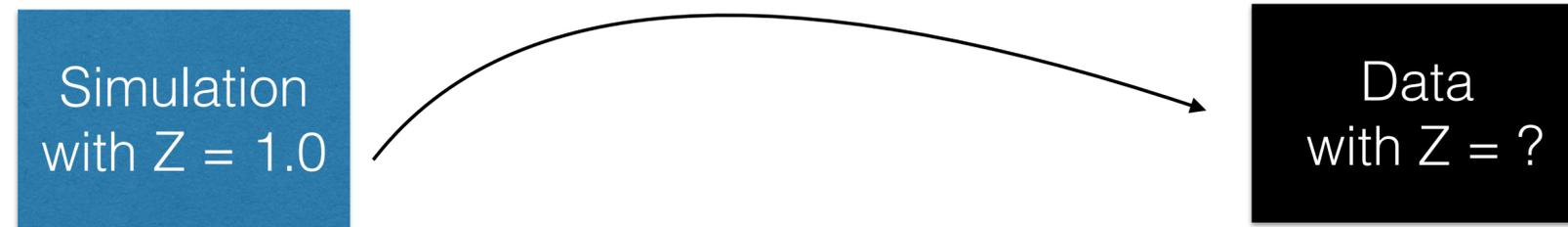


- One way to attack the problem is “Data Augmentation”: Train classifier on simulated data generated with various values of Z , hope that it learns a robust decision function



Nominal Classifier and Data Augmentation

- Baseline solution has been to train a classifier on nominal data ($Z=1$) and just account for uncertainties in measurement – which may be large. Full profile likelihood or shift Z and look at impact.



- One way to attack the problem is “Data Augmentation”: Train classifier on simulated data generated with various values of Z , hope that it learns a robust decision function

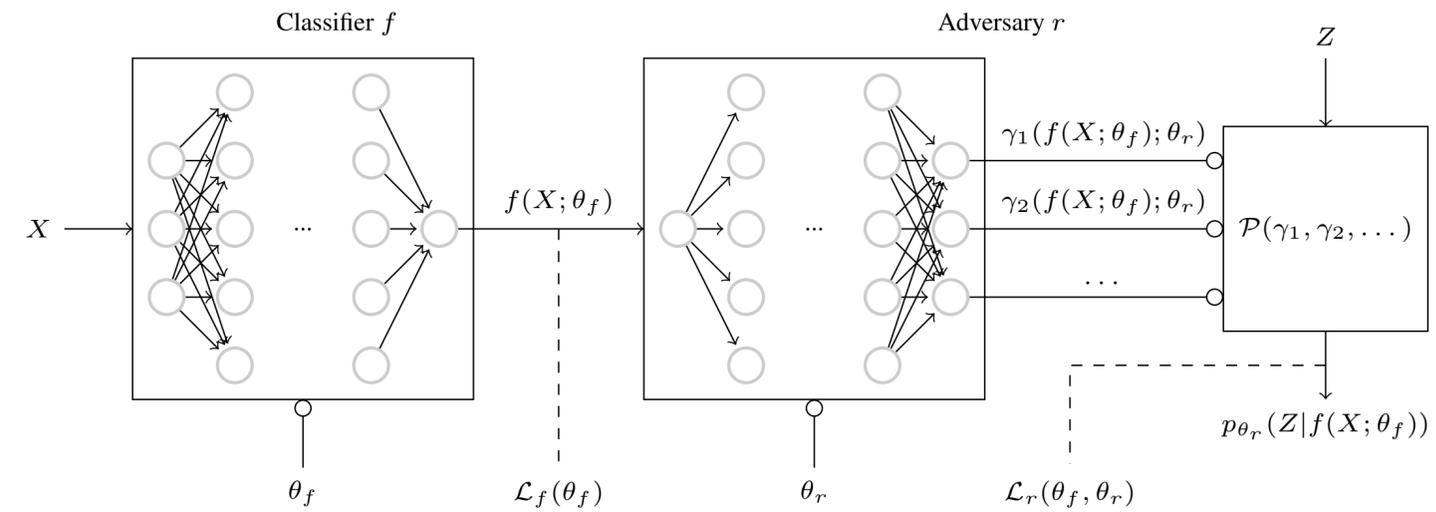


The classifier will learn some general characteristics, but will not be “optimal” for any particular value of Z

“Optimal”: For us means classifier trained at the true value of Z

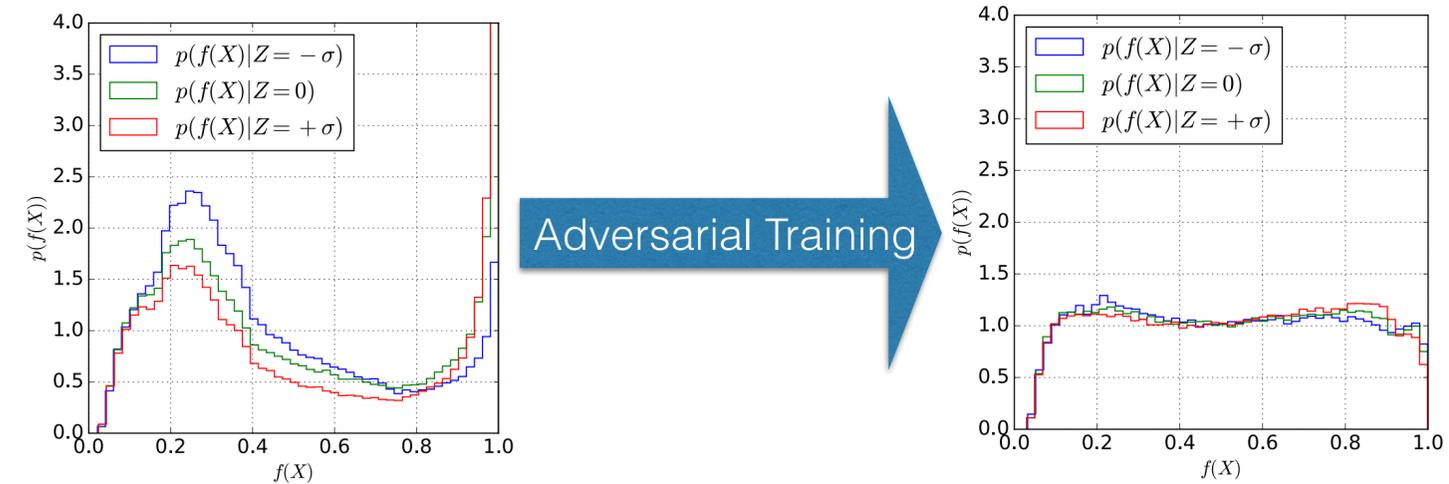
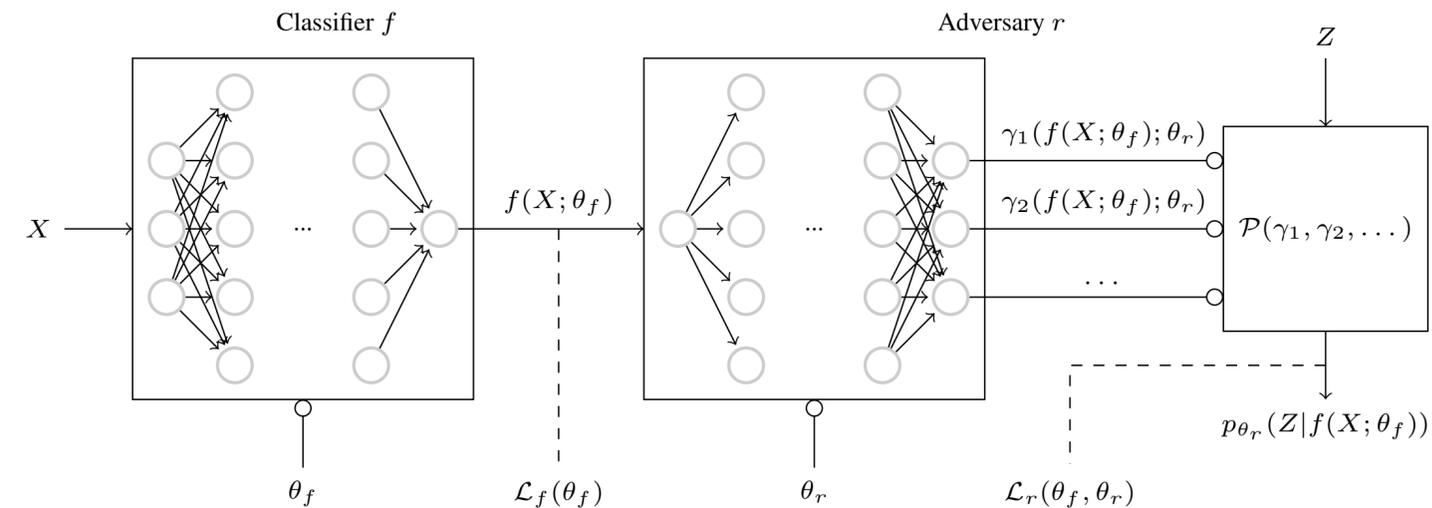
Invariance: Explicitly train for insensitivity to Z

Domain adaptation strategies also used in the past: Eg. Pivot Adversarial Training to make classifier output invariant to systematic (Z)



Invariance: Explicitly train for insensitivity to Z

Domain adaptation strategies also used in the past: Eg. Pivot Adversarial Training to make classifier output invariant to systematic (Z)

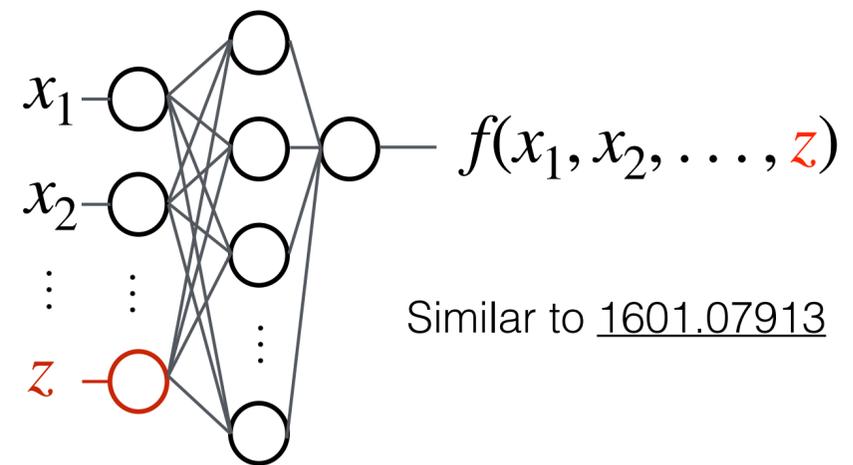


Classifier output for various values of Z

Classifier output becomes invariant to Z , but not necessarily optimal at any particular value of Z , also sometimes not even possible to be invariant

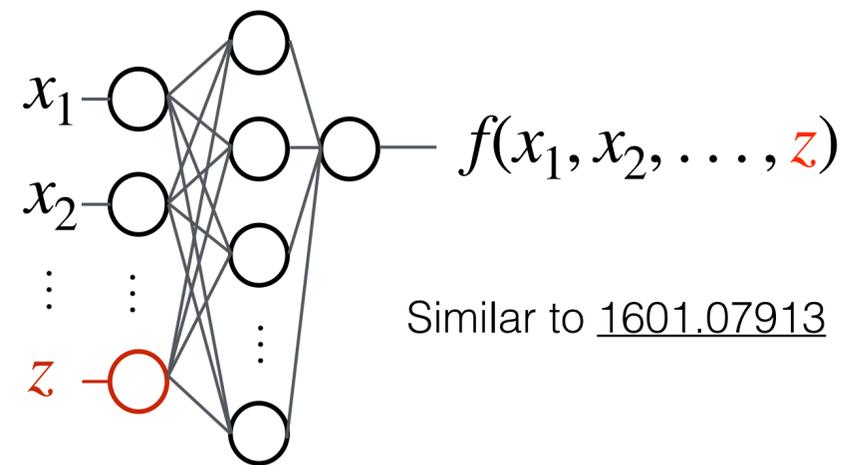
We advocate for the opposite

- Fully parameterise the classifier on Z in a “systematic aware” way



We advocate for the opposite

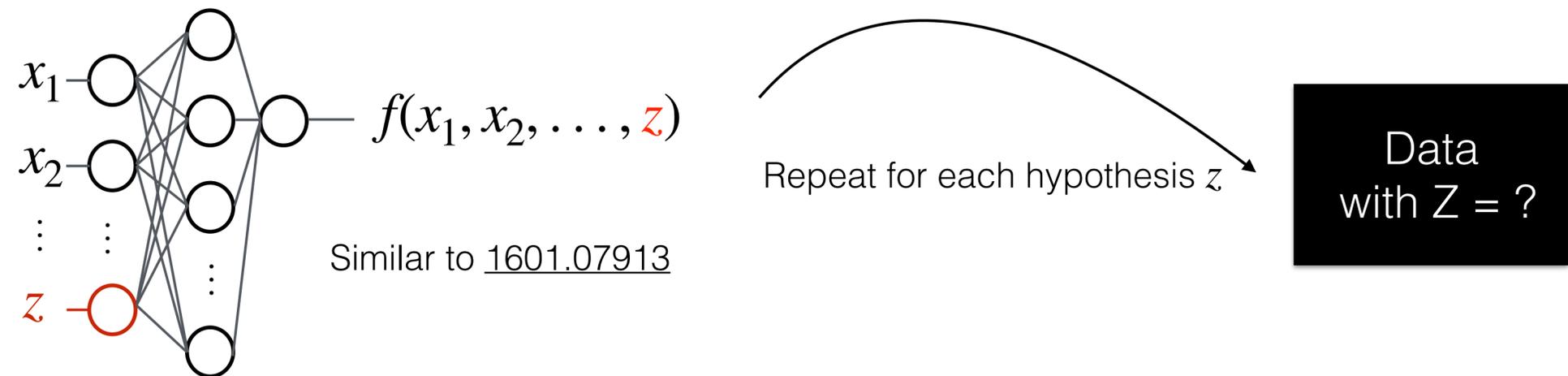
- Fully parameterise the classifier on Z in a “systematic aware” way



- Intuition: Allow the analysis technique to vary with Z
You always get the best classifier for each value of Z

We advocate for the opposite

- Fully parameterise the classifier on Z in a “systematic aware” way



- Intuition: Allow the analysis technique to vary with Z
You always get the best classifier for each value of Z
- Use the parameterised classifier response for final likelihood fit to constrain parameters of interest (POI) and nuisance parameters (NP)

In following slides, POI will be the signal strength parameter ‘ μ ’ and the NP will be denoted ‘ Z ’

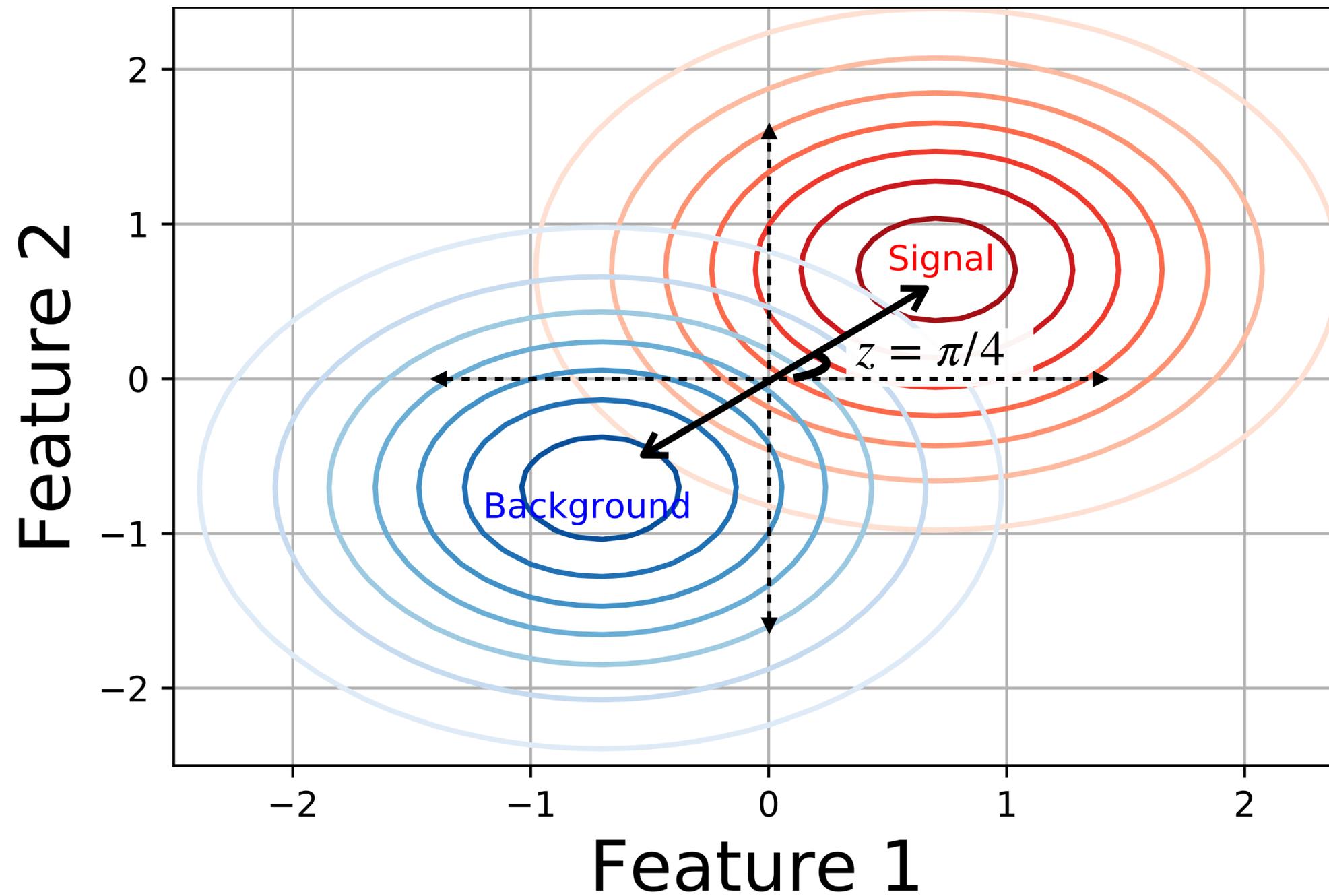
Caveat: When not to use any of these models

Caveat: When not to use any of these models

Two point systematics: Don't lose your only handle on these systematics!

