

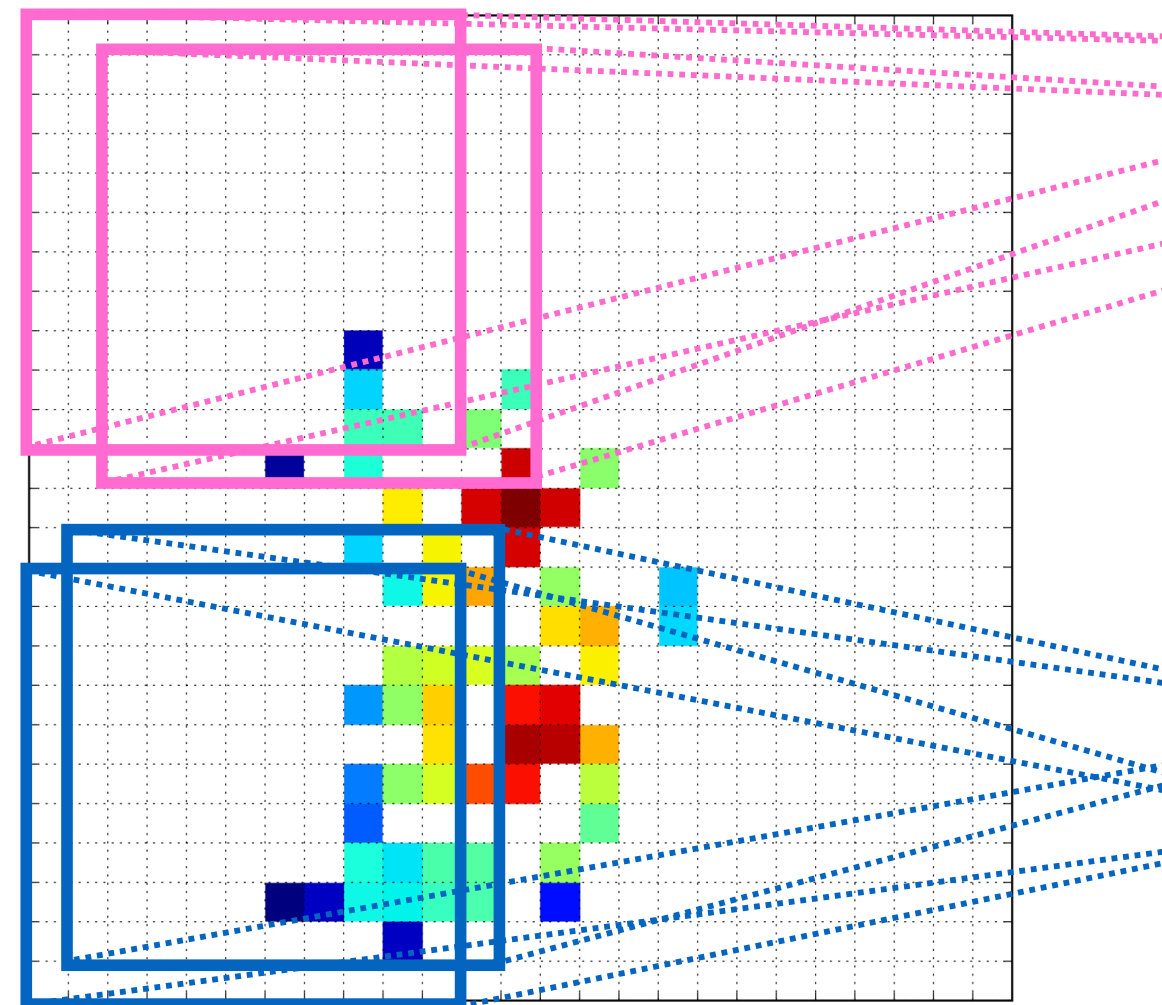
# Introduction to Machine Learning for Physics



Benjamin Nachman

*Lawrence Berkeley  
National Laboratory*

*ML4BSM  
April 3, 2018*



What this is not



What this is not  
*...the Higgs boson?*





The background is a dark, abstract composition of various geometric shapes, including cubes, rectangles, and lines. Some shapes are rendered in a 3D perspective, while others are flat. The color palette is primarily dark blues and greys, with accents of bright yellow and green. A prominent feature is a series of yellow lines radiating from a central point, resembling a starburst or a network diagram. There are also some green rectangular shapes scattered throughout the scene.

**What this is not**

***A replacement for a  
great online tutorial or  
a university course***

***Love ROOT? Try TMVA***

***Want a ROOT-alternative that  
does everything except deep  
learning? Try scikit-learn***

***Want to give deep learning a try?  
Try Keras***

***...***

# What is Machine Learning?



Run: 302347

Event: 753275626

2016-06-18 18:41:48 CEST



# What is Machine Learning?

**Answer: just about everything we do!**

**...algorithms for identifying and analyzing structure in data**

*What can we use machine learning for?*

## **Supervised learning**

Classification

Regression

Generation\*

the machine is  
presented examples of  
multiple classes and  
learns to differentiate