Herwig vs Pythia - If you are invariant to this difference are you sure you are invariant to underlying physics reason for these differences?

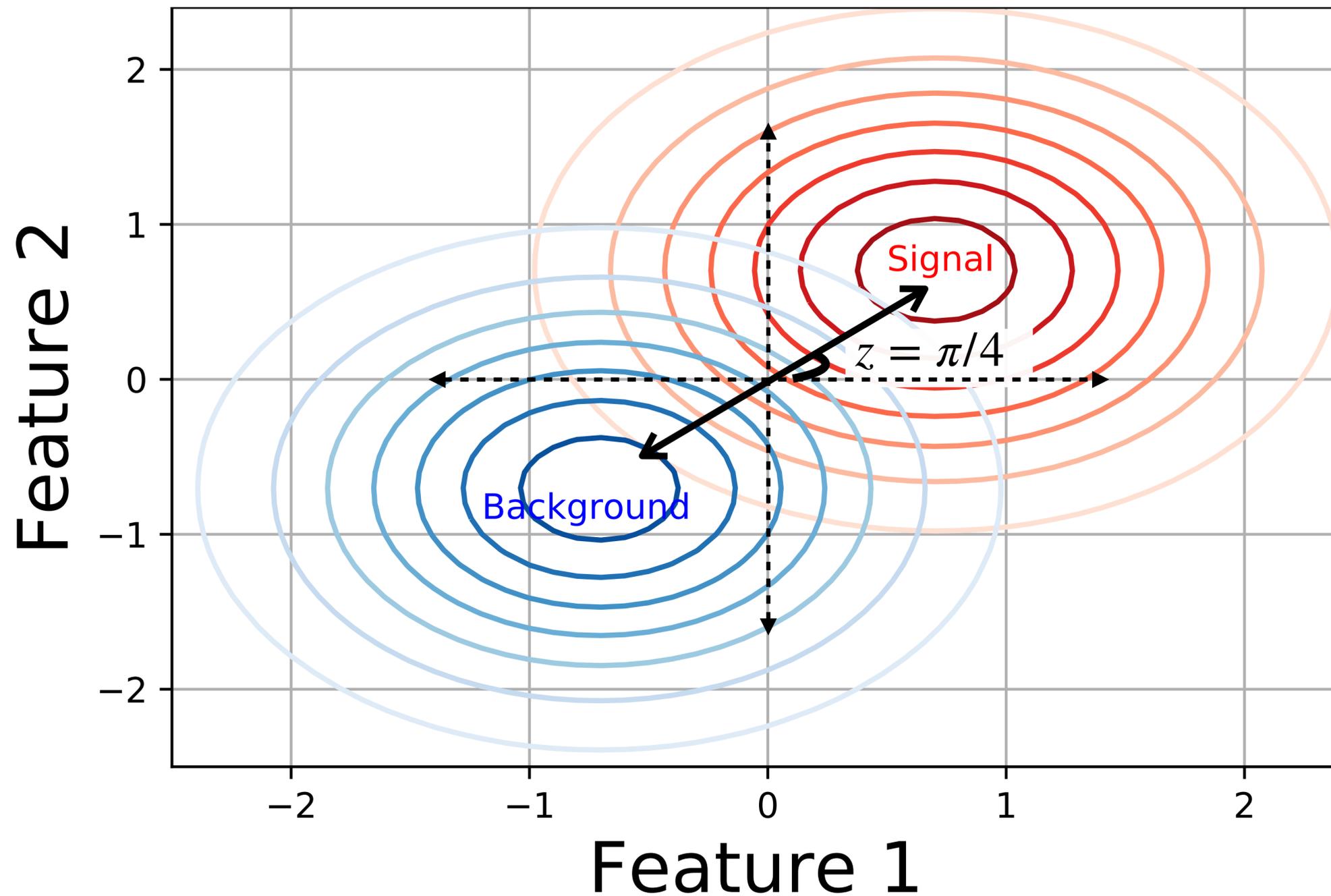
Demonstration on Toy Problem



$$\bar{\mu} = \frac{N_{s,obs}}{N_{s,exp}}$$

$z = \text{Angle}$

Demonstration on Toy Problem



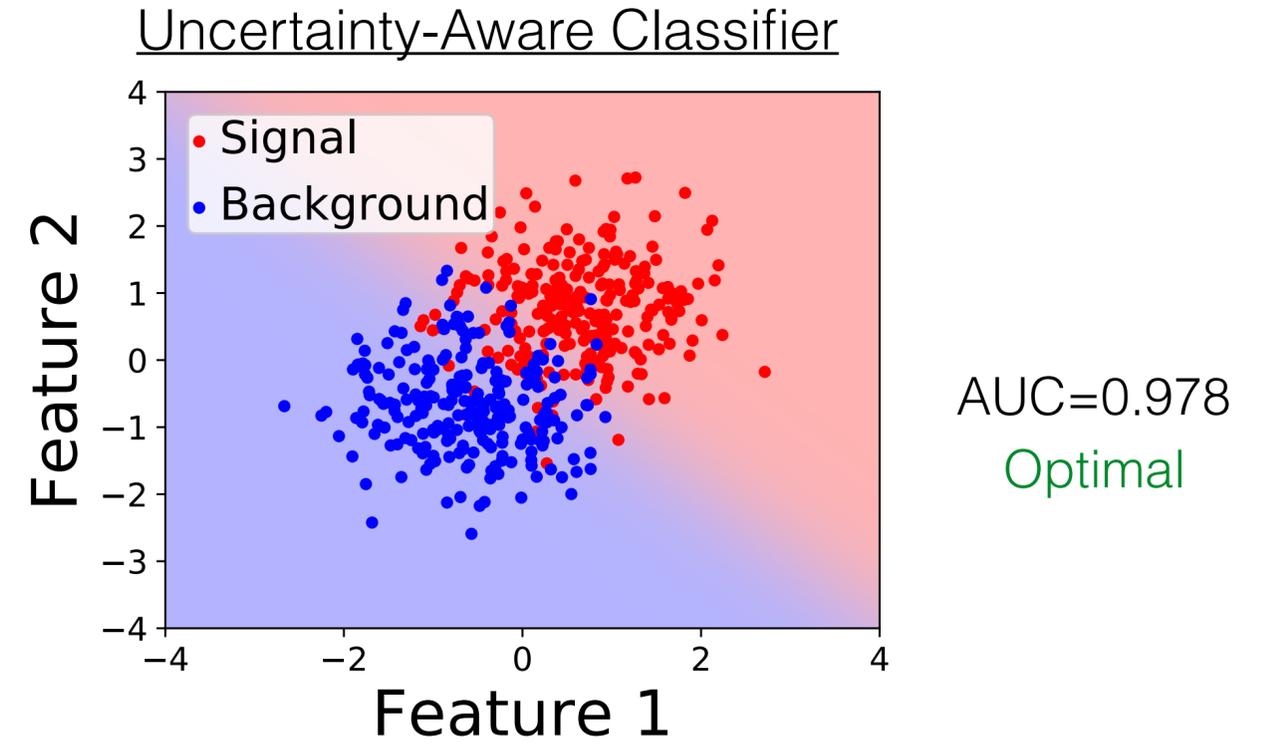
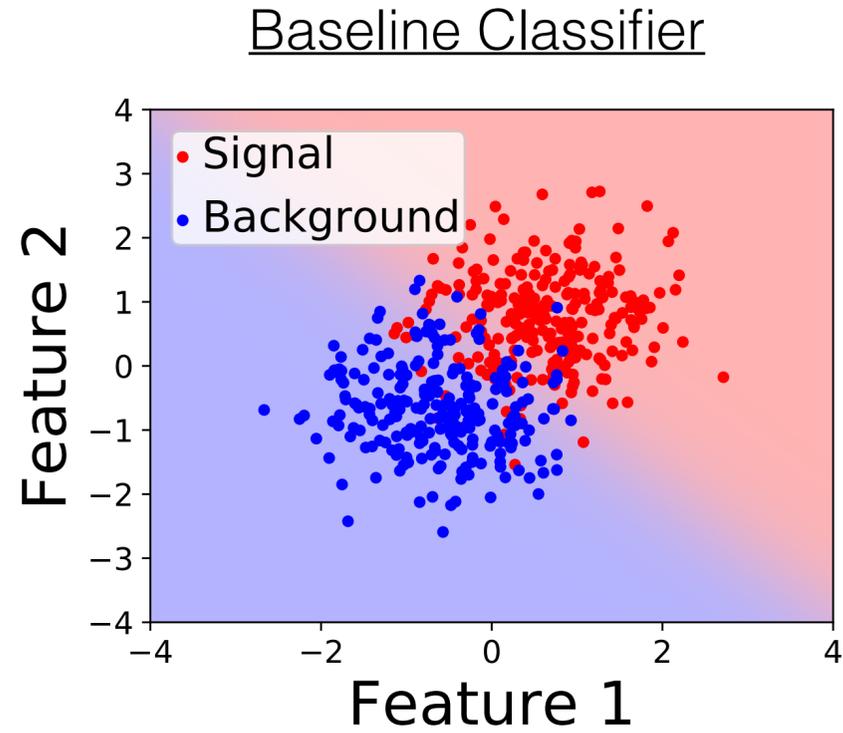
$$\bar{\mu} = \frac{N_{s,obs}}{N_{s,exp}}$$

$z = \text{Angle}$

Being invariant to Z would result in a terrible classifier

Nominal and Systematic Up Examples

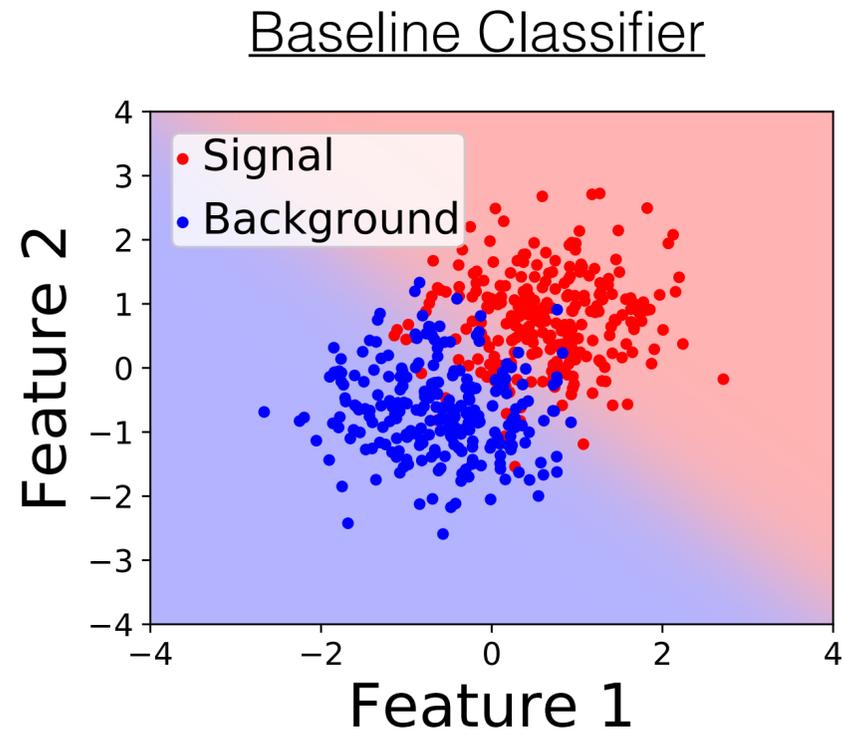
Nominal "Data"



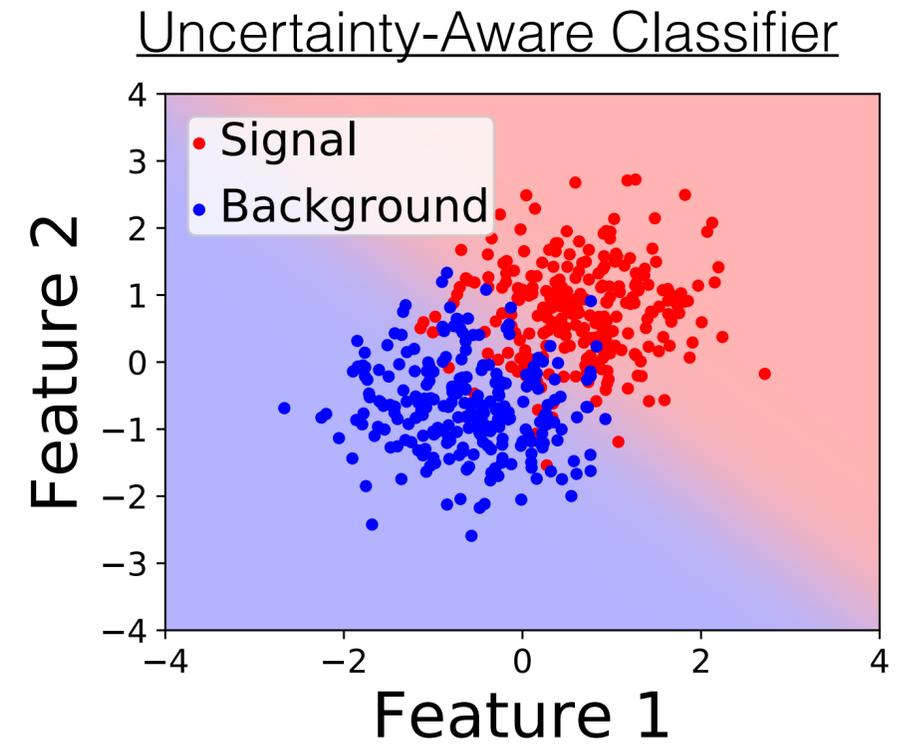
SystUp "Data"

Nominal and Systematic Up Examples

Nominal "Data"

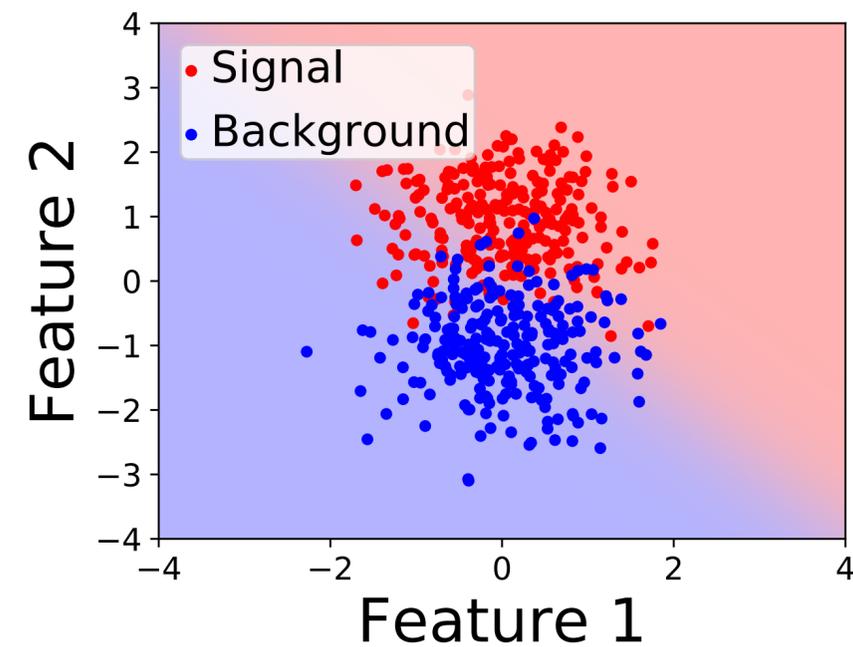


AUC=0.978
Optimal



AUC=0.978
Optimal

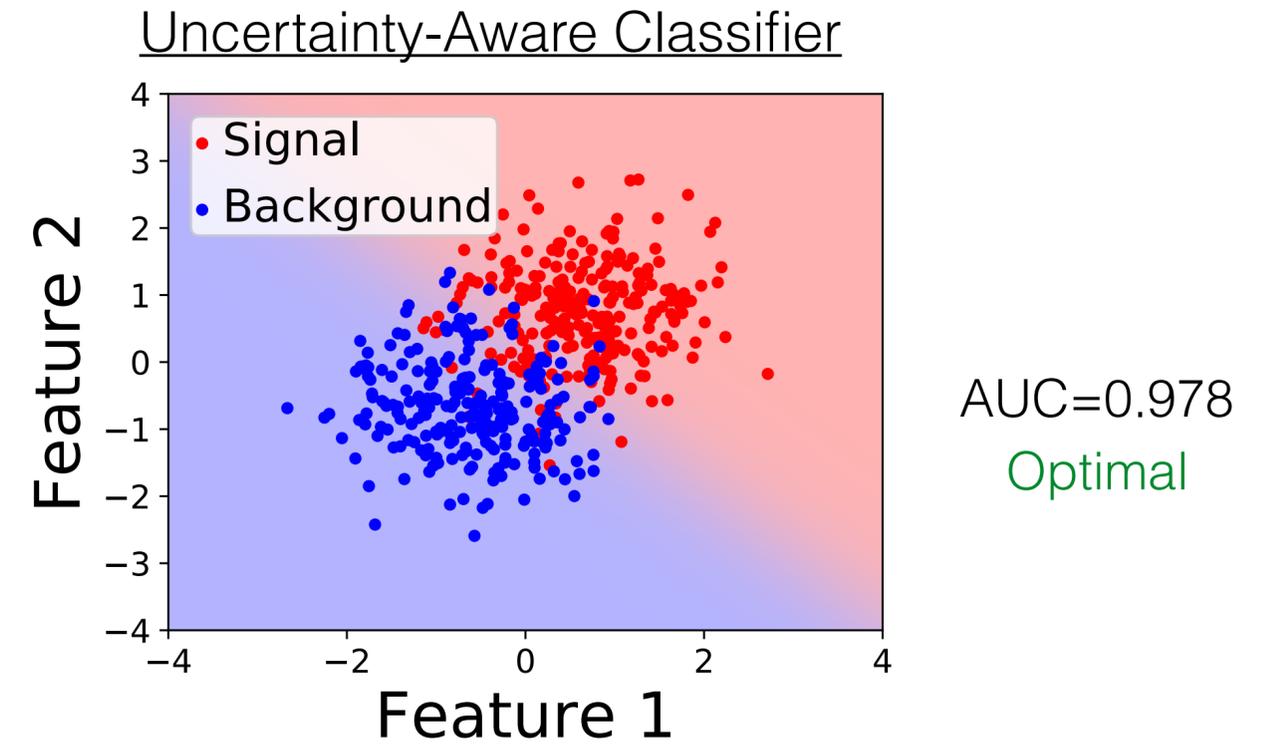
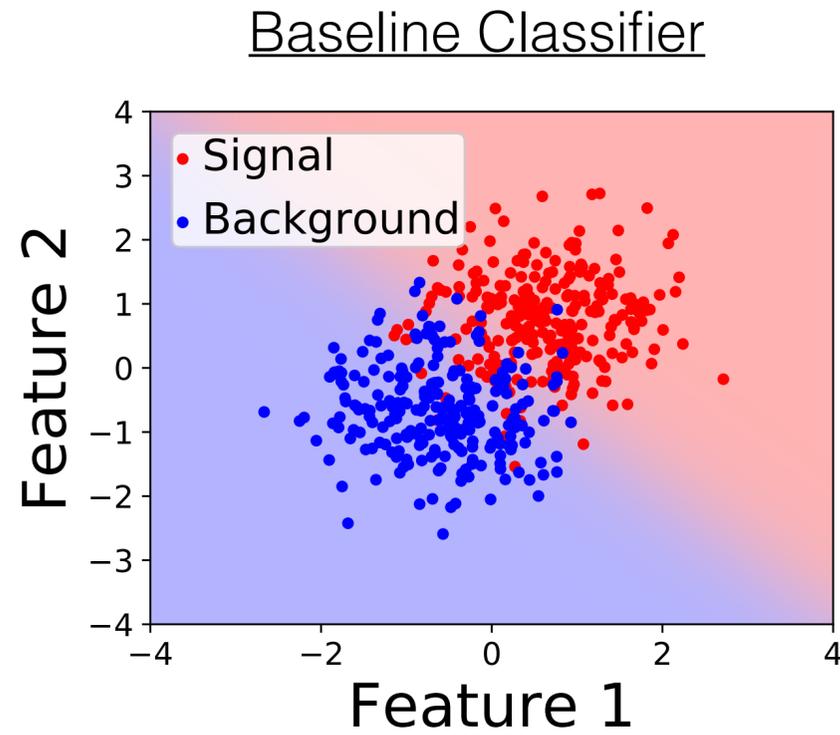
SystUp "Data"



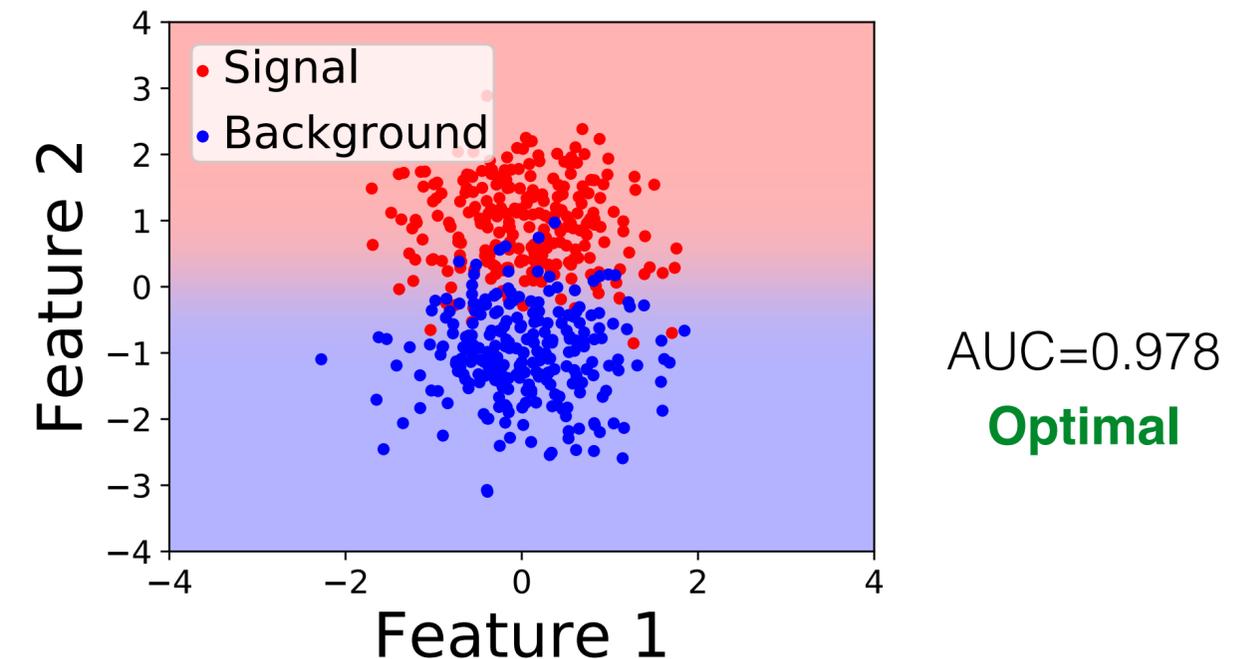
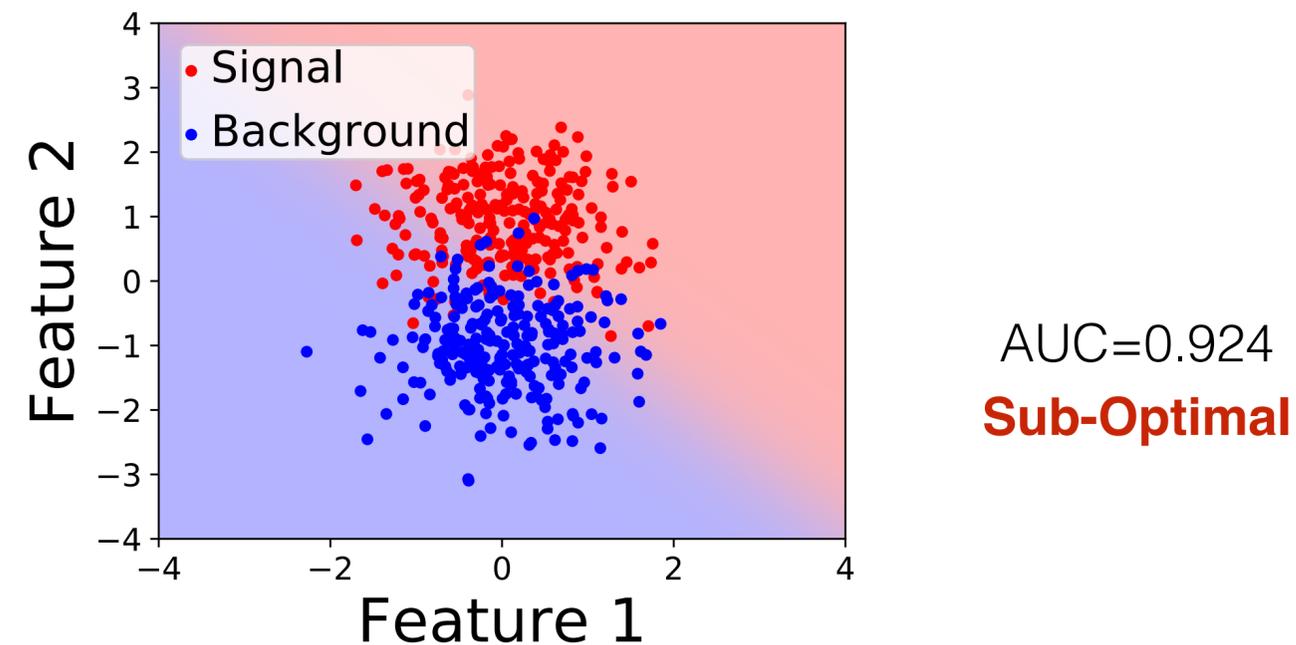
AUC=0.924
Sub-Optimal

Nominal and Systematic Up Examples

Nominal "Data"



SystUp "Data"

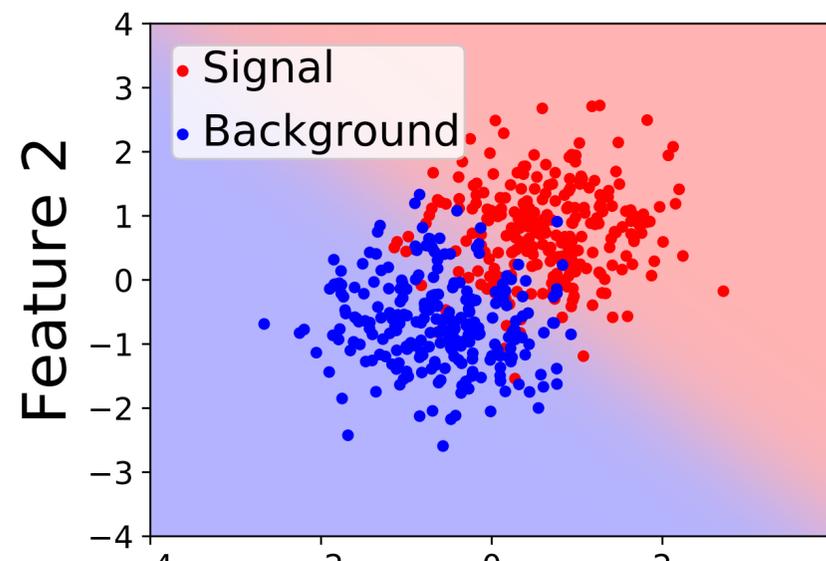


Syst-Aware Classifier is able to rotate its decision function based on Z while the Baseline Classifier decision function remains frozen 11

Nominal and Systematic Up Examples

Nominal "Data"

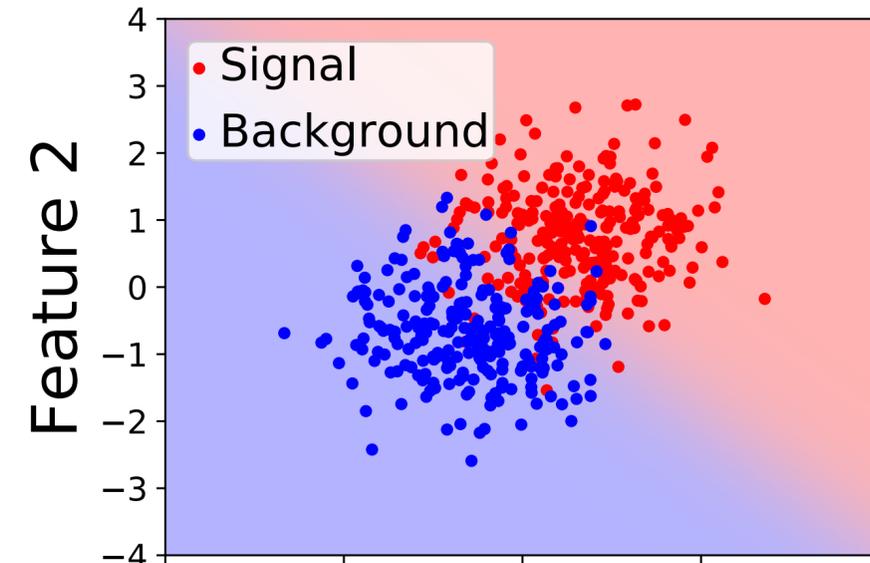
Baseline Classifier



AUC=0.978

Optimal

Uncertainty-Aware Classifier

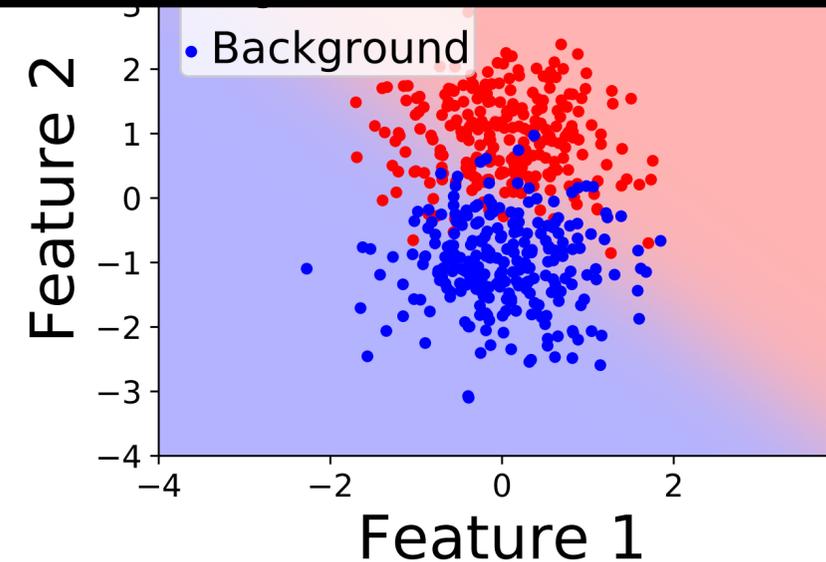


AUC=0.978

Optimal

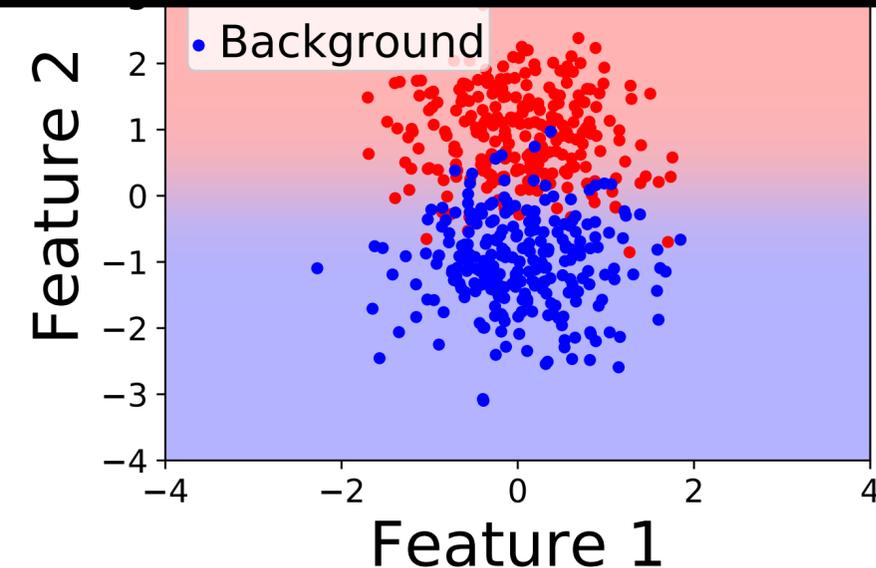
In fact for the toy problem we used analytical classifiers instead of training networks