## **Unsupervised learning**

Clustering

Anomaly detection

the machine is  
presented data and  
asked to give you  
multiple classes

*\*Depending on your point of view, this may also be considered unsupervised.*

*What can we use machine learning for?*

## Supervised learning

### **Classification**

Regression

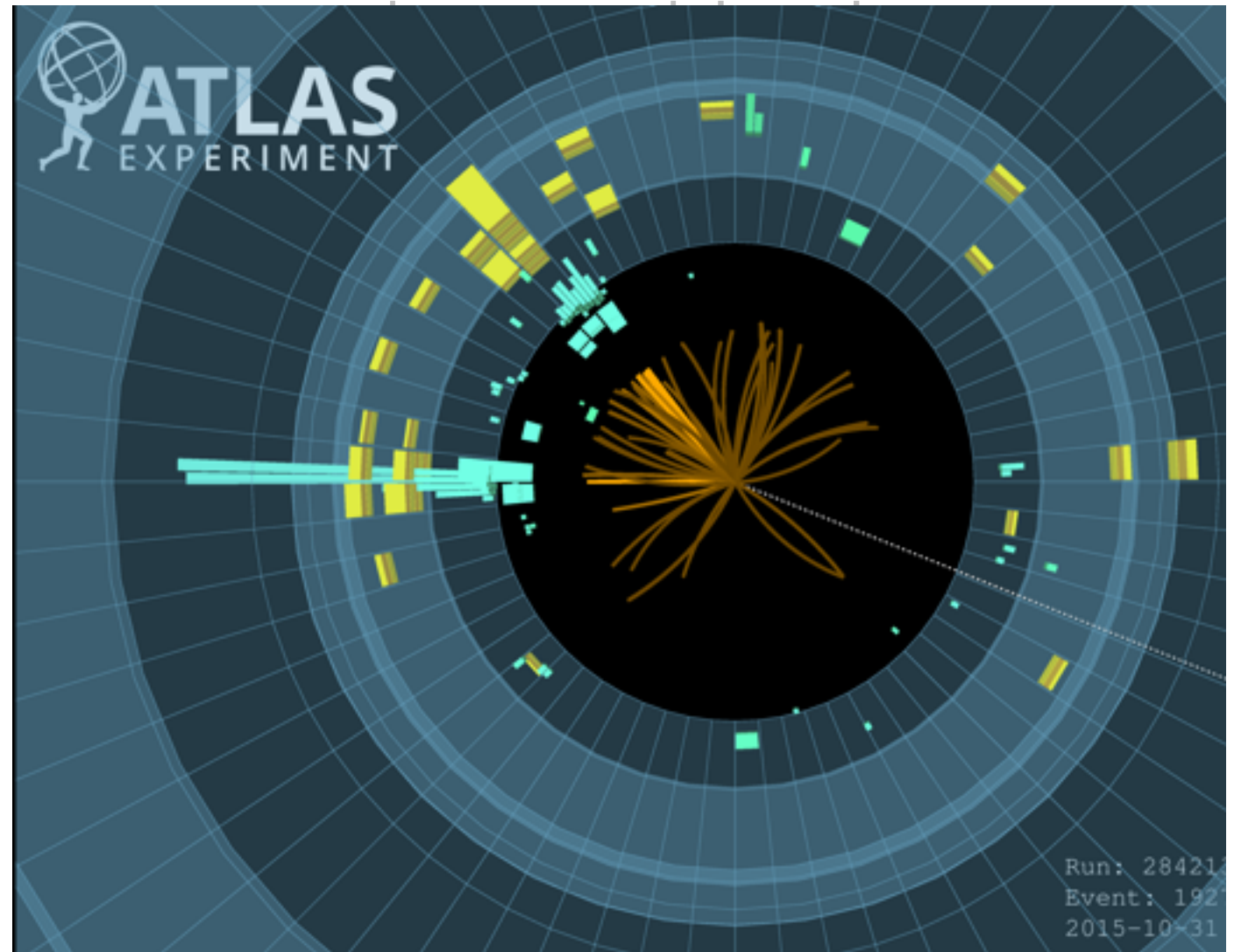
Generation

## Unsupervised learning

Clustering

Anomaly detection

## Higgs boson or gluon?



multiple classes



*What can we use machine learning for?*

## Supervised learning

Classification

**Regression**

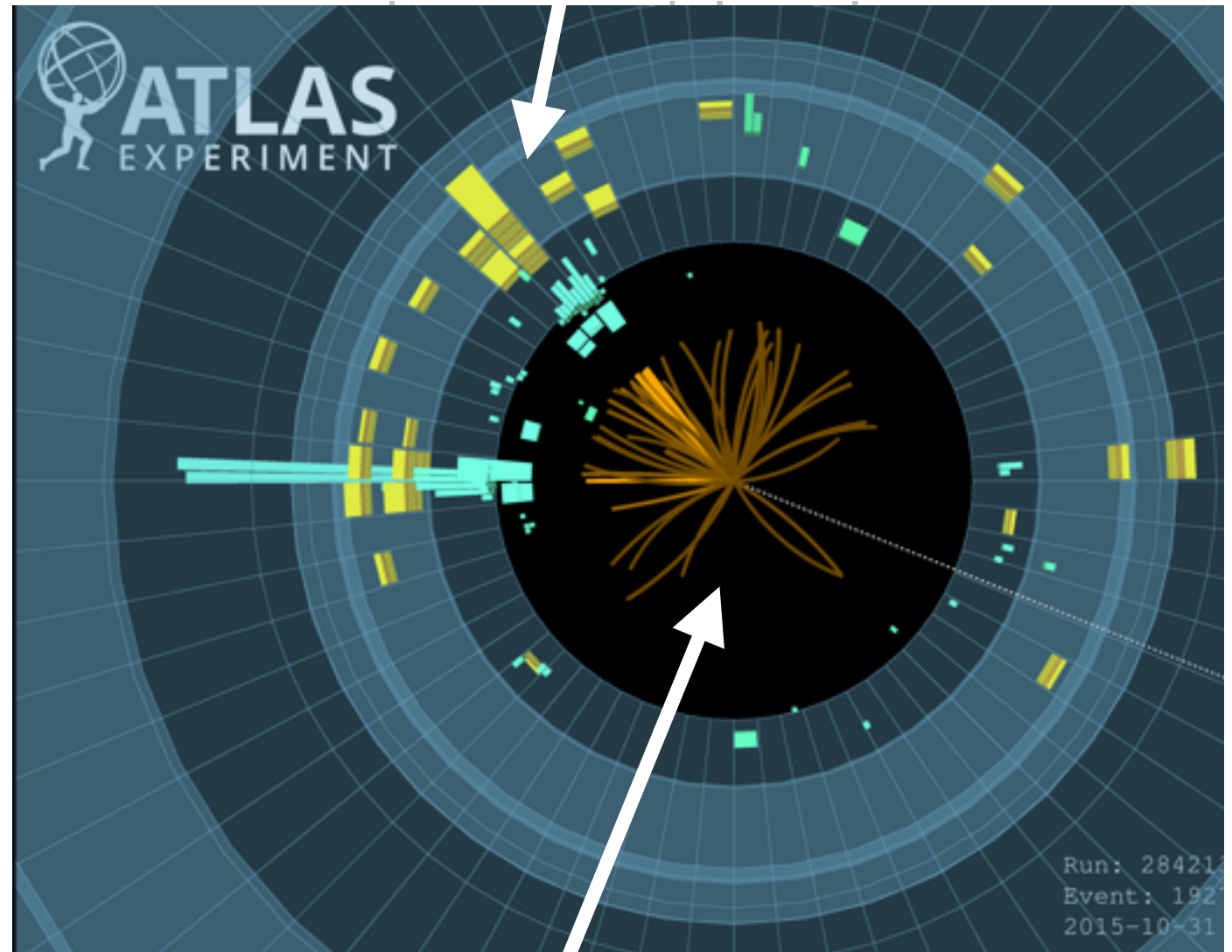
Generation

## Unsupervised learning

Clustering

Anomaly detection

**What is the energy of this spray of particles (jet)?**



**What are the momenta of these charged particles?**

*What can we use machine learning for?*

## Supervised learning

Classification

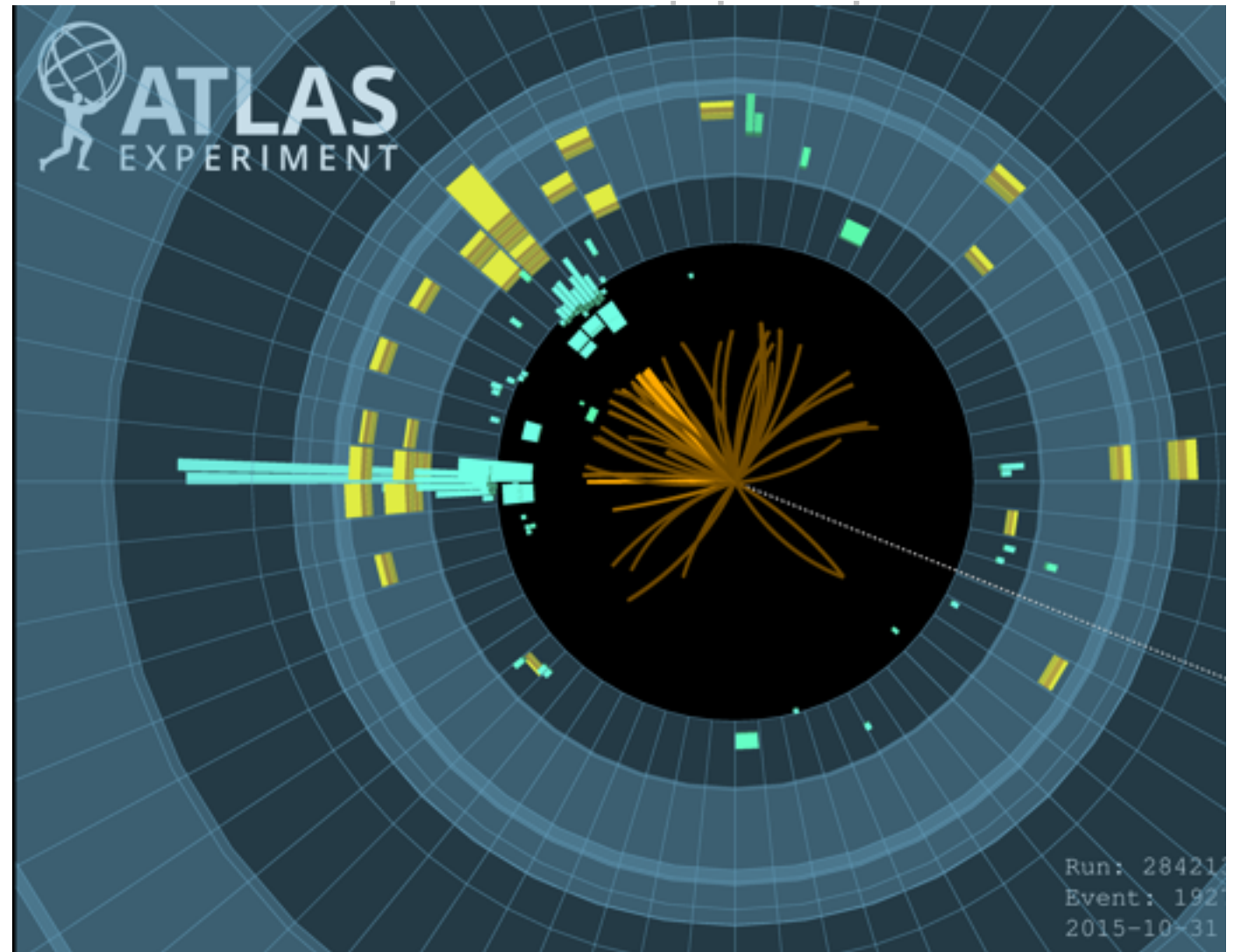
Regression

**Generation**

## Unsupervised learning

Clustering

Anomaly detection



multiple classes

**What would Higgs boson events look like with a different mass?**

What can we use machine learning for?

## Supervised learning

Classification

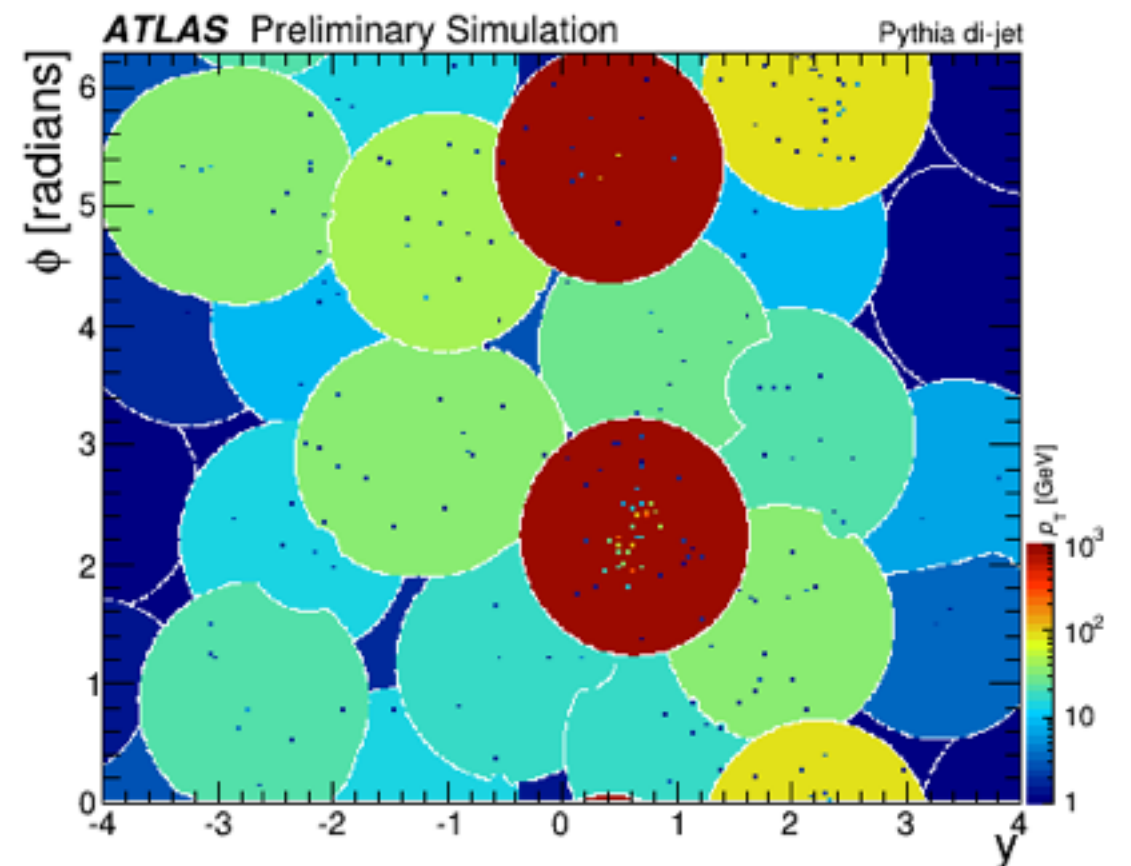
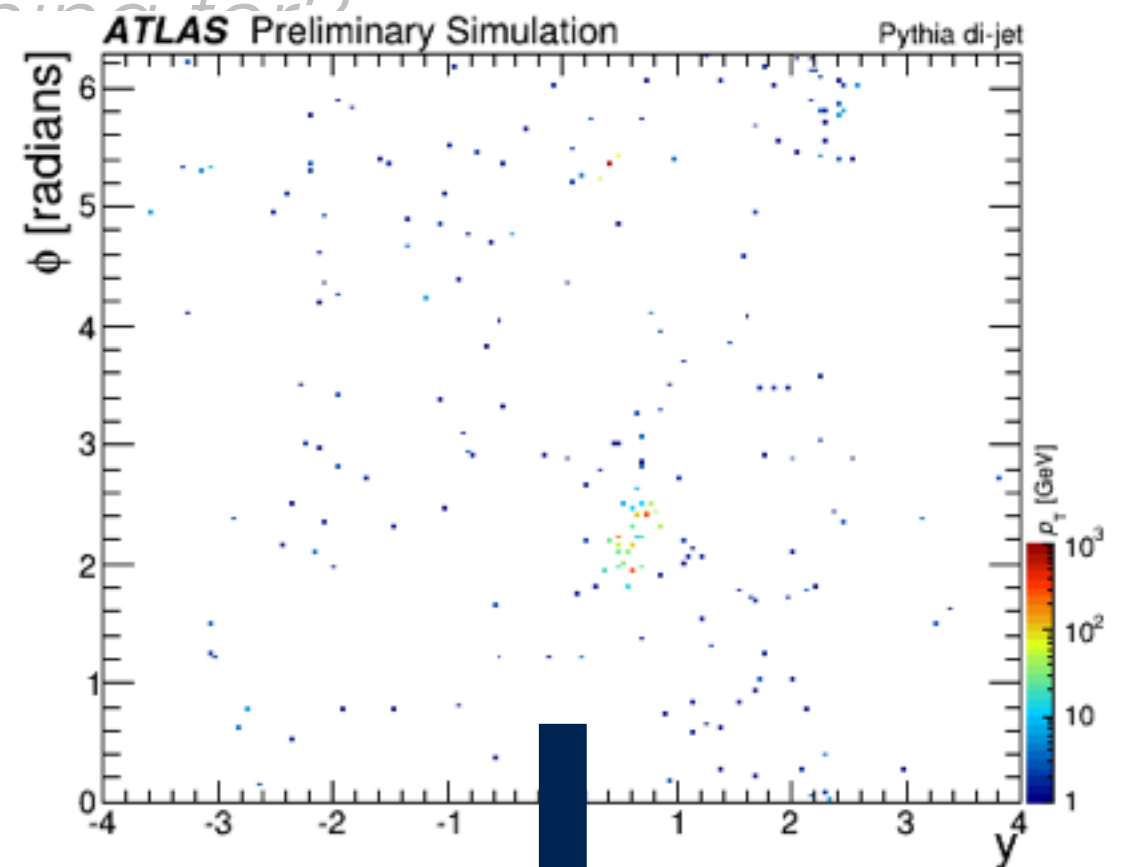
Regression