SystUp "Data"



AUC=0.924

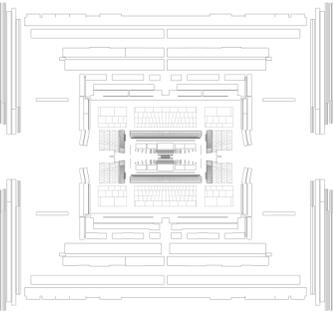
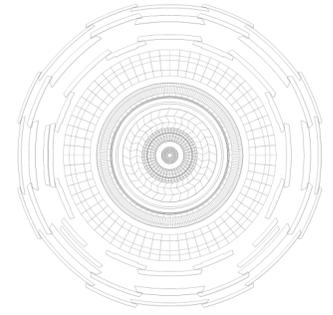
Sub-Optimal



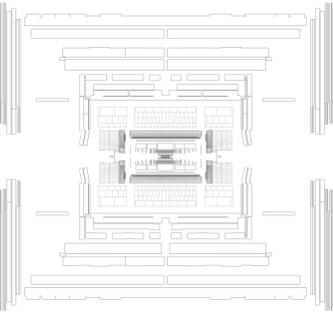
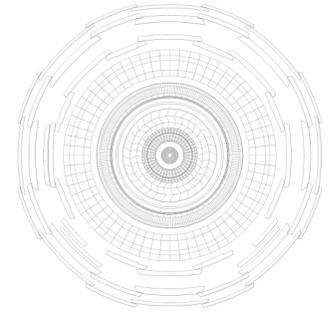
AUC=0.978

Optimal

Syst-Aware Classifier is able to rotate its decision function based on Z while the Baseline Classifier decision function remains frozen 11



But in a real measurement we don't know true Z a priori,
would this still help?

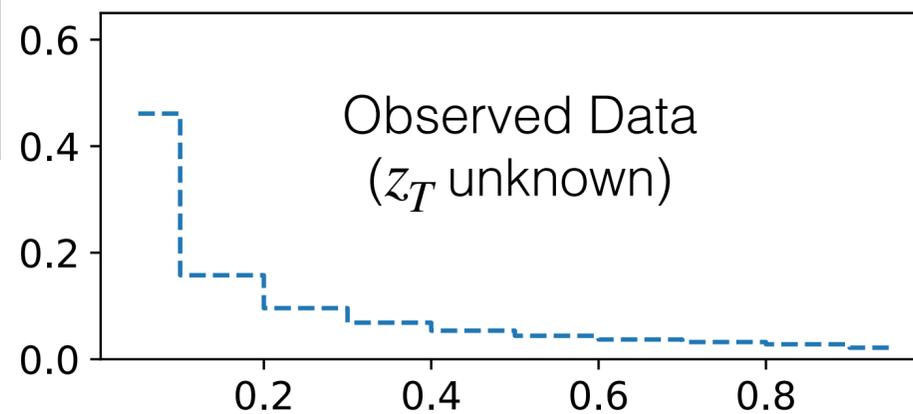
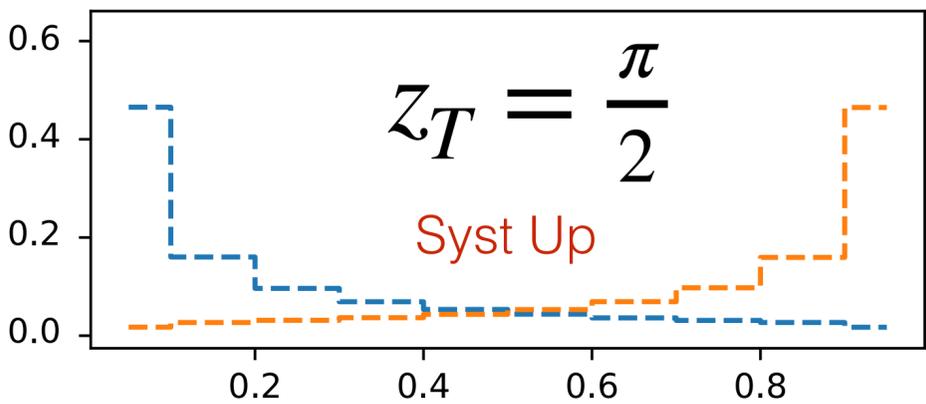
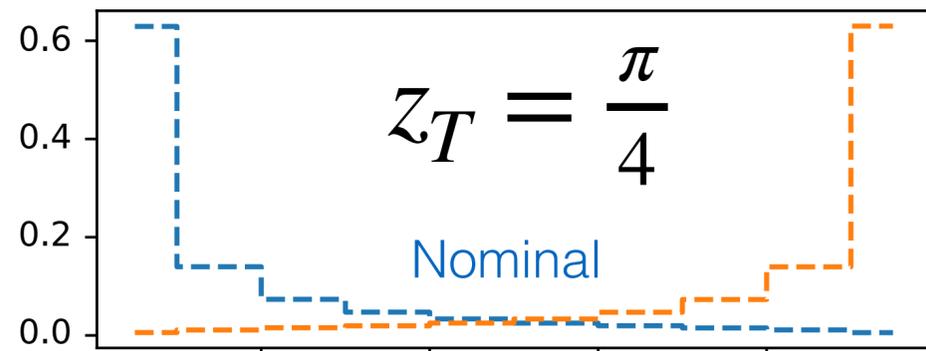
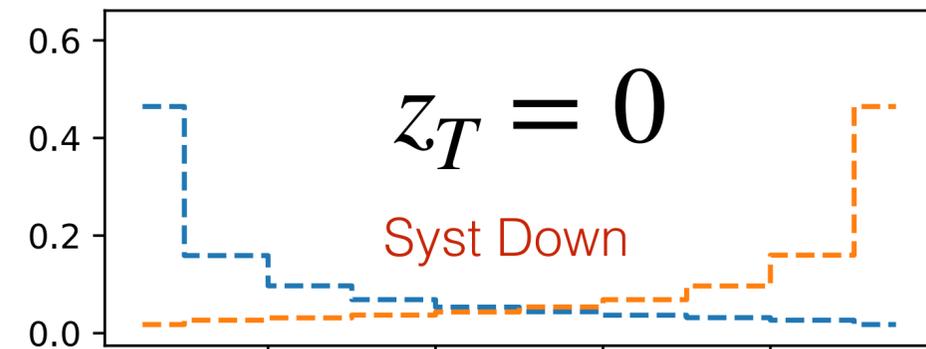


But in a real measurement we don't know true Z a priori,
would this still help?

Let's see what we'll need to do..

Scan the 2D Likelihood space in Z vs μ

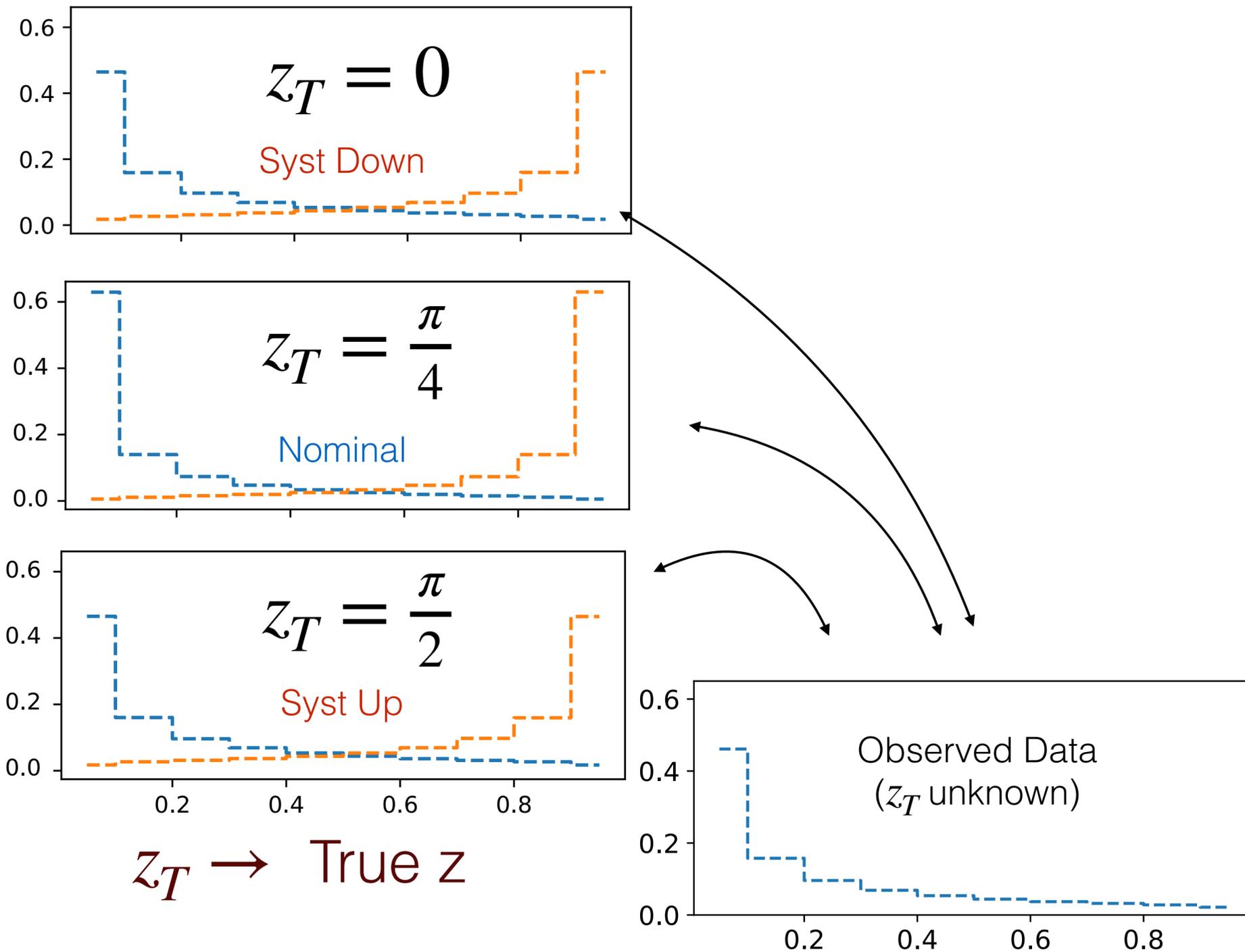
Template **Baseline Classifier** Score Histograms for various Z



$z_T \rightarrow$ True z

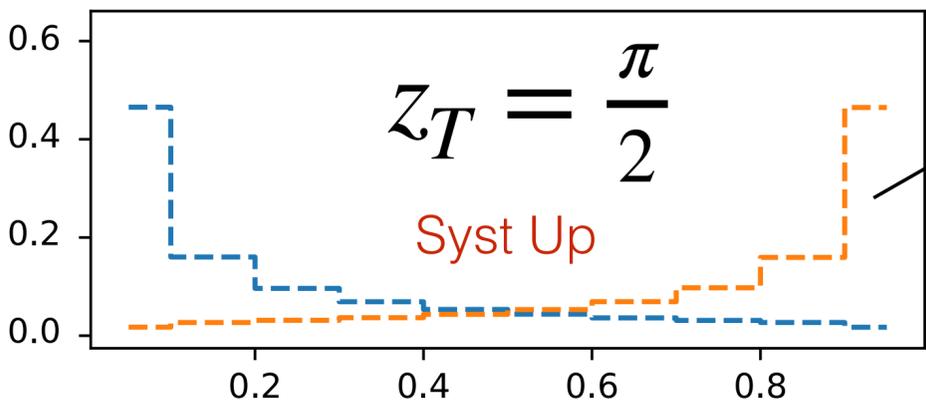
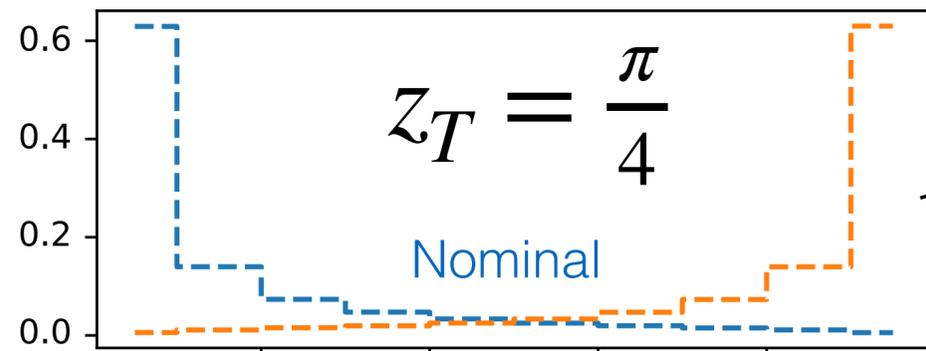
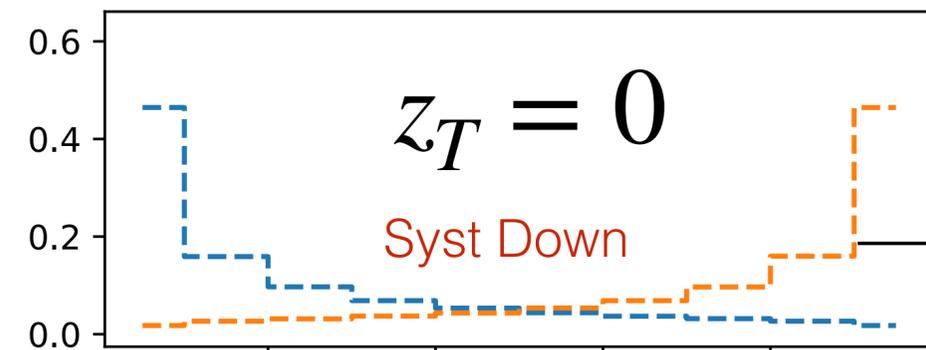
Scan the 2D Likelihood space in Z vs μ

Template **Baseline Classifier** Score Histograms for various Z

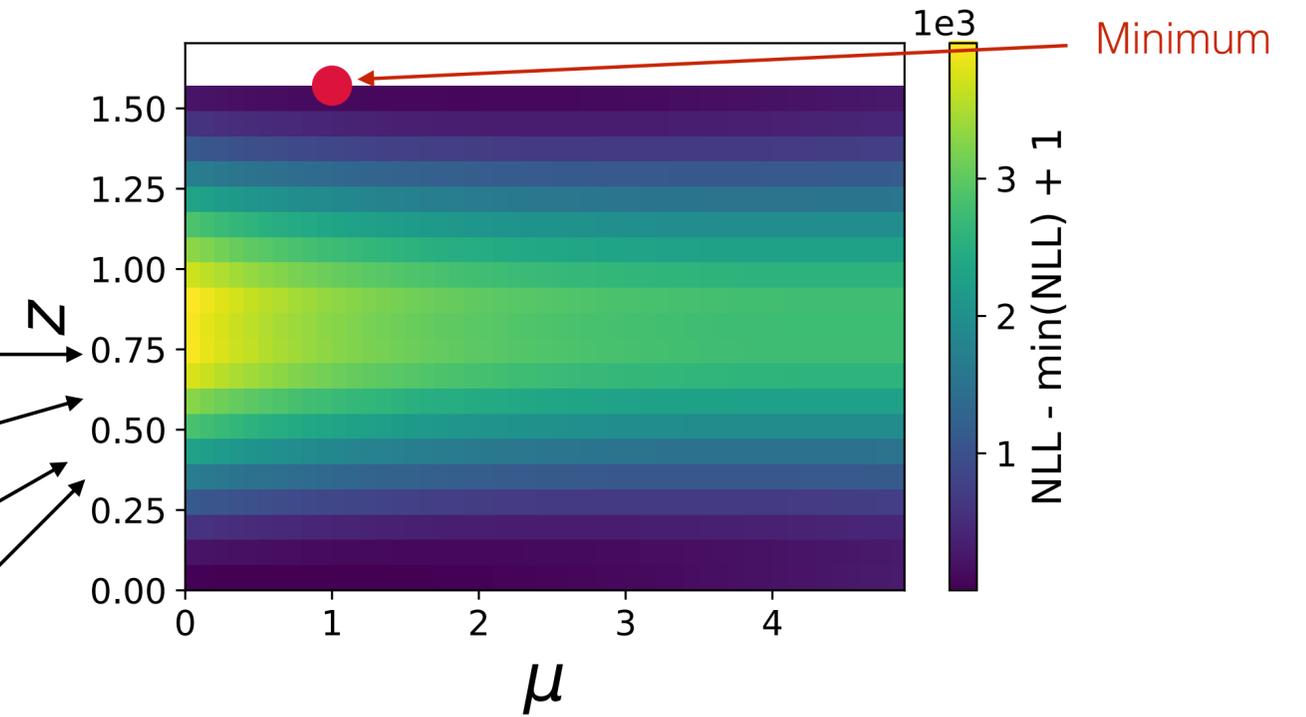
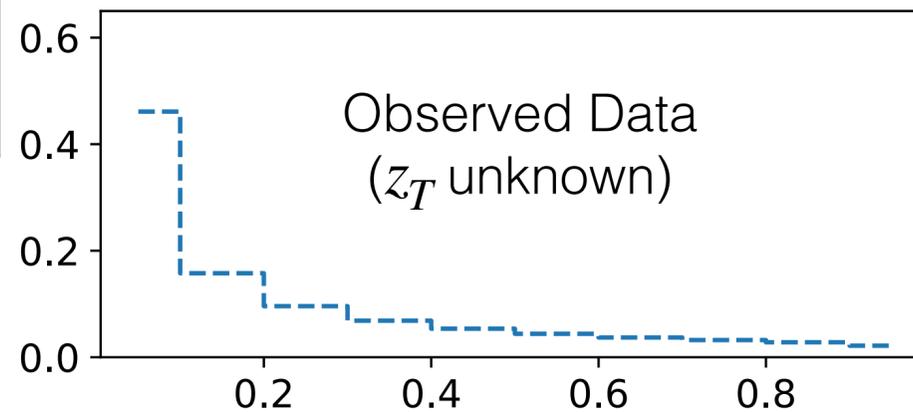


Scan the 2D Likelihood space in Z vs μ

Template **Baseline Classifier** Score Histograms for various Z

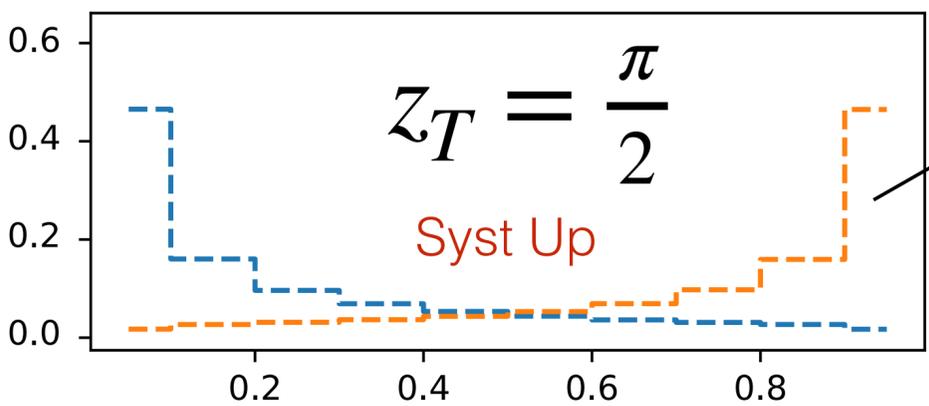
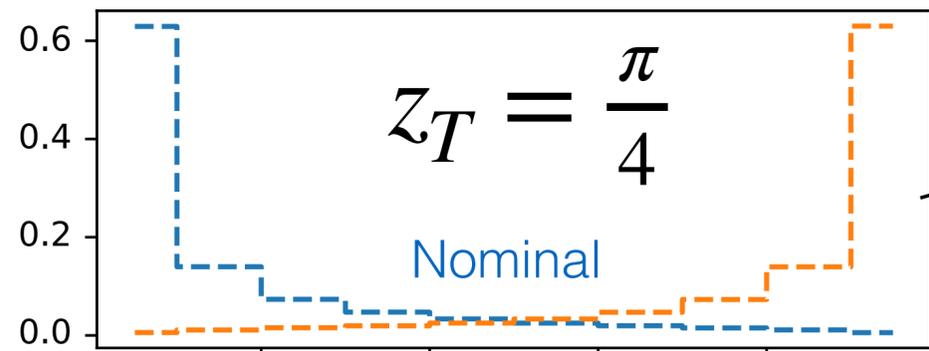
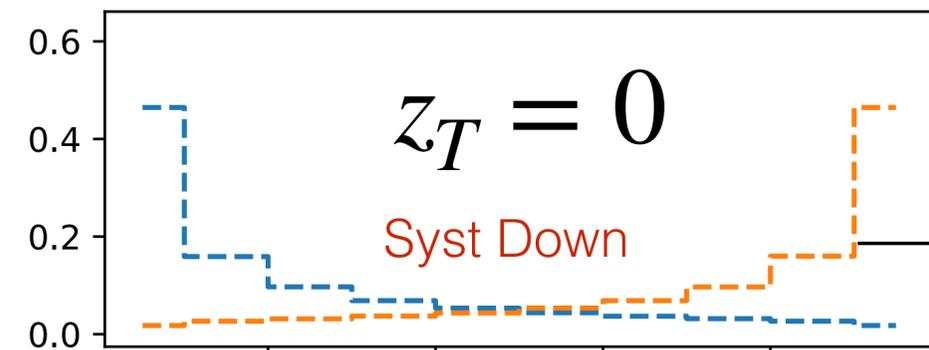


$z_T \rightarrow$ True z

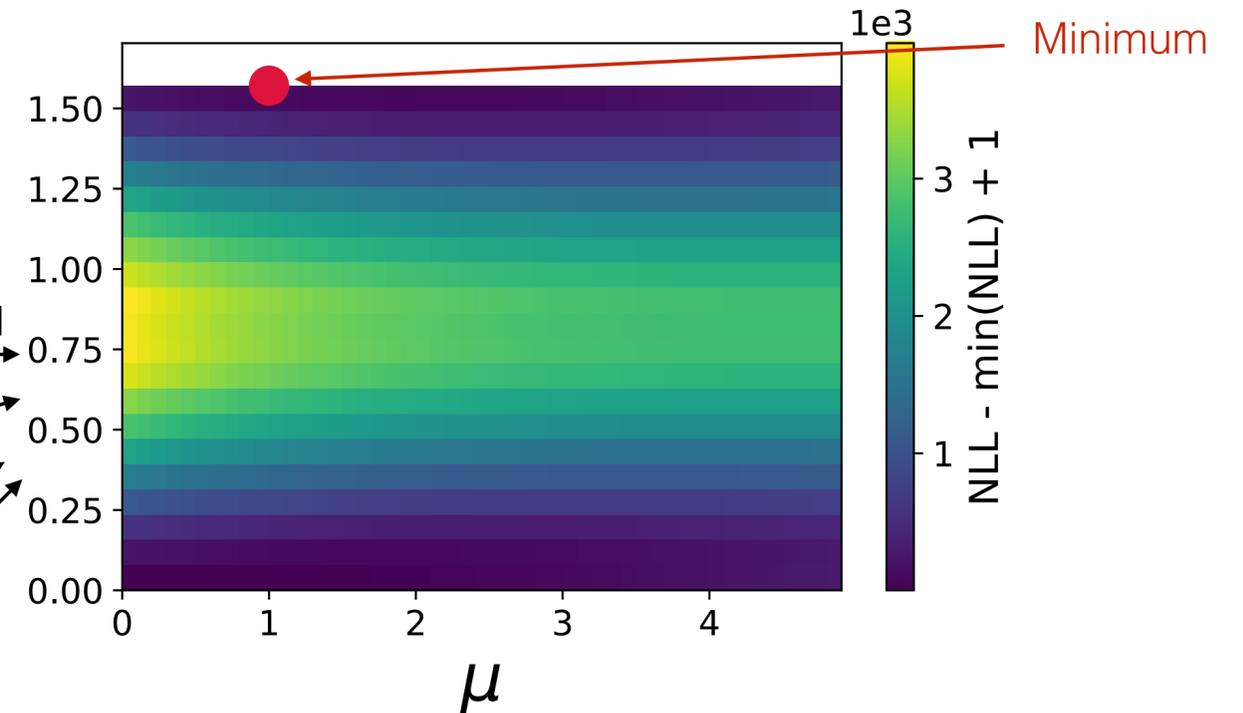
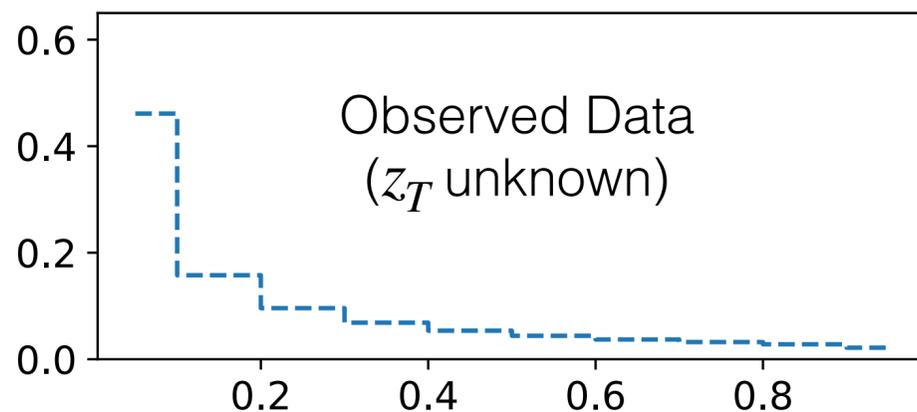


Scan the 2D Likelihood space in Z vs μ

Template **Baseline Classifier** Score Histograms for various Z



$z_T \rightarrow$ True z

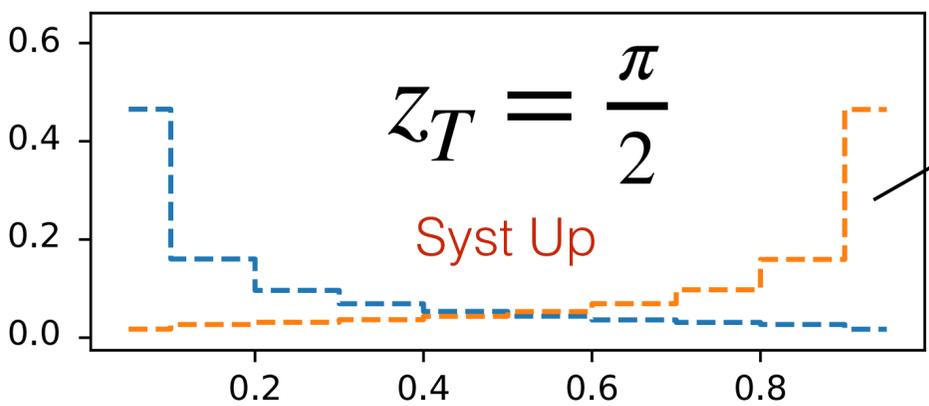
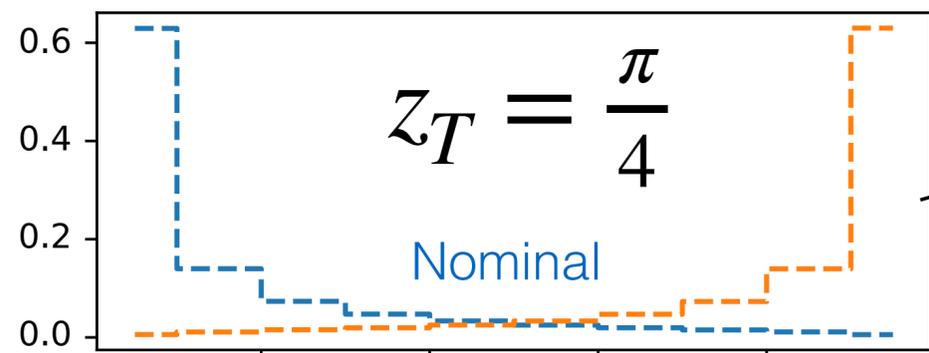
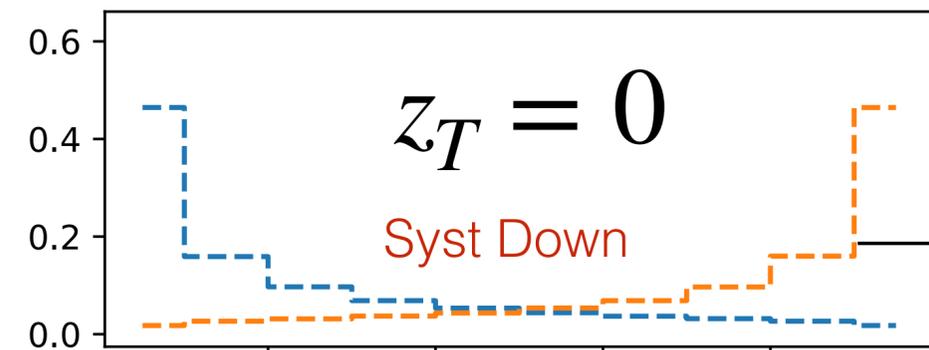


Likelihood statistical component = Poisson per histogram bin
Likelihood systematic component = Gaussian (1, 0.5) as prior on Z
Full Likelihood = statistical + systematic

$$\begin{aligned}
 & -\log \mathcal{L}(\mu, z | \{x_i\}) \\
 &= -\sum_{j=1}^{n_{\text{bins}}} \left[N_j \cdot \log(\mu s_j + b_j) - \mu s_j - b_j - \log(\Gamma(N_j)) \right] \\
 & \quad + \left(\frac{z - z_0}{\sqrt{2}\sigma_z} \right)^2,
 \end{aligned}$$

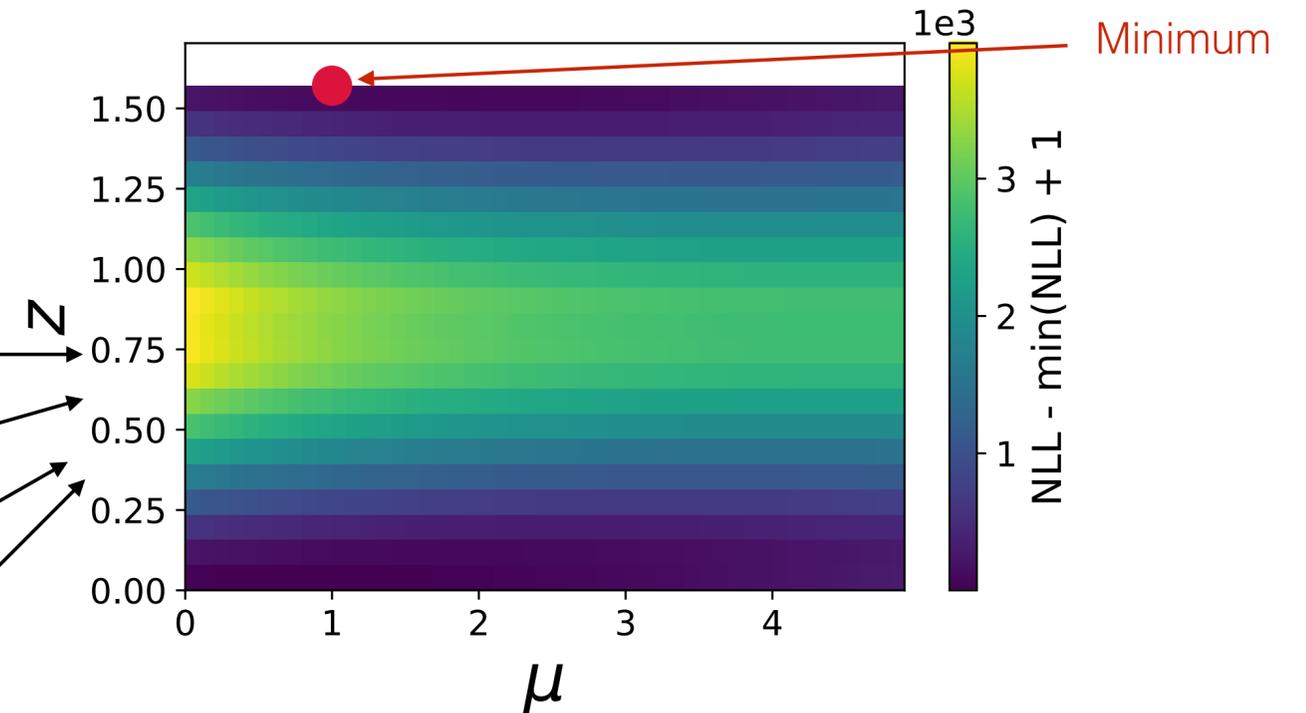
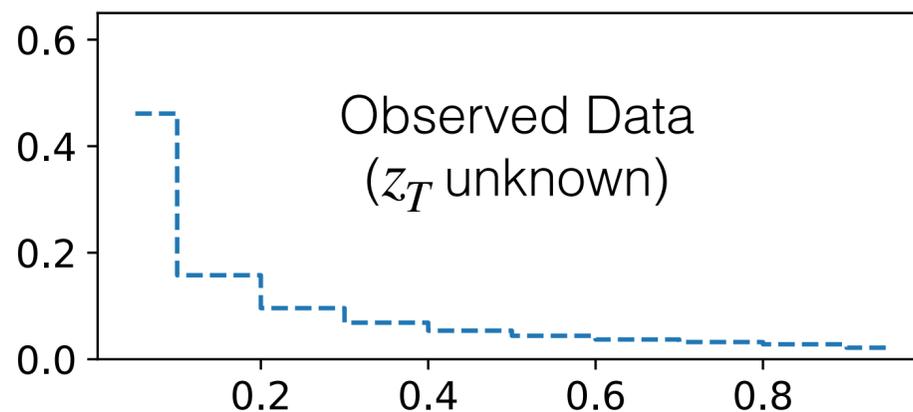
Scan the 2D Likelihood space in Z vs μ