Generation

## Unsupervised learning

### Clustering

Anomaly detection





# What can we use machine learning for?

## Supervised learning

Classification

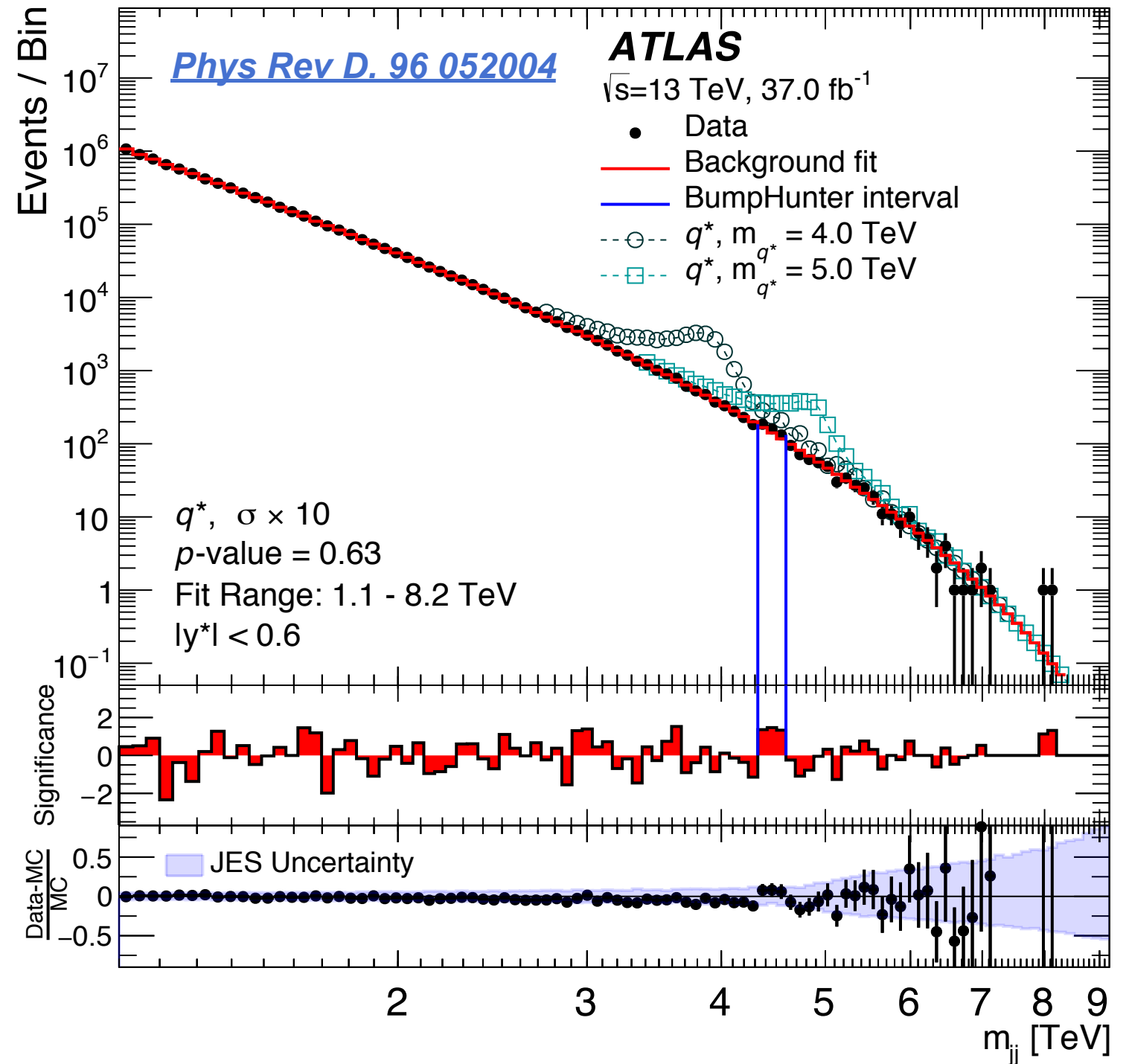
Regression

Generation

## Unsupervised learning

Clustering

## Anomaly detection



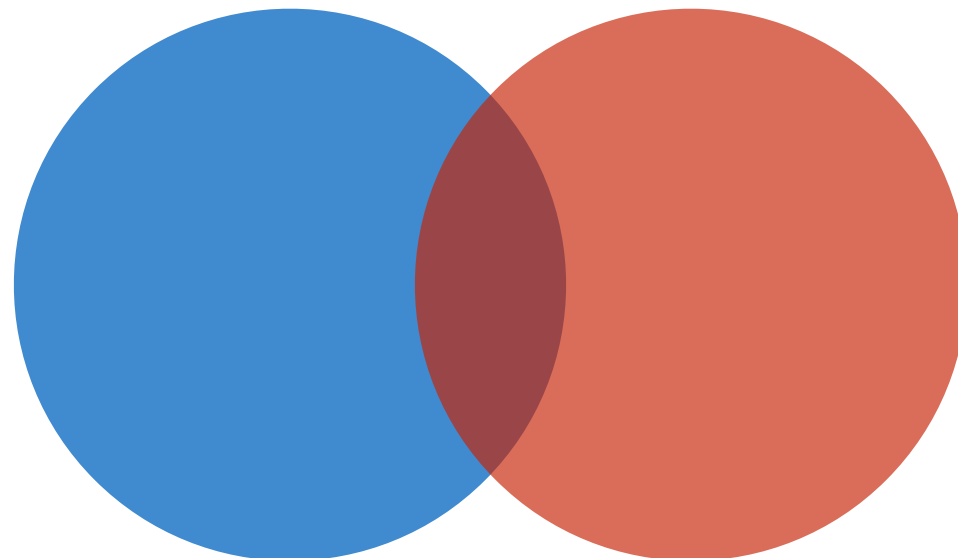
# Classification

Goal: Given a *feature vector*, return an integer indexed by the set of possible *classes*.

In most cases, we care about *binary* classification in which there are only two classes (signal versus background)

There are some cases where we care about *multi-class classification*

Feature vector  
can be many-  
dimensional



Harder = more  
overlap between  
for **S** and **B**

# Classification

Goal: Given a *feature vector*, return an integer indexed by the set of possible *classes*.

In practice, we don't just want one classifier, but an entire set of classifiers indexed by:

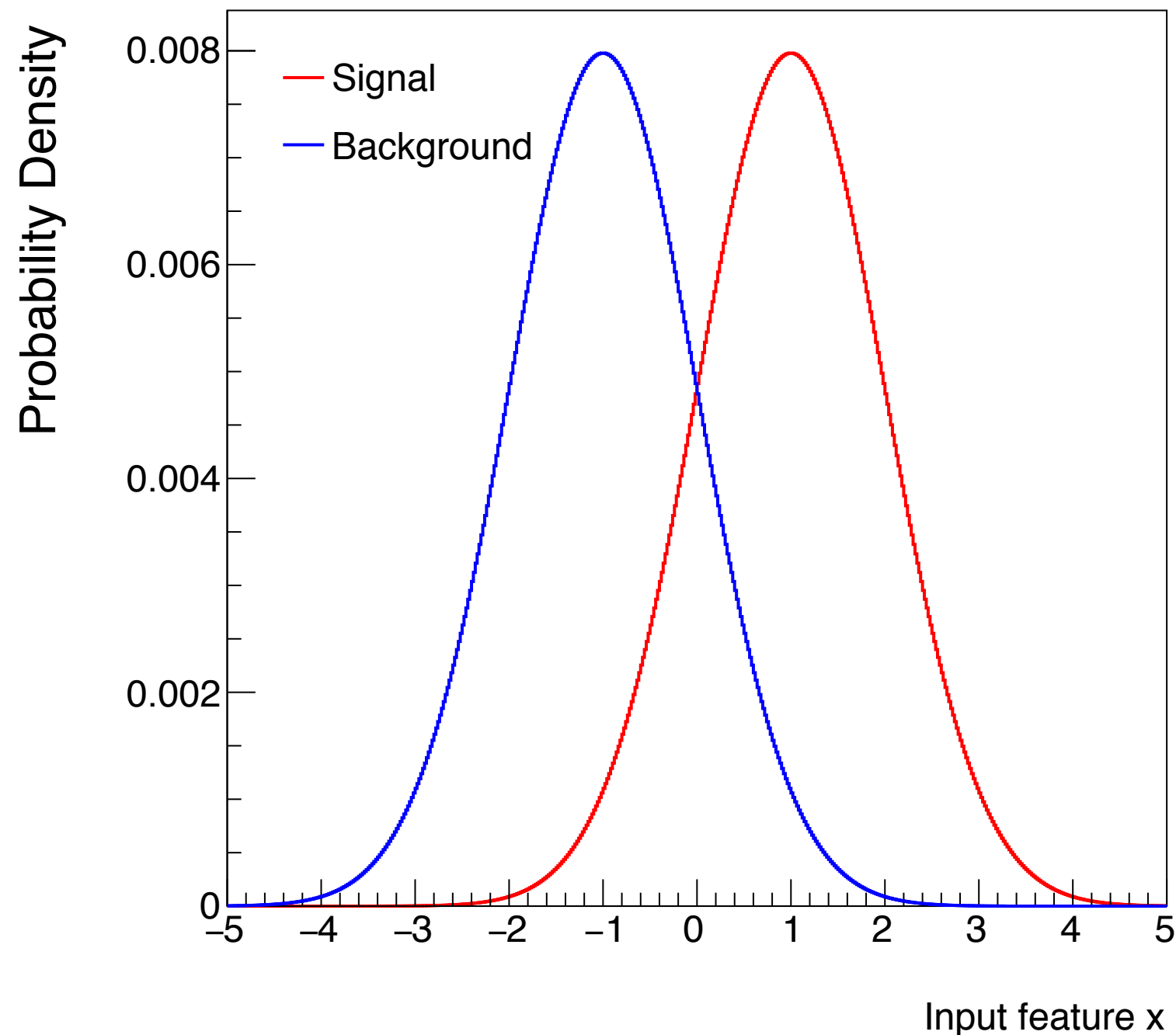
**True Positive Rate** = signal efficiency =  
 $\Pr(\text{label signal} \mid \text{signal}) = \text{sensitivity}$

**True Negative Rate** = 1 - background efficiency =  
rejection =  $\Pr(\text{label background} \mid \text{background}) = \text{specificity}$