Template **Baseline Classifier** Score Histograms for various Z



$z_T \rightarrow$ True z

But could be done unbinned/KDE too



Likelihood statistical component = Poisson per histogram bin
Likelihood systematic component = Gaussian (1, 0.5) as prior on Z
Full Likelihood = statistical + systematic

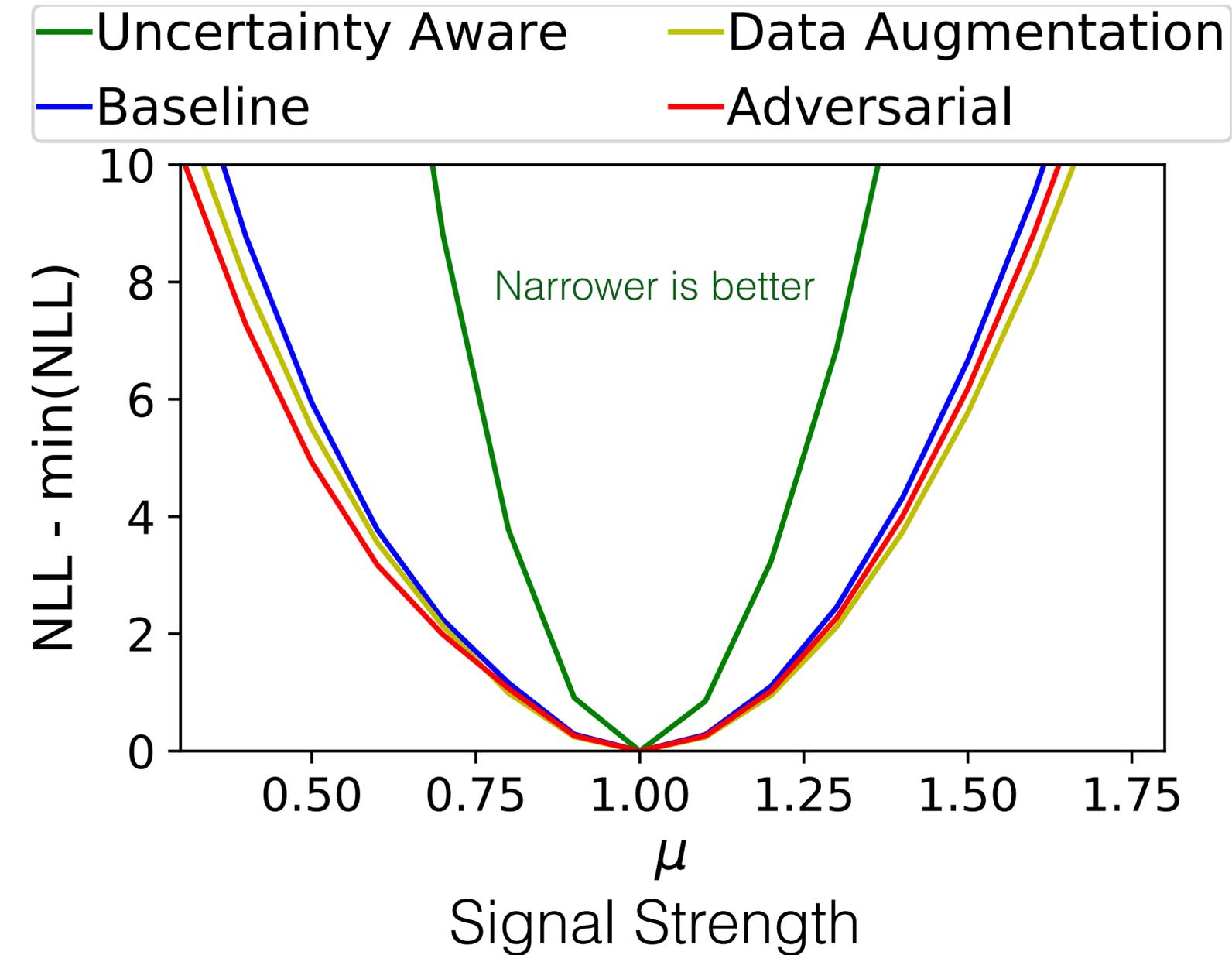
$$-\log \mathcal{L}(\mu, z | \{x_i\})$$

$$= - \sum_{j=1}^{n_{\text{bins}}} \left[N_j \cdot \log(\mu s_j + b_j) - \mu s_j - b_j - \log(\Gamma(N_i)) \right]$$

$$+ \left(\frac{z - z_0}{\sqrt{2}\sigma_z} \right)^2,$$

Next step: profile over Z dimension (take the bin with maximum likelihood in each column)

Profile away Z - Example at $(\mu, Z)_{\text{True}} = (1, 1.57)$



Narrower is better: We can exclude wrong values of μ with greater confidence.

The profiled (Negative-Log-) Likelihood curve for Uncertainty-Aware classifier is much narrower \Rightarrow smallest [statistical + systematic] uncertainty on measurement

Profile Likelihood

Standard method of including the systematic uncertainty into the likelihood computation

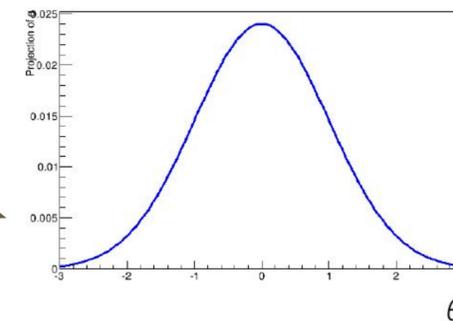
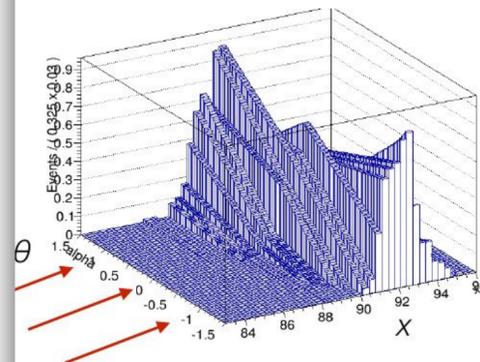
We simply make the selection/observable a function of z

In principle could also be done in cut-based analysis: make cut a continuous function of z

The Profile Likelihood approach

- The profile likelihood is a way to include **systematic uncertainties in the likelihood**
 - systematics included as "**constrained**" nuisance parameters
 - the idea behind is that systematic uncertainties on the measurement of μ come from **imperfect knowledge** of parameters of the model (S and B prediction)
 - still *some knowledge* is implied: " $\theta = \theta_0 \pm \Delta\theta$ "

$$\mathcal{L}(\mathbf{n}, \theta^0 | \mu, \theta) = \prod_{i \in \text{bins}} \mathcal{P}(n_i | \mu \cdot S_i(\theta) + B_i(\theta)) \times \prod_{j \in \text{syst}} \mathcal{G}(\theta_j^0 | \theta_j, \Delta\theta_j)$$



- usually $\theta^0=0$ and $\Delta\theta=1$ (convention)
- define **effect of systematic j** on prediction x in bin i at "+1" and "-1",
- then interpolate & extrapolate for any value of θ

- external / *a priori* knowledge interpreted as "**auxiliary/subsidiary measurement**", implemented as **constraint/penalty term**, i.e. probability density function (usually Gaussian, interpreting " $\pm\Delta\theta$ " as Gaussian standard deviation)

3

From Michele Pinamonti's talk:

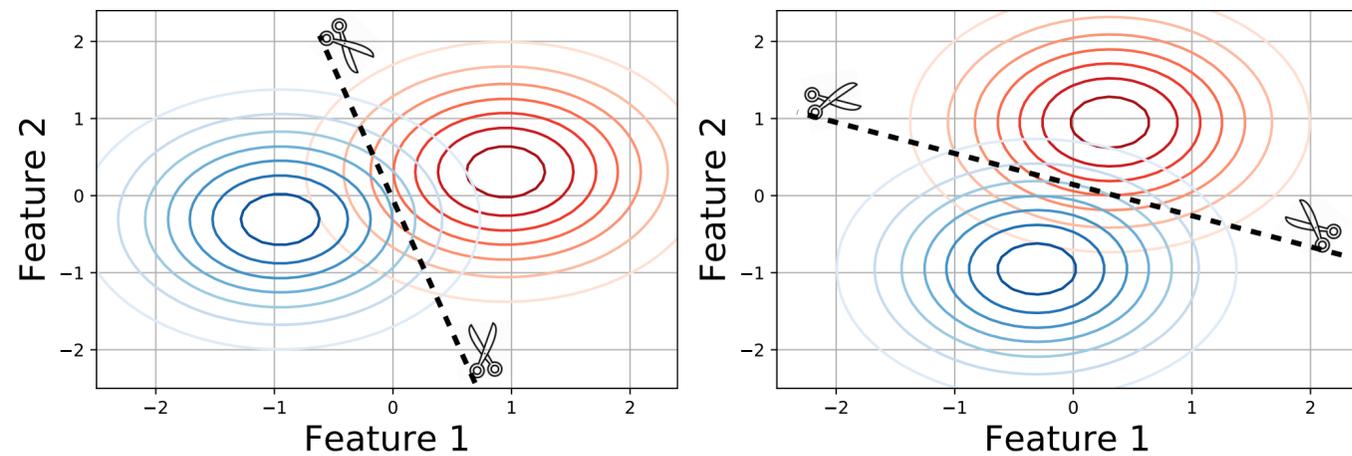
https://indico.cern.ch/event/727396/contributions/3021899/attachments/1657532/2654085/Statistical_methods_at_ATLAS_and_CMS_2.pdf

Profile Likelihood

Standard method of including the systematic uncertainty into the likelihood computation

We simply make the selection/observable a function of z

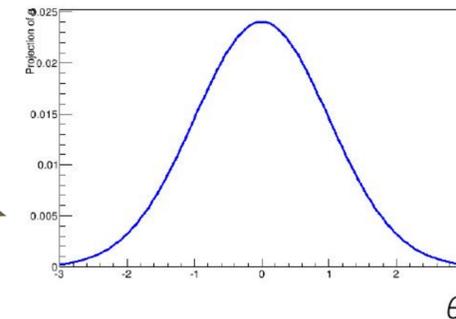
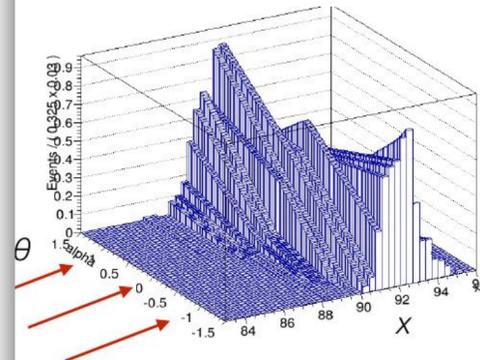
In principle could also be done in cut-based analysis: make cut a continuous function of z



The Profile Likelihood approach

- The profile likelihood is a way to include **systematic uncertainties in the likelihood**
 - systematics included as "**constrained**" nuisance parameters
 - the idea behind is that systematic uncertainties on the measurement of μ come from **imperfect knowledge** of parameters of the model (S and B prediction)
 - still *some knowledge* is implied: " $\theta = \theta_0 \pm \Delta\theta$ "

$$\mathcal{L}(\mathbf{n}, \theta^0 | \mu, \theta) = \prod_{i \in \text{bins}} \mathcal{P}(n_i | \mu \cdot S_i(\theta) + B_i(\theta)) \times \prod_{j \in \text{syst}} \mathcal{G}(\theta_j^0 | \theta_j, \Delta\theta_j)$$



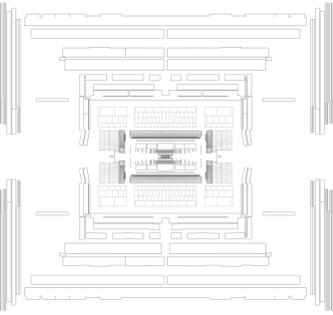
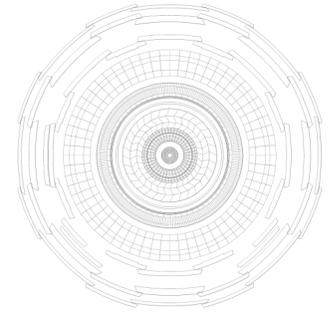
- usually $\theta^0=0$ and $\Delta\theta=1$ (convention)
- define **effect of systematic j** on prediction x in bin i at "+1" and "-1",
- then interpolate & extrapolate for any value of θ

- external / *a priori* knowledge interpreted as "**auxiliary/subsidiary measurement**", implemented as **constraint/penalty term**, i.e. probability density function (usually Gaussian, interpreting " $\pm\Delta\theta$ " as Gaussian standard deviation)

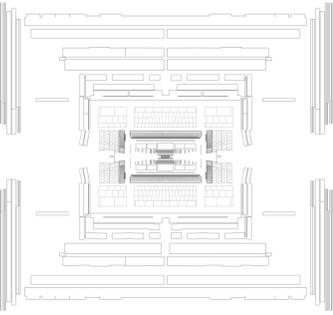
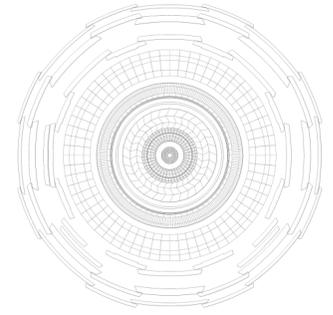
3

From Michele Pinamonti's talk:

https://indico.cern.ch/event/727396/contributions/3021899/attachments/1657532/2654085/Statistical_methods_at_ATLAS_and_CMS_2.pdf

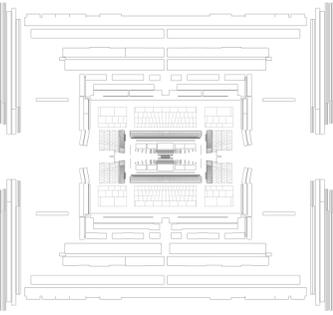
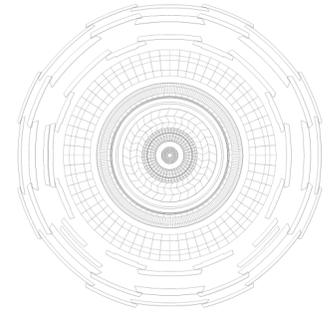


But in a real measurement we don't know true Z a priori,
would this still help?



But in a real measurement we don't know true Z a priori,
would this still help?

Yes!

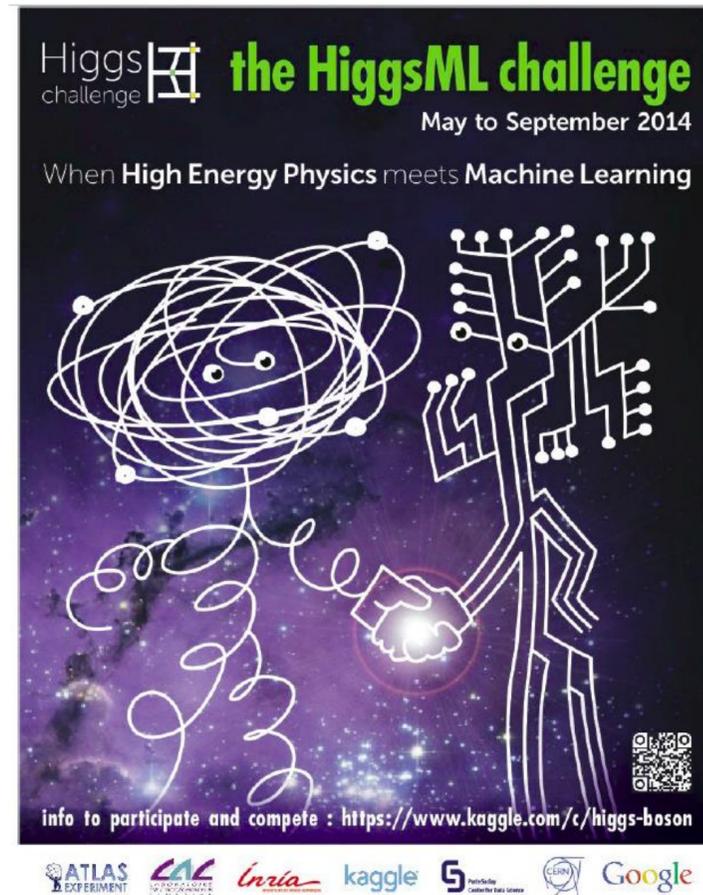


But in a real measurement we don't know true Z a priori,
would this still help?

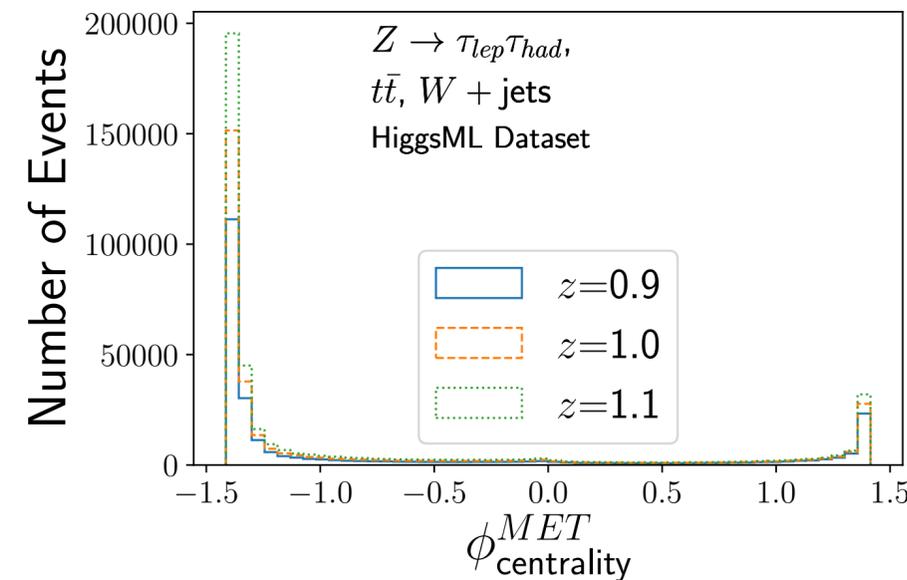
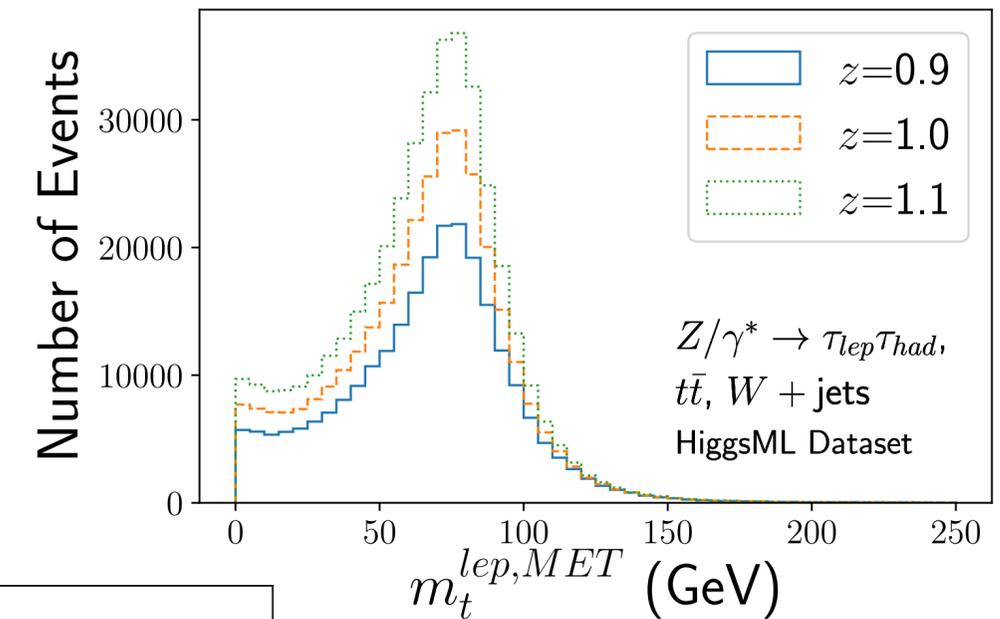
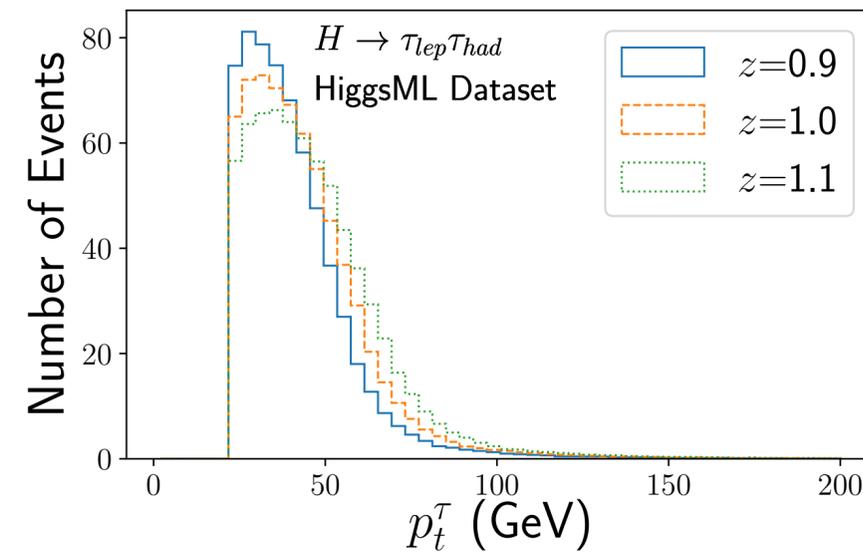
Yes!

Okay, it works on your handcrafted toy problem.
What about a real physics dataset?

HiggsML Public Dataset with Tau Energy Scale (TES) as Z



Parameter of Interest is Higgs signal strength μ , and TES is the nuisance parameter Z



We later realised dataset isn't ideal, stats limited..

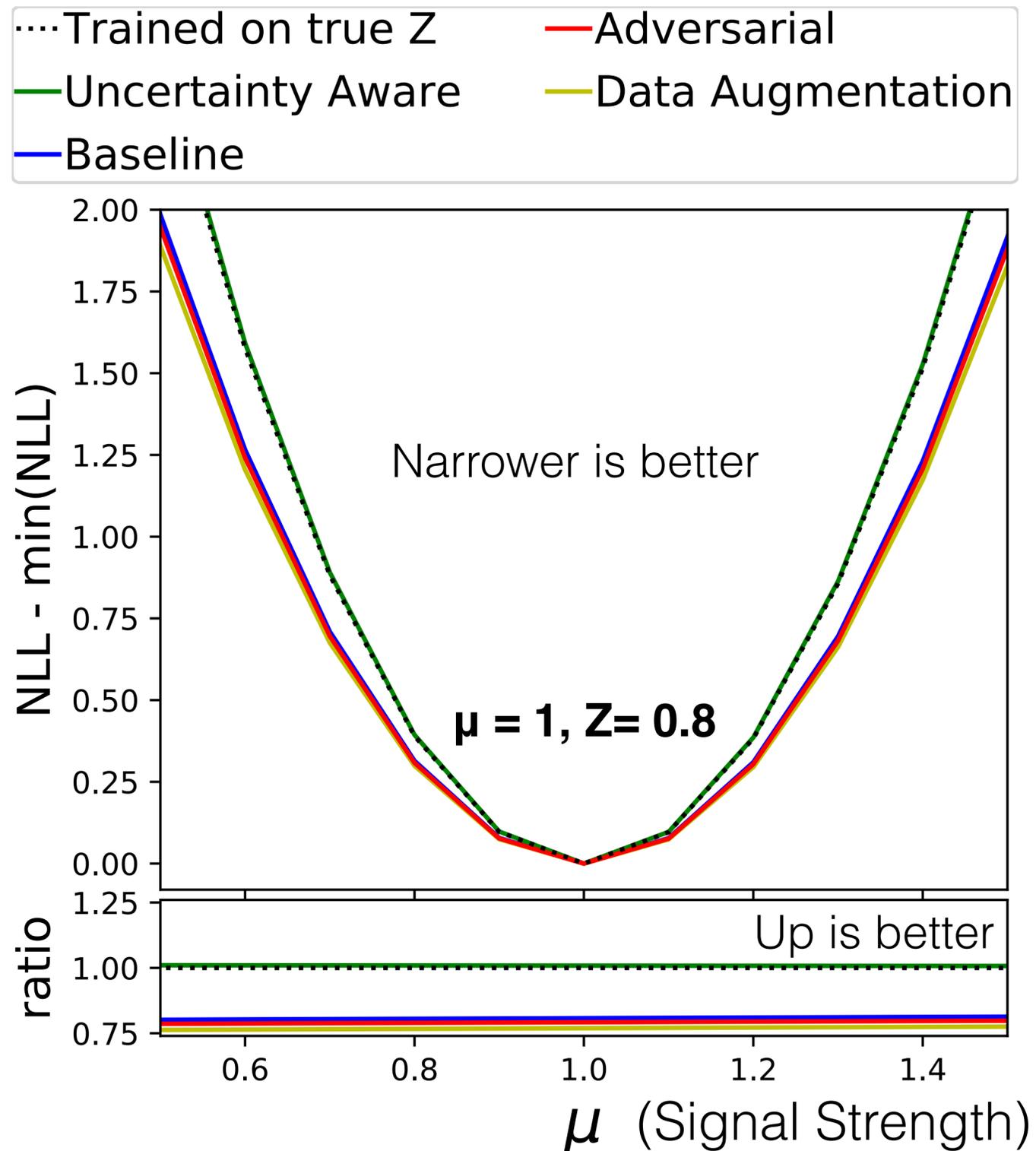
Test performance for “observed” at Systematic below Nominal

Train actual networks this time

$\mu = 1, Z = 0.8$

(Signal Strength)

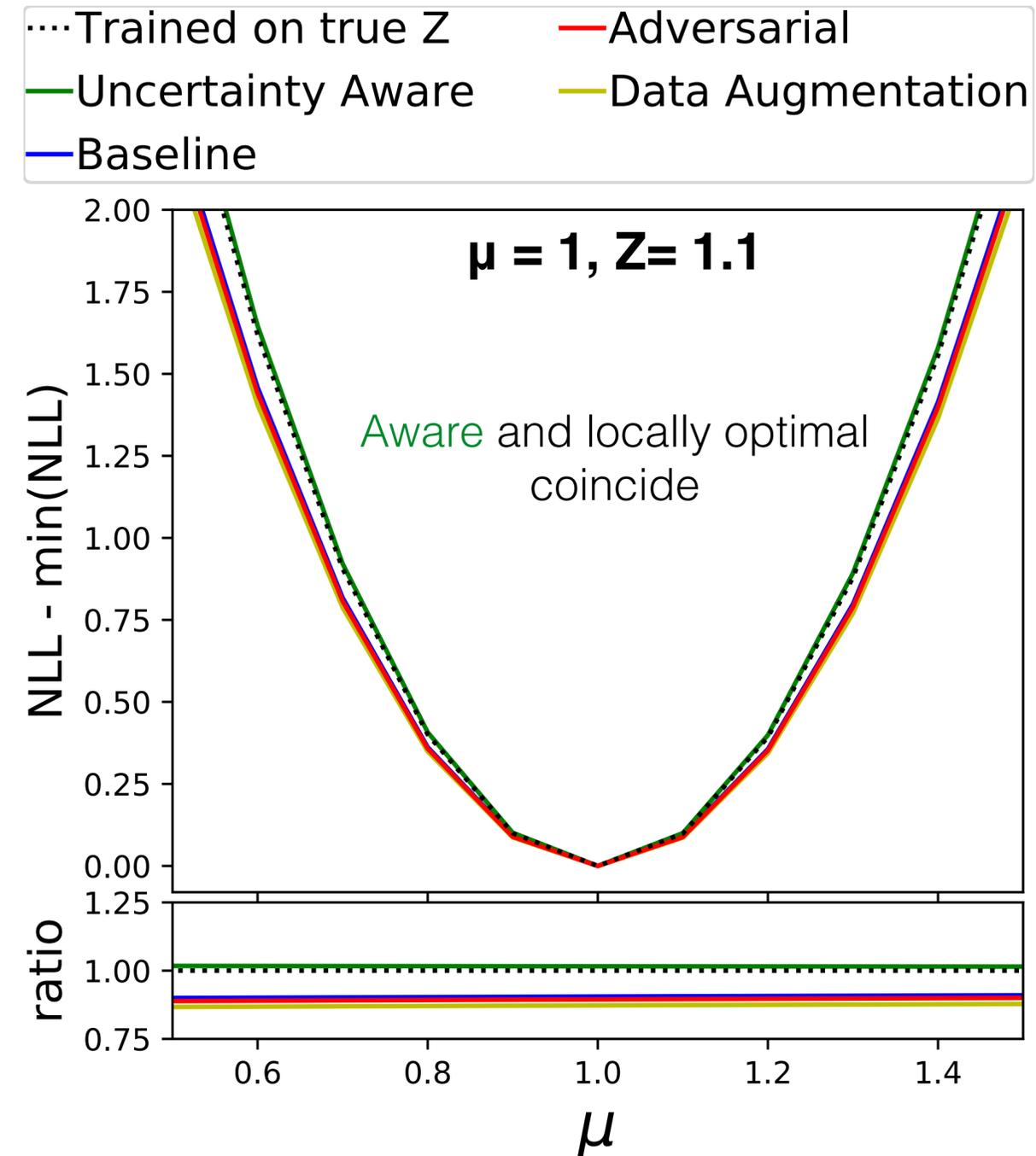
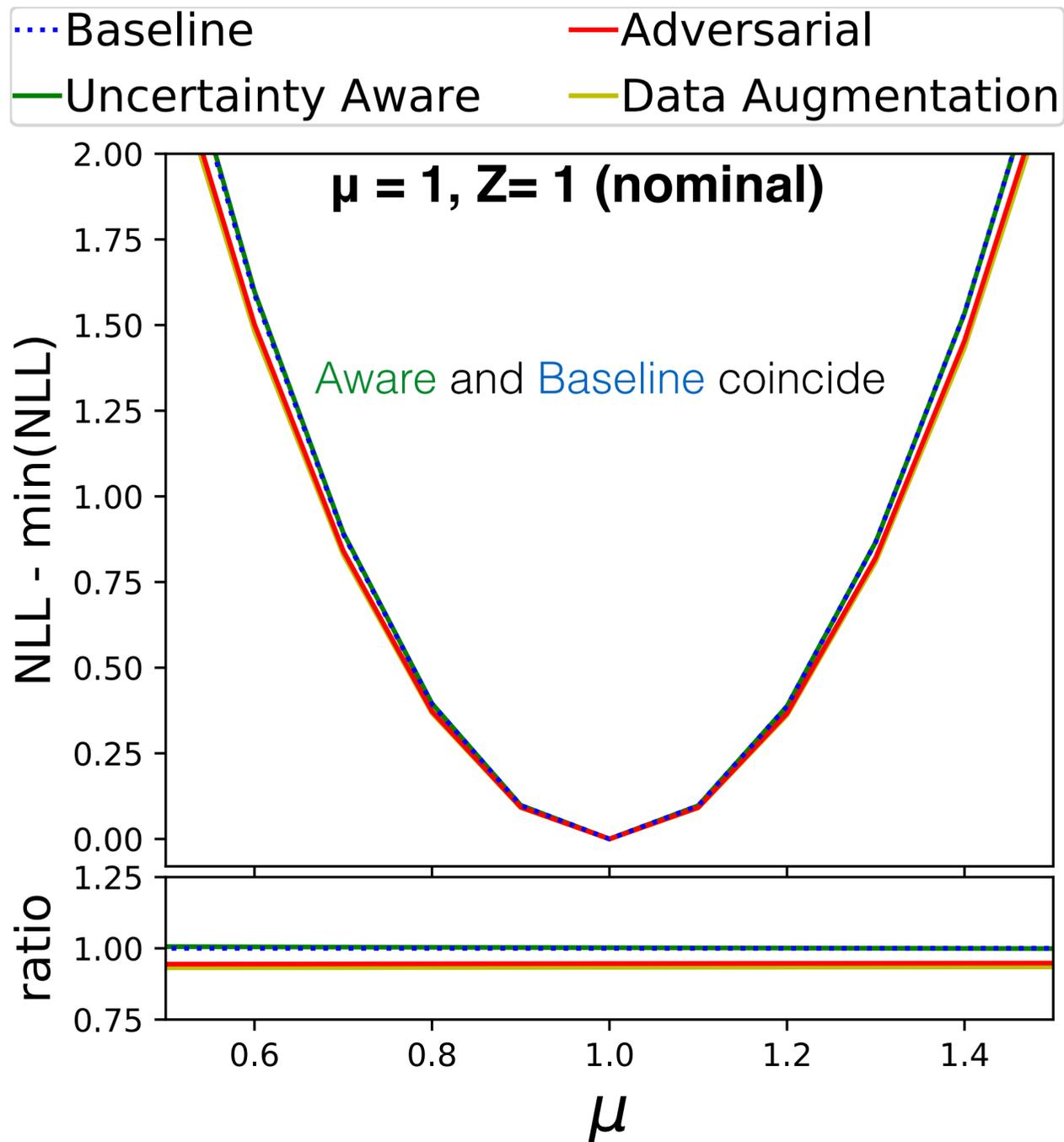
Test performance for “observed” at Systematic below Nominal



Train actual networks this time

Uncertainty-Aware coincides with classifier trained on true Z
 \Rightarrow It is optimal!