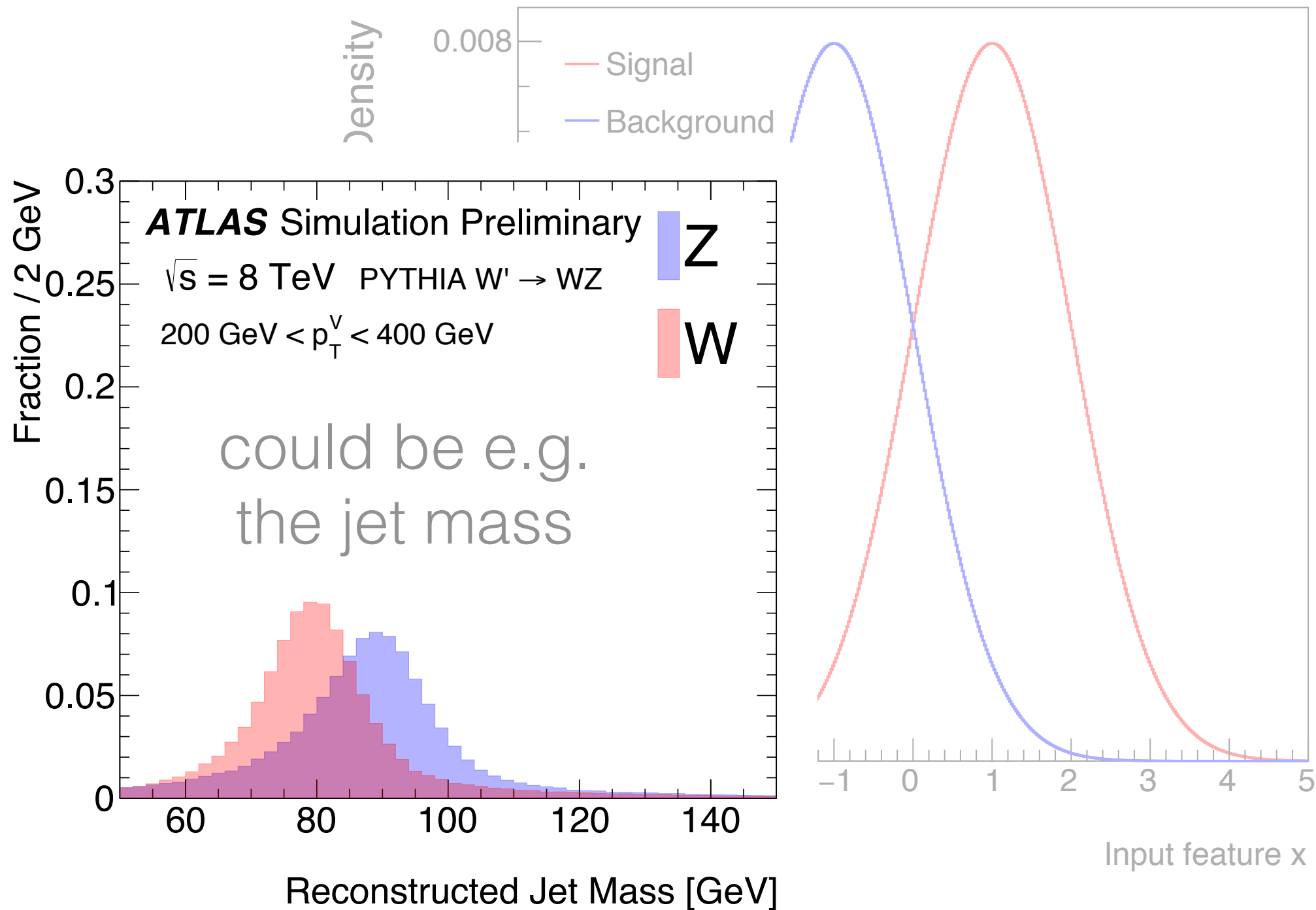
*For a given TPR, we want the lowest possible TNR!*



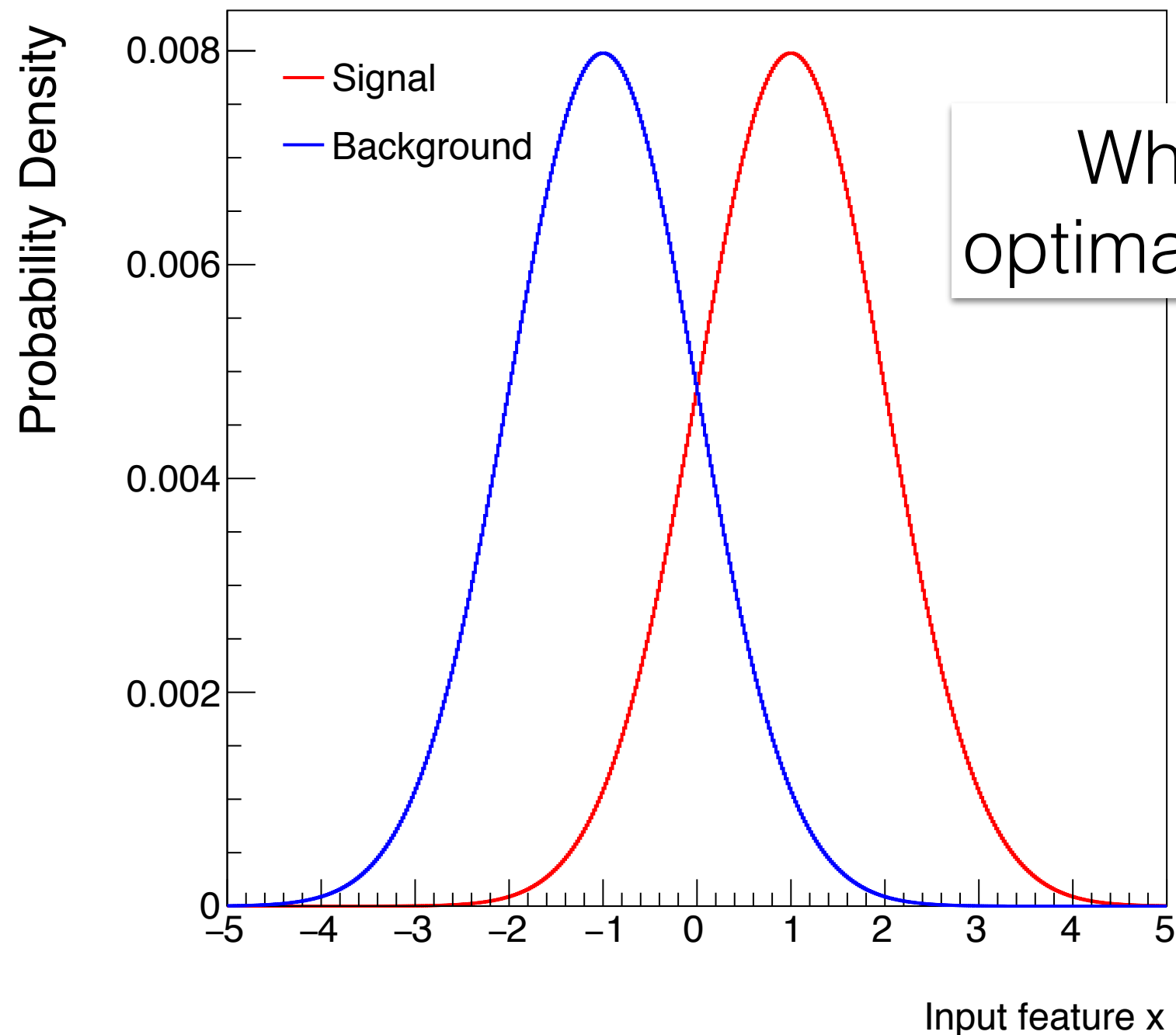
Let's consider an important special case:  
binary classification in 1D



Let's consider an important special case:  
binary classification in 1D



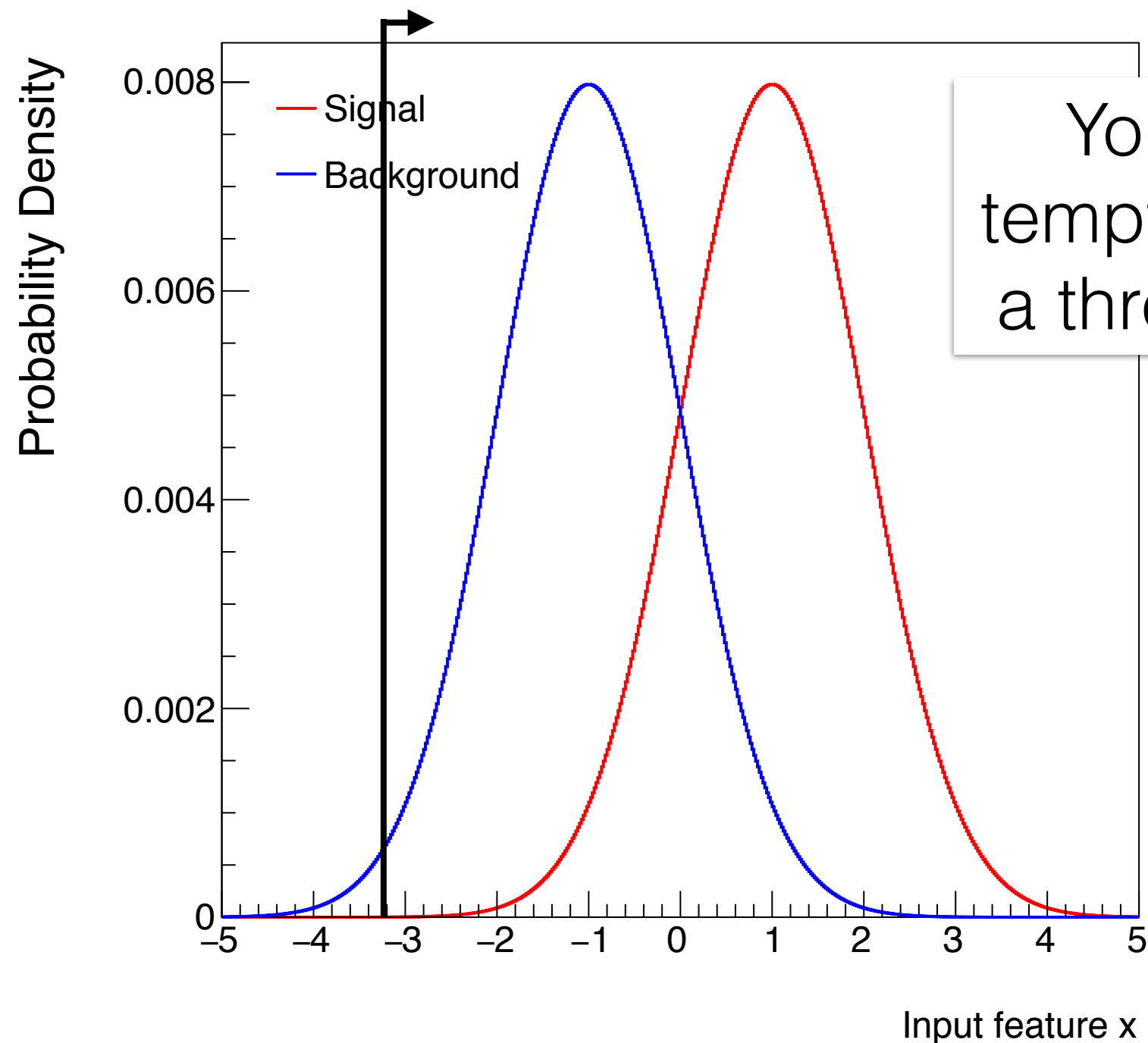
Let's consider an important special case:  
binary classification in 1D