Test performance for “observed” datasets at nominal and above nominal Z

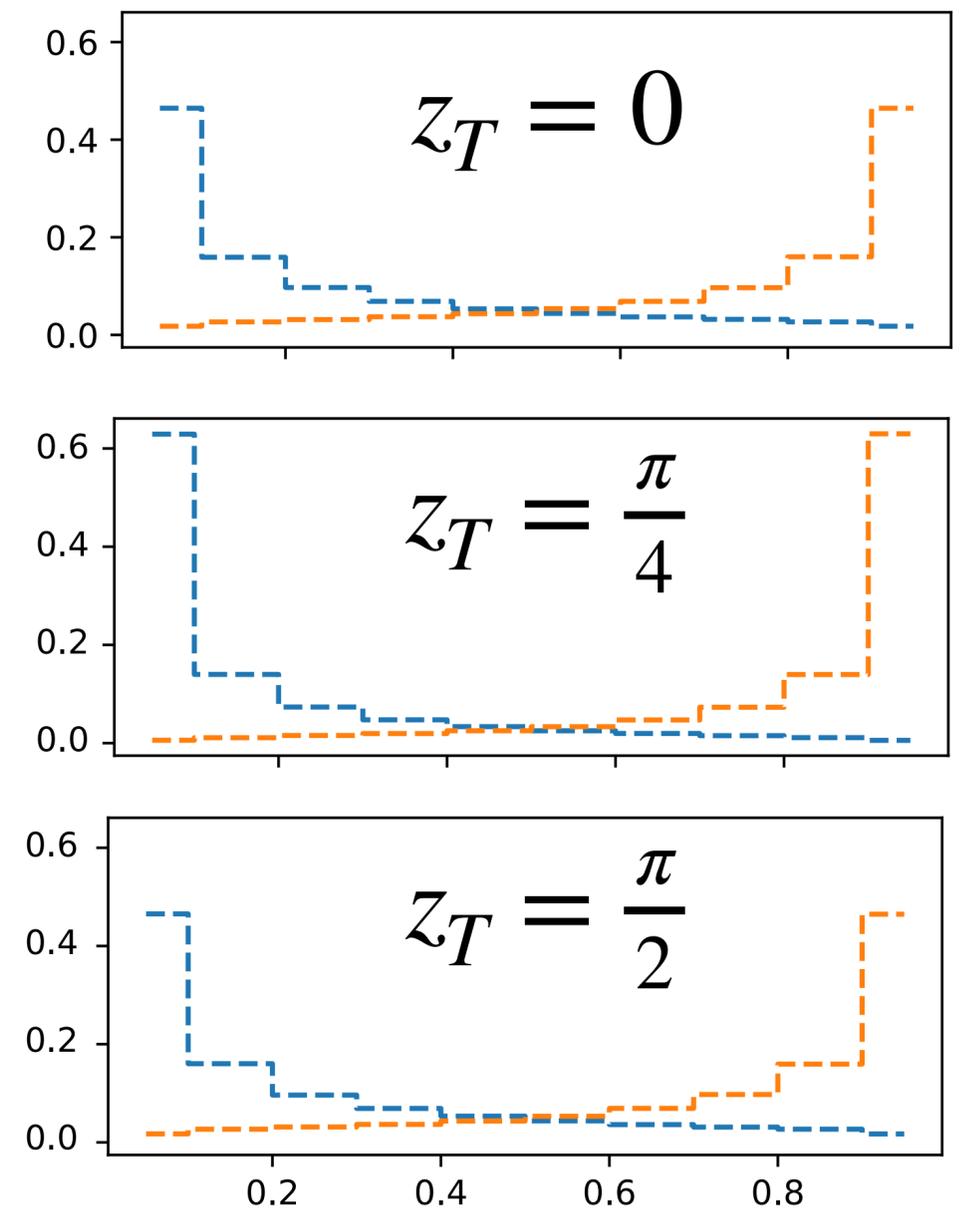


In every case the **Aware Classifier** is as good as the optimal one, no other technique matches its performance everywhere

Practical advantages of factorising inference

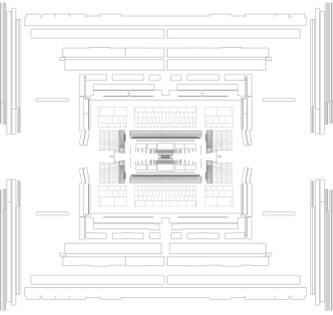
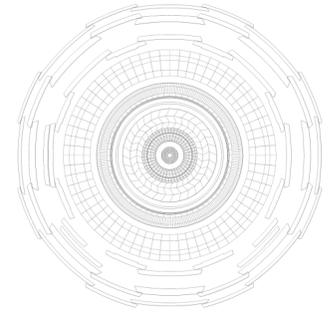
While using histogram (or KDE) templates seems clunky, it has practical advantages:

- More diagnostic tools: look at histograms, test for over-constraining of z
- Study impact of/profile over untrained nuisance parameters
- No worries about calibration of NN



Conclusion

- Systematic uncertainties are a nuisance in HEP, even more relevant as we collect more data
- Training a systematic aware classifier and profiling over the nuisance parameter provides performance similar to a locally optimal classifier
- This prescription can also handle auxiliary measurements of the systematic straightforwardly by combining the likelihoods
- Not a black-box procedure: Can also study impact of untrained systematics on sensitivity
- Solution scales to real physics dataset, easy to integrate into ATLAS/CMS chain



Backup

Auxiliary measurement of Z instead of prior

Simplistic auxiliary measurement of z_T

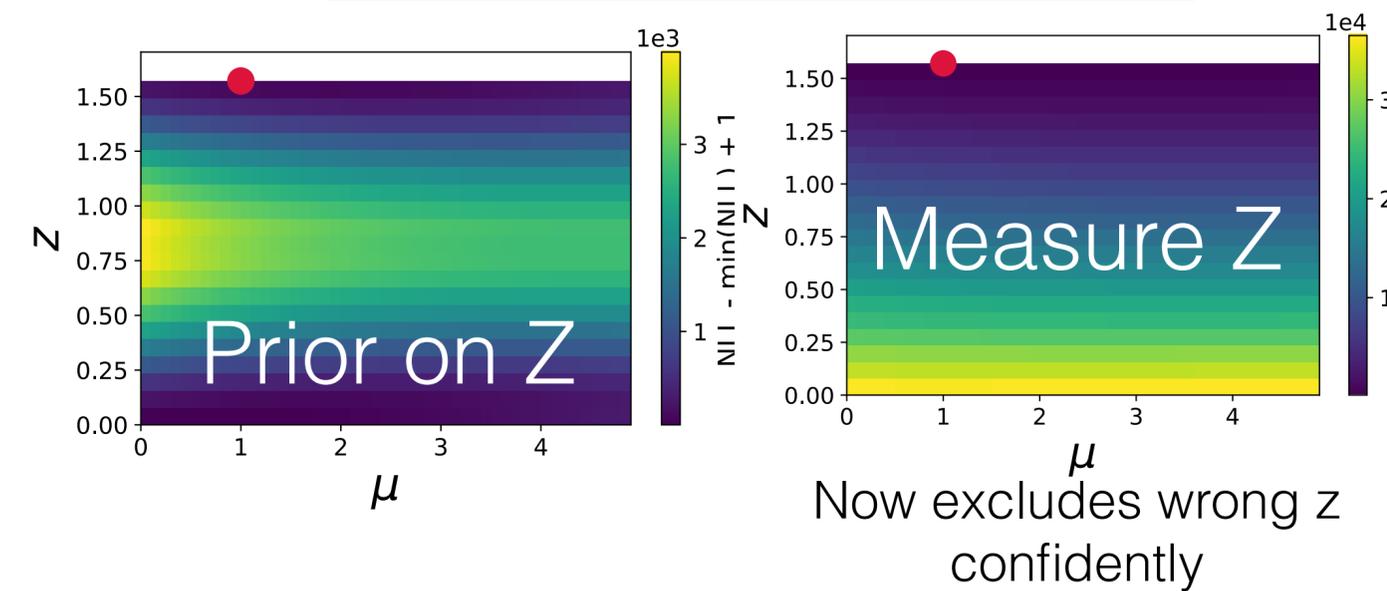
No need to re-train any network, change only in likelihood computation step

All methods provide improved limits on μ if Z is tightly constrained

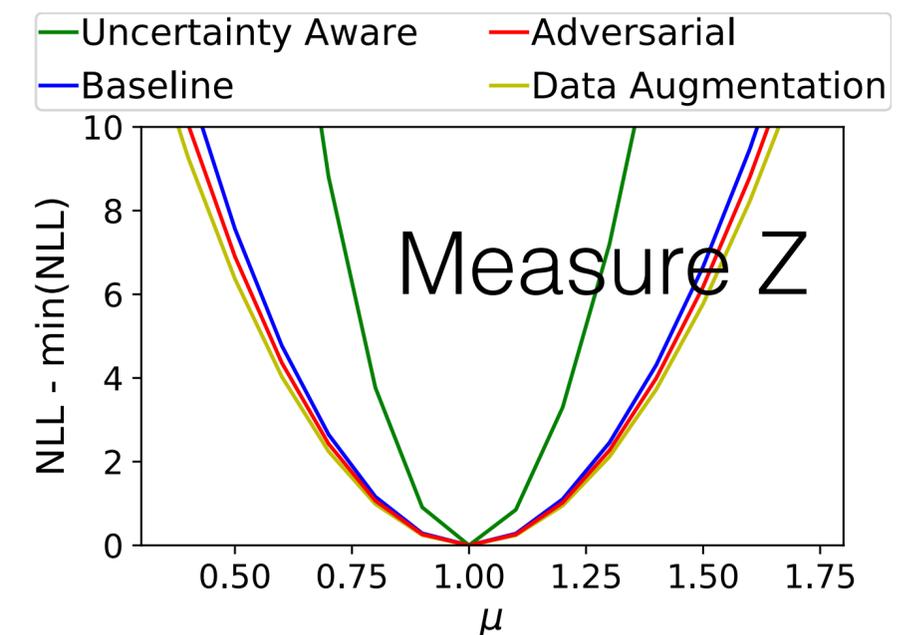
Aware classifier still best one to use

$$\begin{aligned}
 & -\log \mathcal{L}(\mu, z | \{x_i\}) \\
 &= -\sum_{j=1}^{n_{\text{bins}}} \left[N_j \cdot \log(\mu s_j + b_j) - \mu s_j - b_j - \log(\Gamma(N_j)) \right] \\
 & \quad - \sum_{k=1}^{m_{\text{bins}}} \left[N_k^{\text{aux}} \cdot \log(a_k^z) - a_k^z - \log(\Gamma(N_k^{\text{aux}})) \right],
 \end{aligned}$$

Baseline classifier trained on nominal

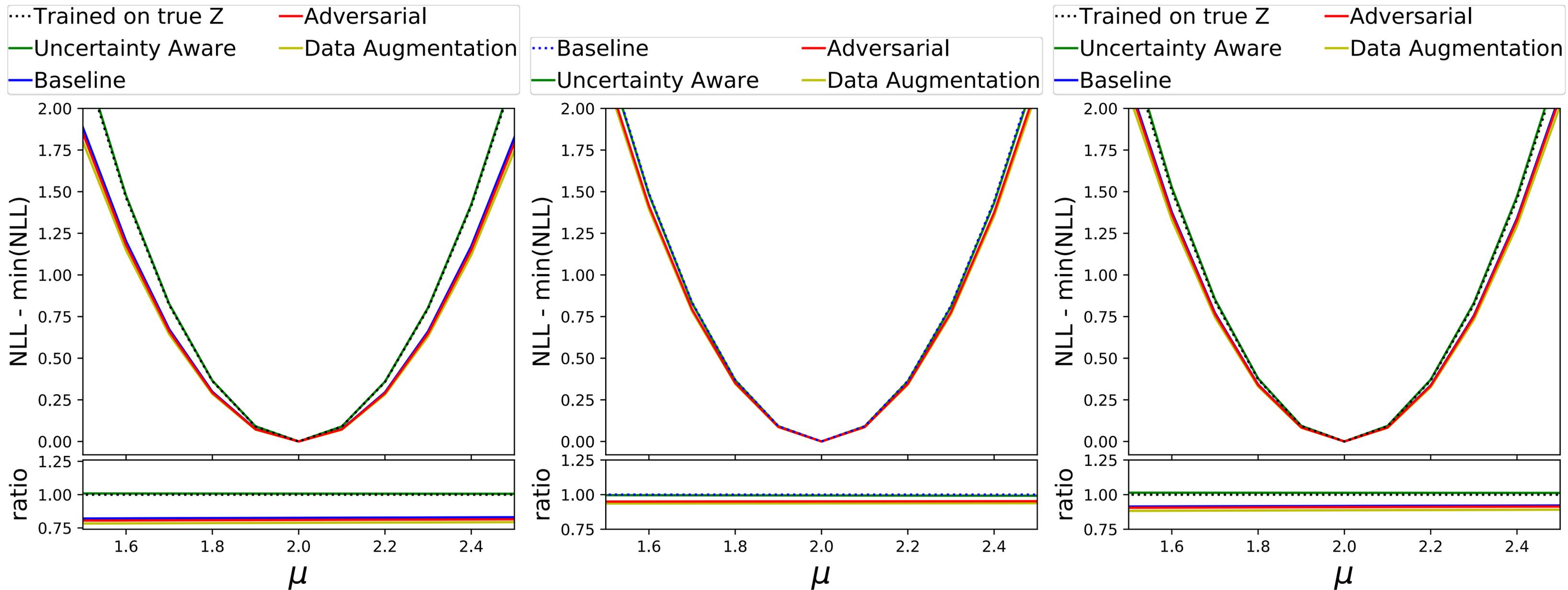


Now excludes wrong z confidently



(b) Data generated with $z = \frac{\pi}{2}$.

Test performance for “observed” datasets at $\mu = 2$



In every case the **Aware Classifier** is as good as the optimal one, no other technique matches its performance everywhere

Training Sub-Networks Individually

- Systematic effects are very subtle and often difficult to learn (effect of random noise much larger than systematic)
- A simple dense network parameterised on Z tends to overtrain before learning the full dependence on Z
- Solution: Train two subnetworks independently for $Z < 1$ and $Z \geq 1$ and combine with a if-else

```
def combineModels(ModelUp, ModelDown):
    input1 = Input(shape=(X_syst_train.shape[1],))
    input2 = Input(shape=(1,))

    selectModel1 = Lambda(lambda x: K.greater_equal(x, K.constant(1.)))(input2)
    selectModel2 = Lambda(lambda x: K.less(x, K.constant(1.)))(input2)

    selectModel1 = Lambda(lambda x: K.cast(x, dtype='float32'))(selectModel1)
    selectModel2 = Lambda(lambda x: K.cast(x, dtype='float32'))(selectModel2)

    out1 = Multiply()(ModelUp([input1, input2]), selectModel1)
    out2 = Multiply()(ModelDown([input1, input2]), selectModel2)

    out = Add()(out1, out2)
    model = Model(inputs=[input1, input2], outputs=out)
    return model

aware_model = combineModels(netAweUp_model, netAweDown_model)
aware_model.compile(optimizer='RMSProp',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])
```

Possible Extensions

- Uncertainty aware Bayesian Networks?
- Combine uncertainty awareness with inference awareness?
- How to deal with uncertainties when training directly on calo images / raw data ?
- ...