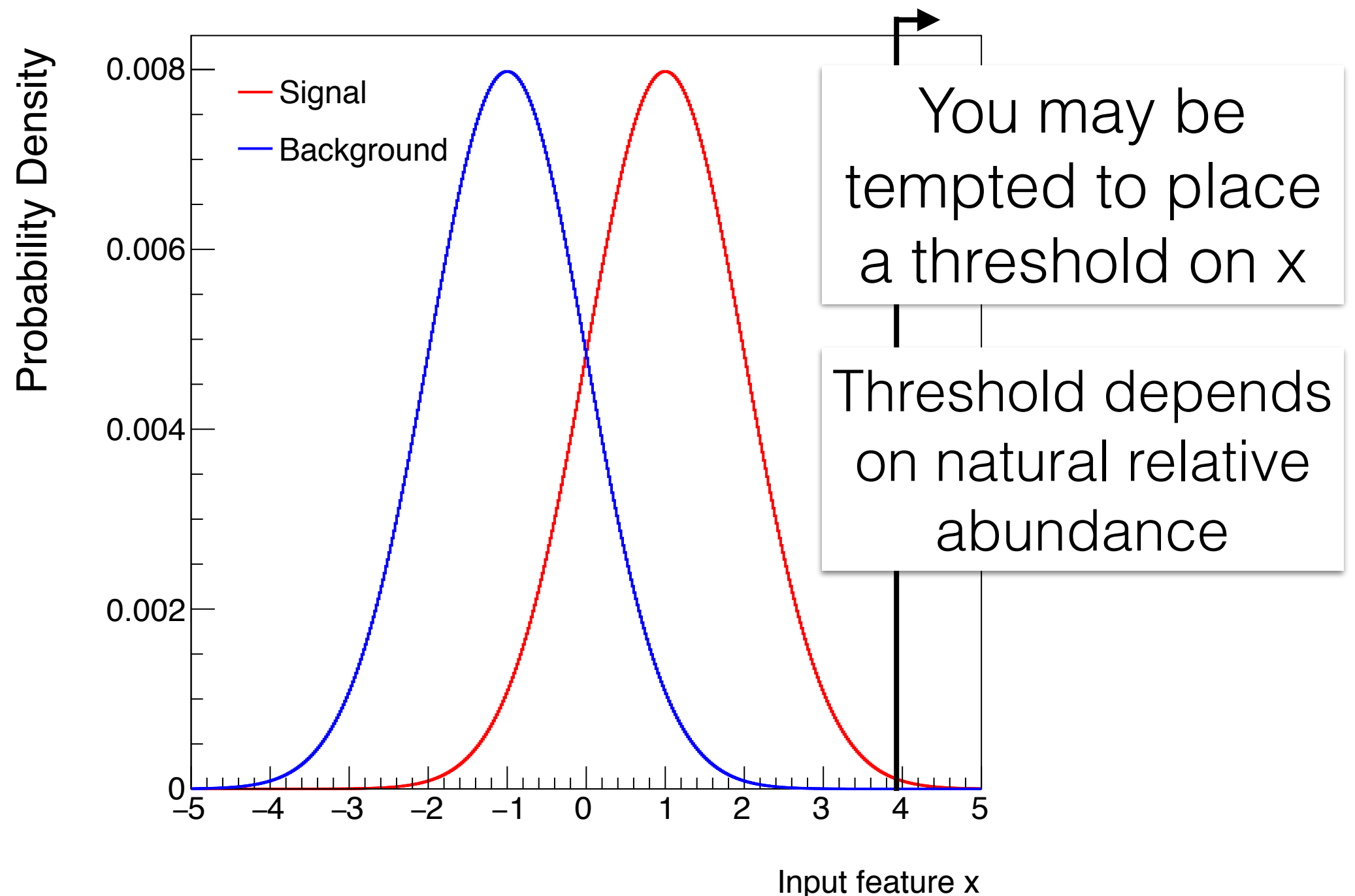
What is the  
optimal classifier?

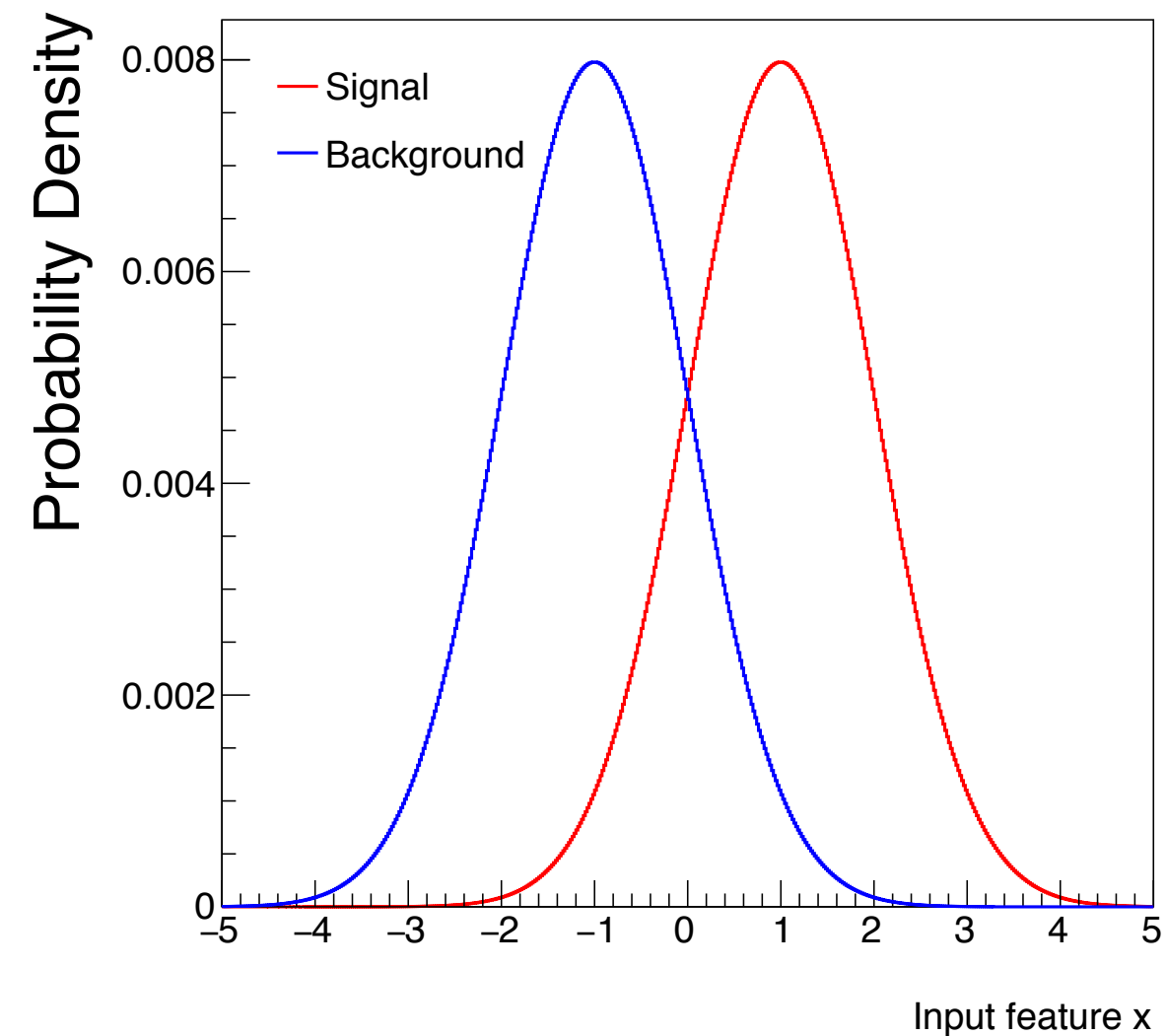


Let's consider an important special case:  
binary classification in 1D

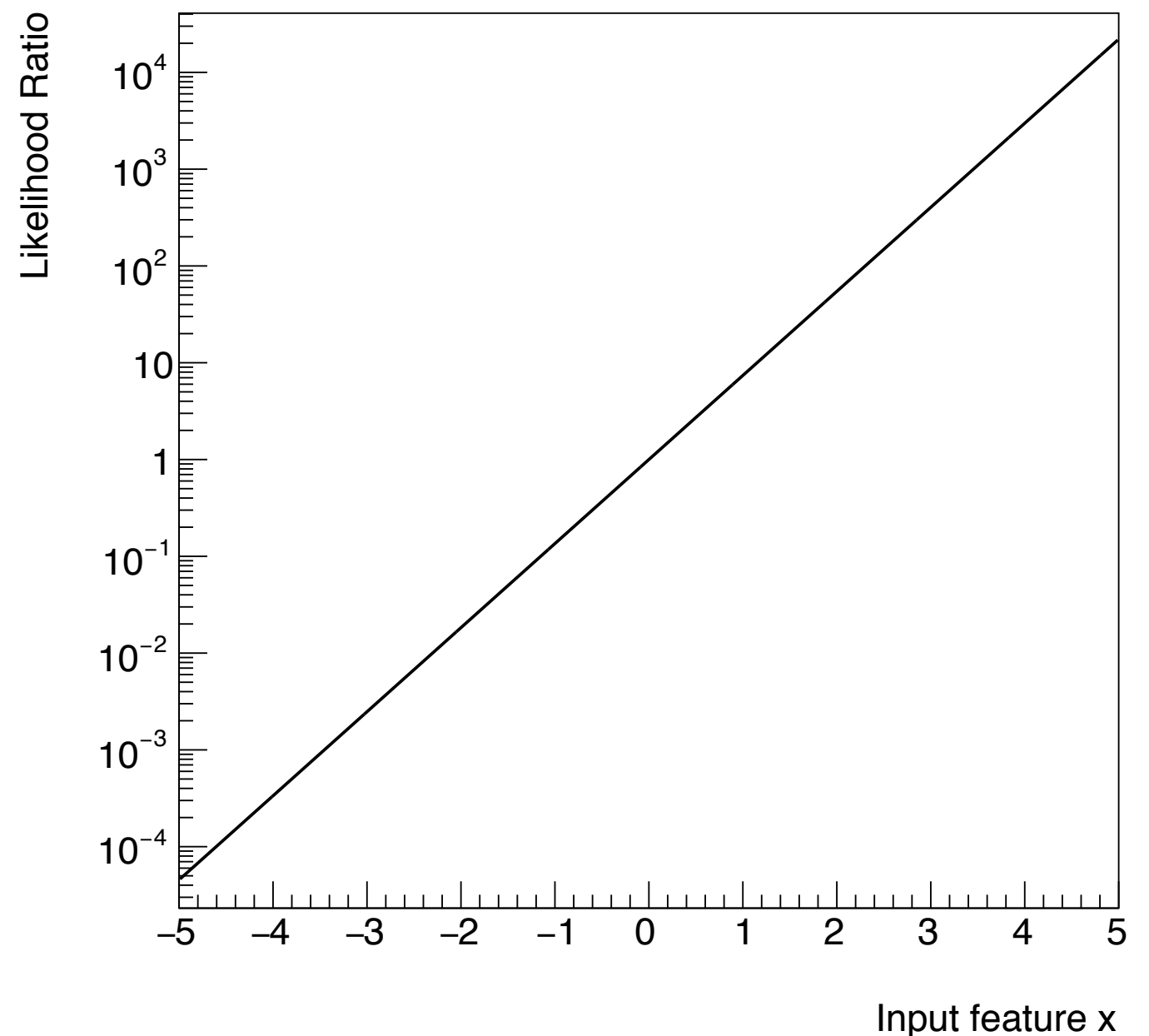


Let's consider an important special case:  
binary classification in 1D





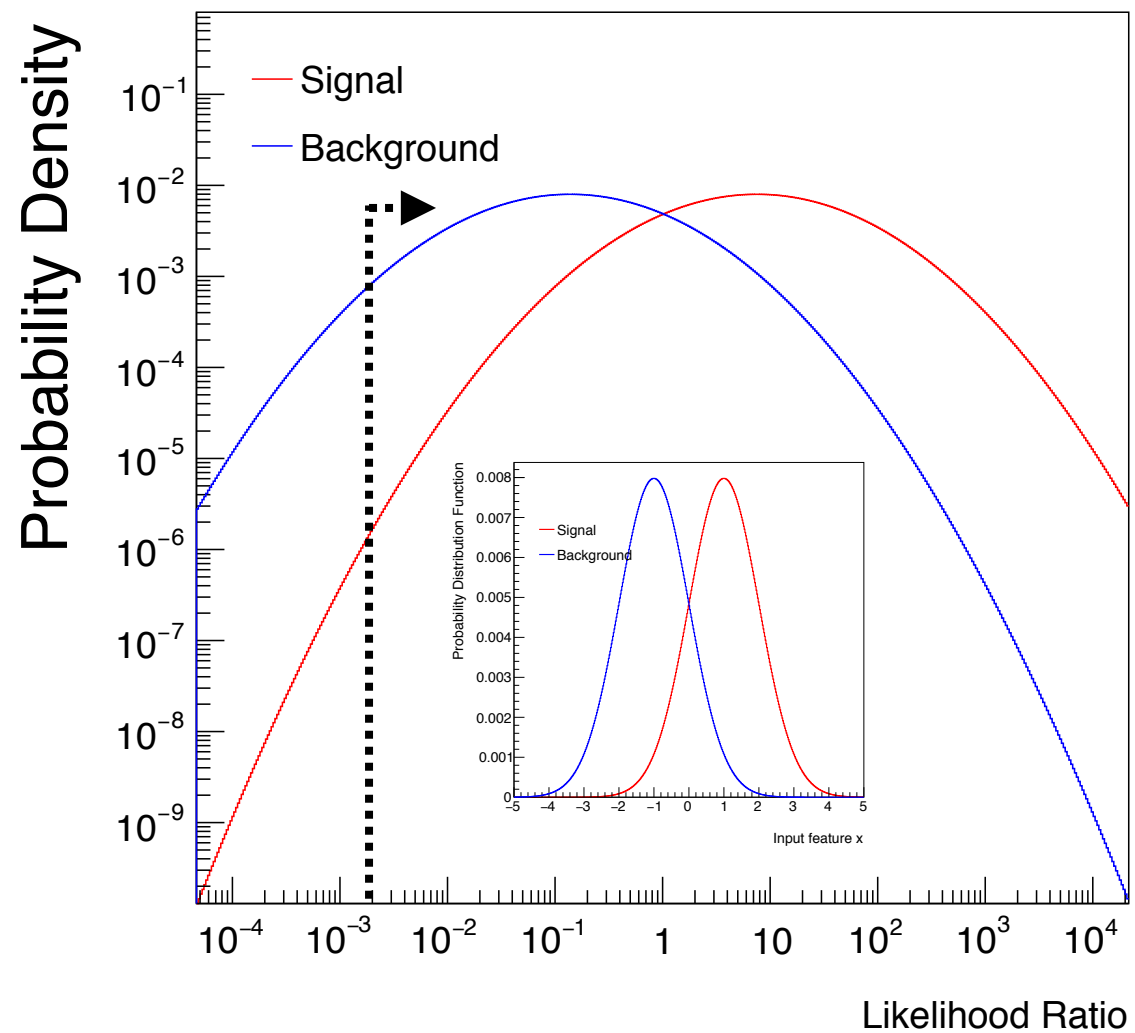
Is the simple  
threshold cut optimal?



In this simple case, the log  
LL is proportional to x:  
**no need for non-linearities!**

*Threshold cut is optimal*

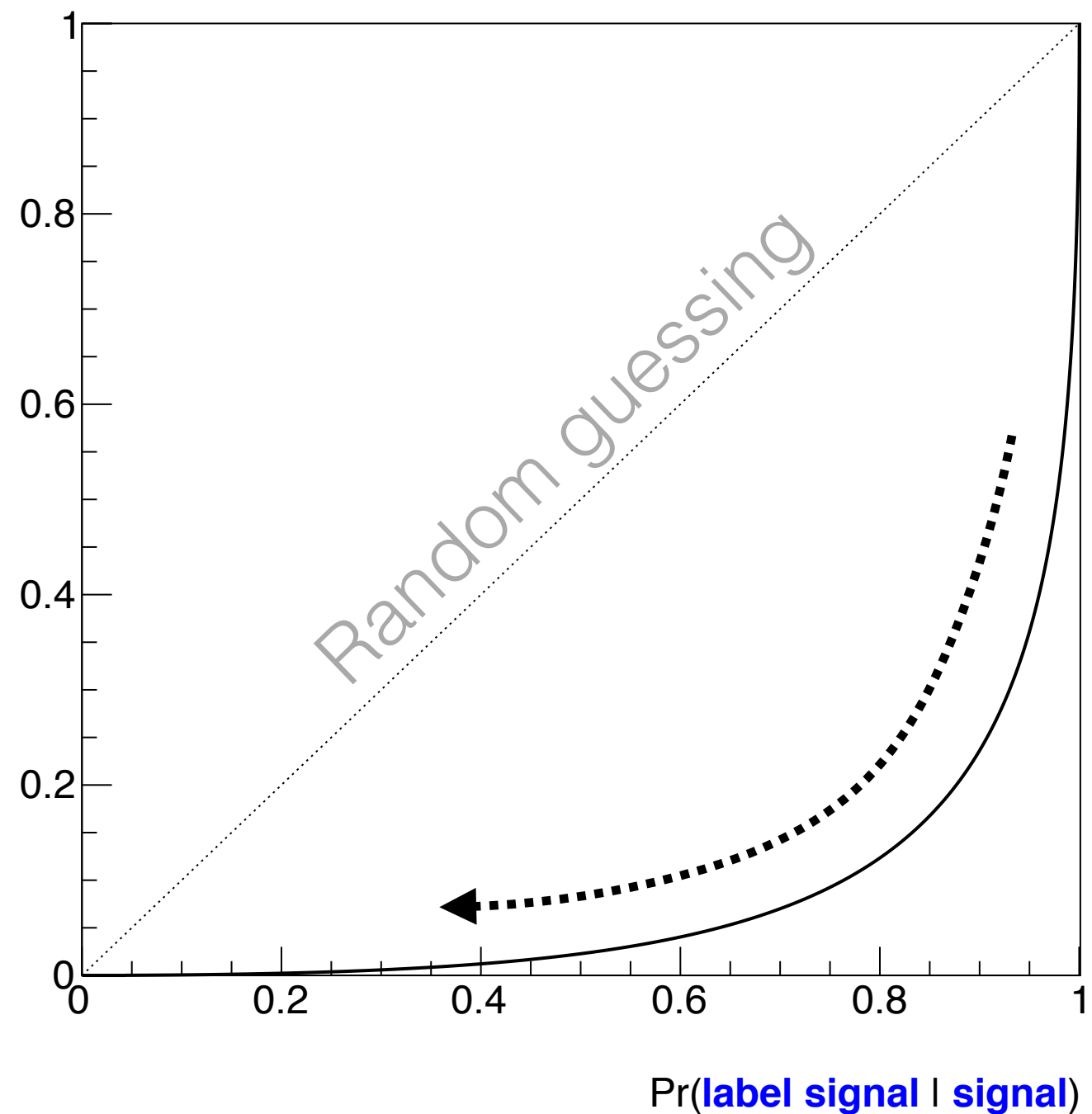




“Receiver Operating Characteristic” (**ROC**) Curve

The optimal procedure is a threshold on the LL

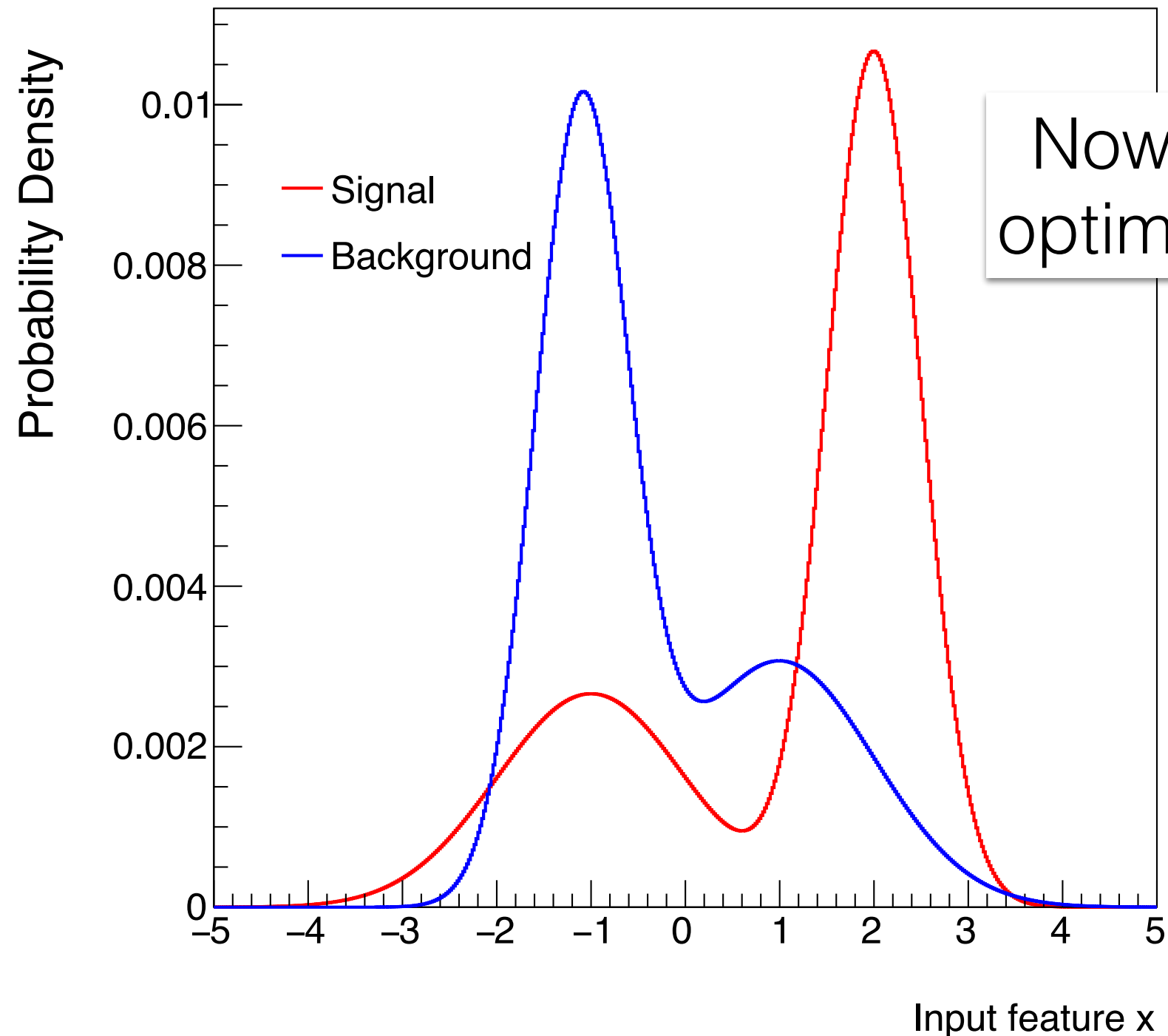
$\Pr(\text{label signal} \mid \text{background})$



$\Pr(\text{label signal} \mid \text{signal})$

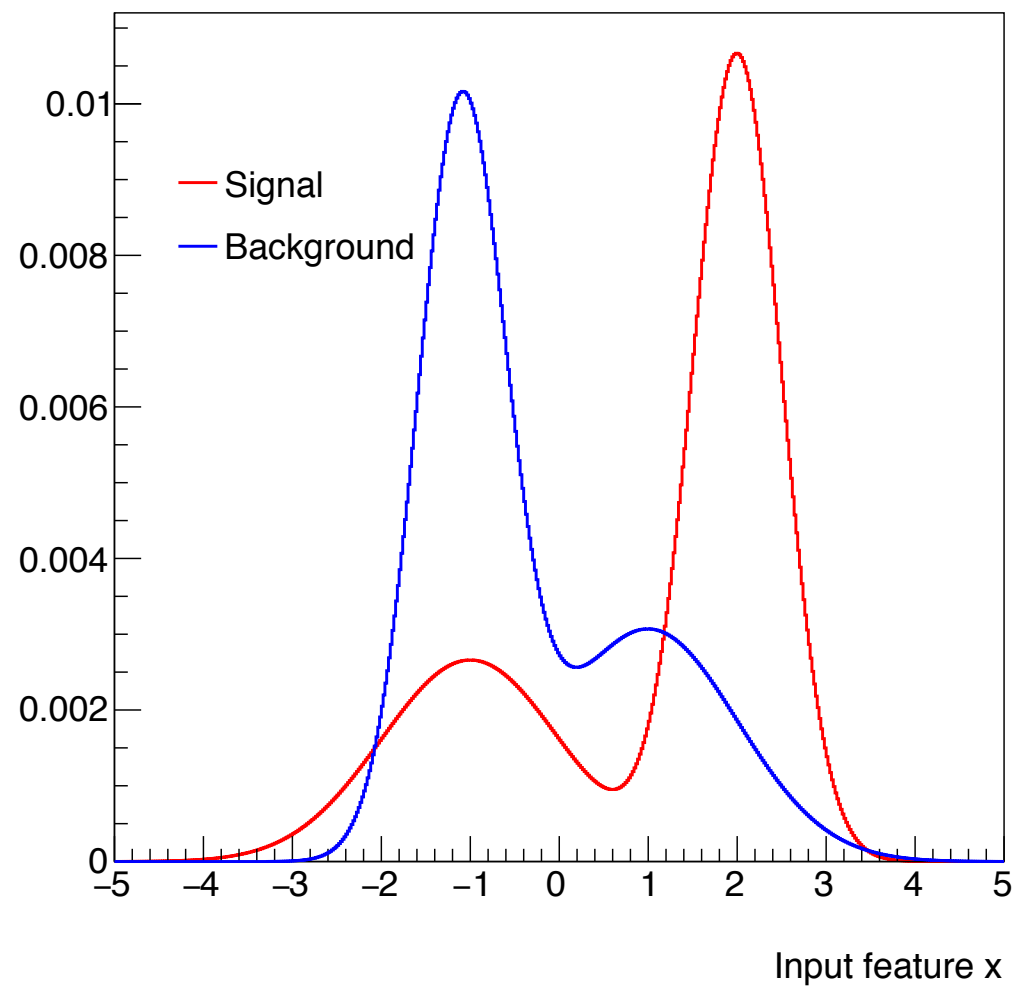
What if the distribution of  $x$  is complicated?

Real life is complicated!



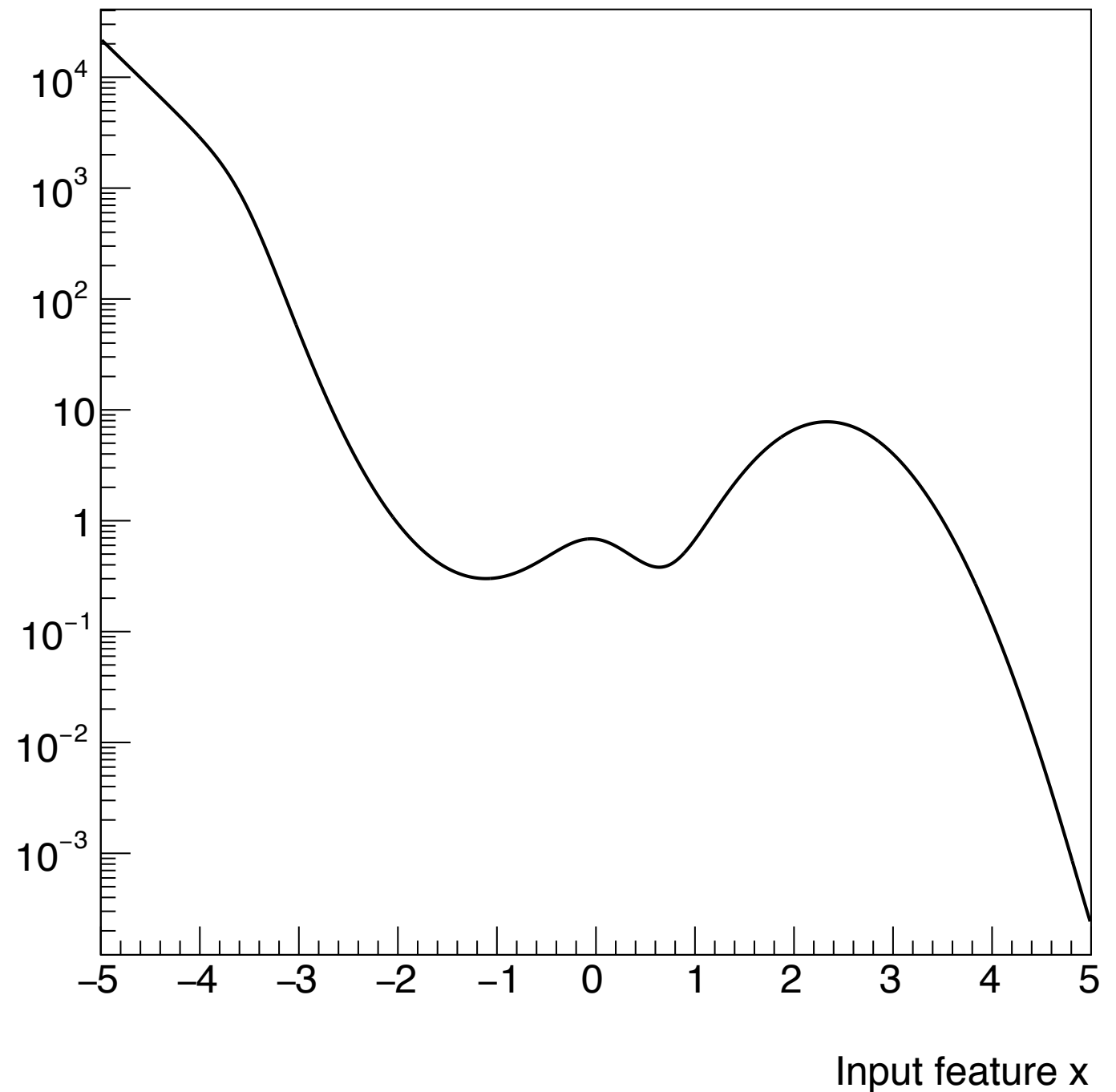
Now what is the optimal classifier?

Probability Density



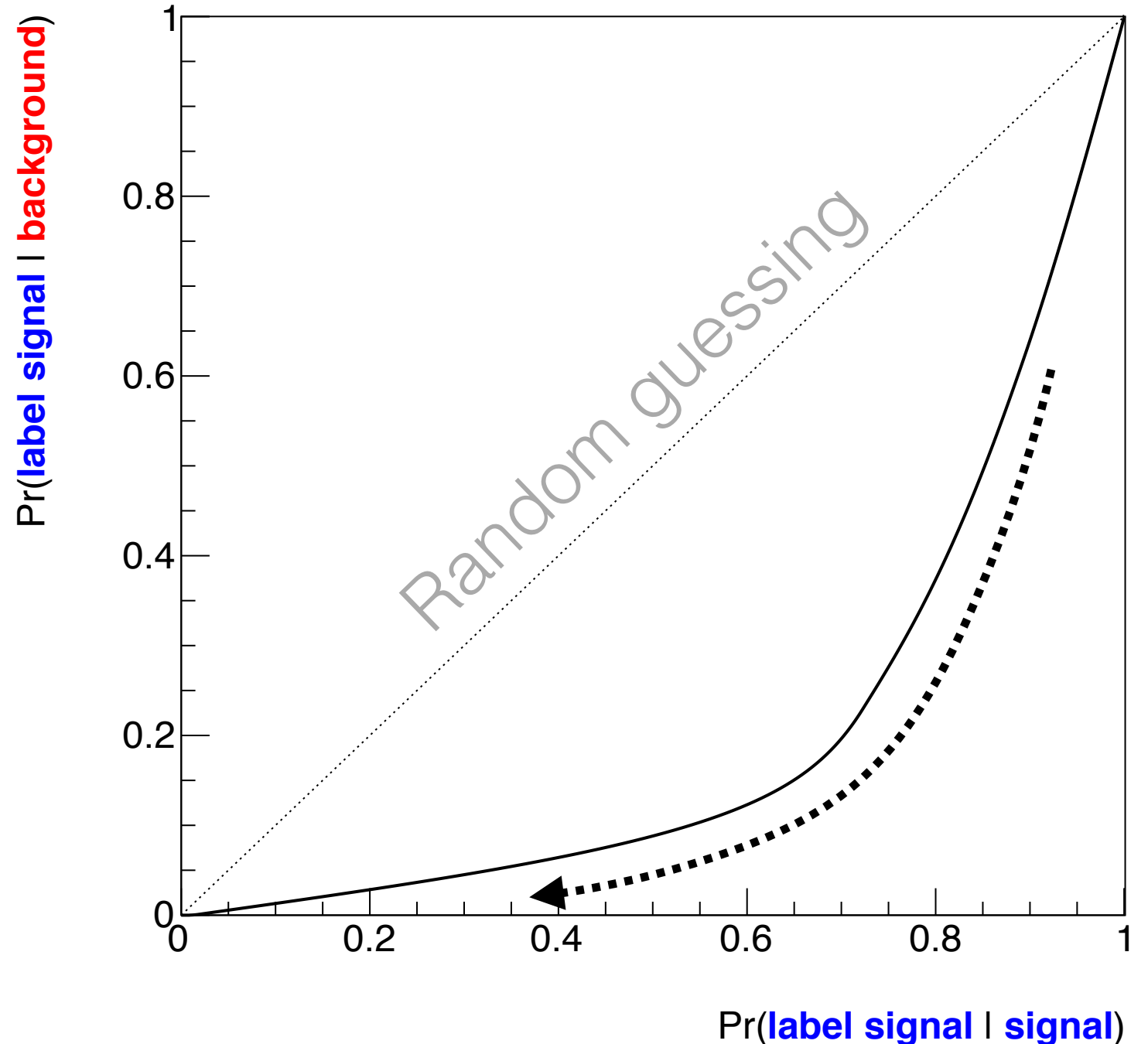
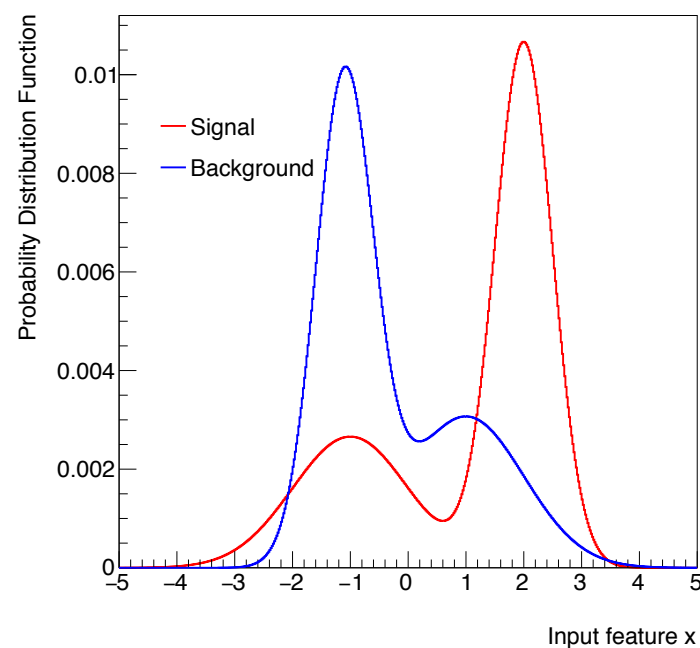
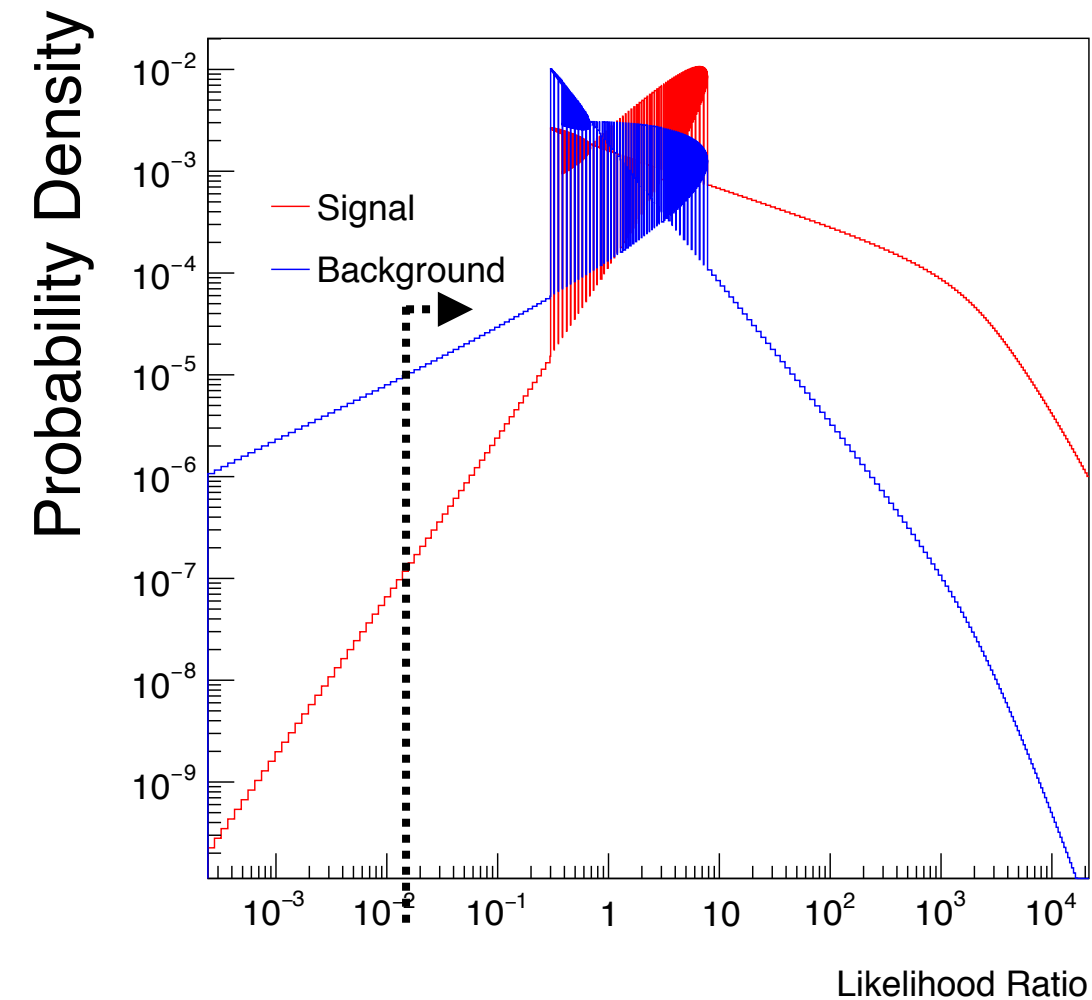
In this case, LL is highly non-linear  
(**non-monotonic**) function of  $x$

Likelihood Ratio



A threshold on  $x$   
would be sub-optimal

ROC is worse than the Gaussians,  
but that is expected since the  
overlap in their PDFs is higher.



Why don't we always just  
compute the optimal classifier?

In the last slides, we had to estimate the  
likelihood ratio - this required binning the PDF

binning works very well in 1D, but becomes  
quickly intractable as the feature vector  
dimension  $\gg 1$  ("curse of dimensionality")

machine learning for classification is simply  
**the art of estimating the likelihood ratio  
with limited training examples**



# Tools for Classification

=tools for likelihood ratio estimation

- “Histograming”
- Nearest Neighbors
- Support Vector Machines (SVM)
- (Boosted) Decision Trees
- (Deep) Neural Networks
- ...

Not widely used; only  
useful if decision  
boundary is ‘simple’

has most things and ROOT-compatible but the  
community base is **much** smaller than the other ones

Software: TMVA, scikit-learn, keras, ...

does “everything” exempt DNNs

python interface  
to DNN tools  
TensorFlow,  
Theano, CNTK

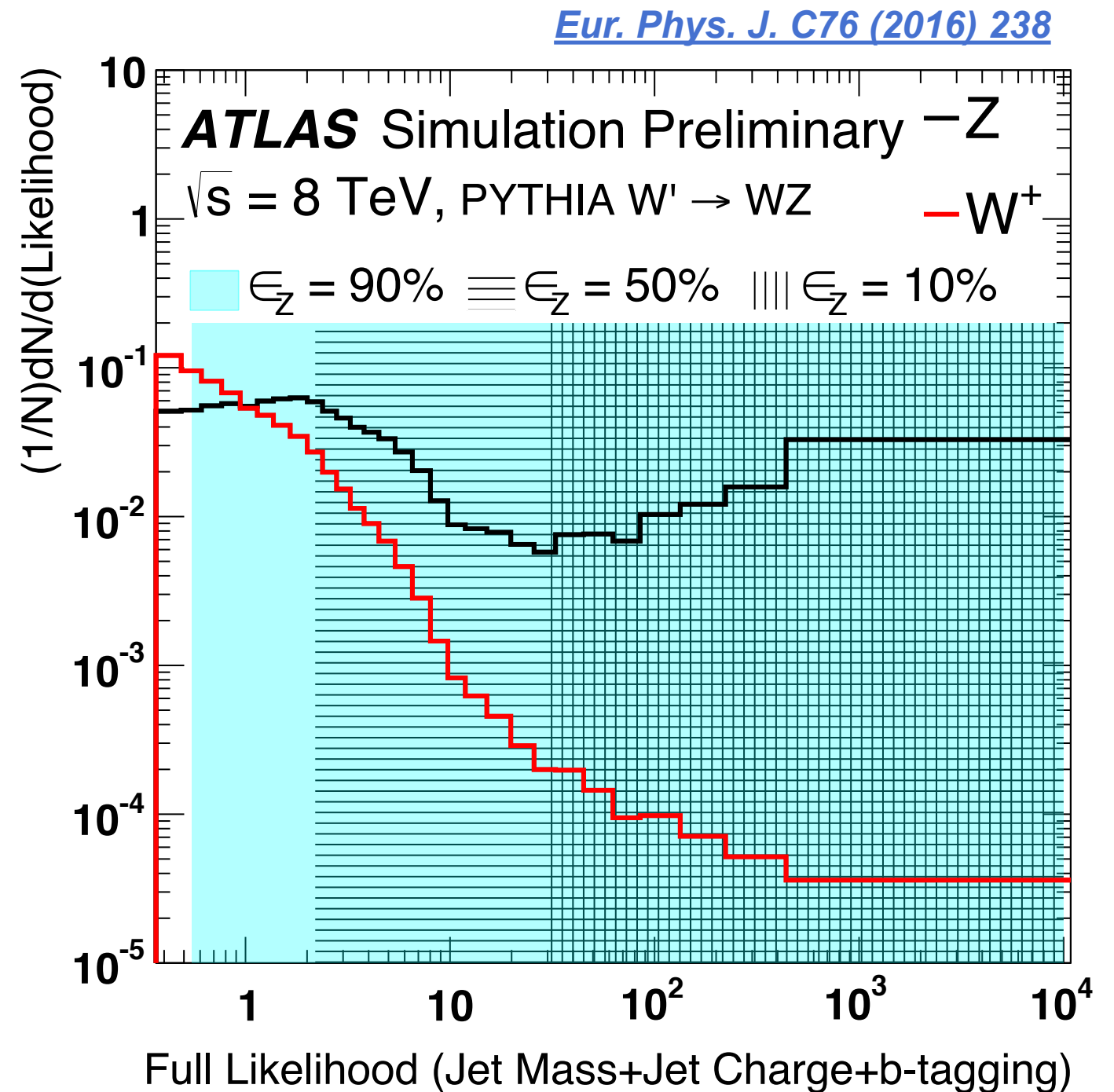
Data formats: .root, .npy, .hdf5

# Histogramming

If you have a 1D problem, look no further!

If your problem can be decomposed into a product/sum of 1D problems...look no further!

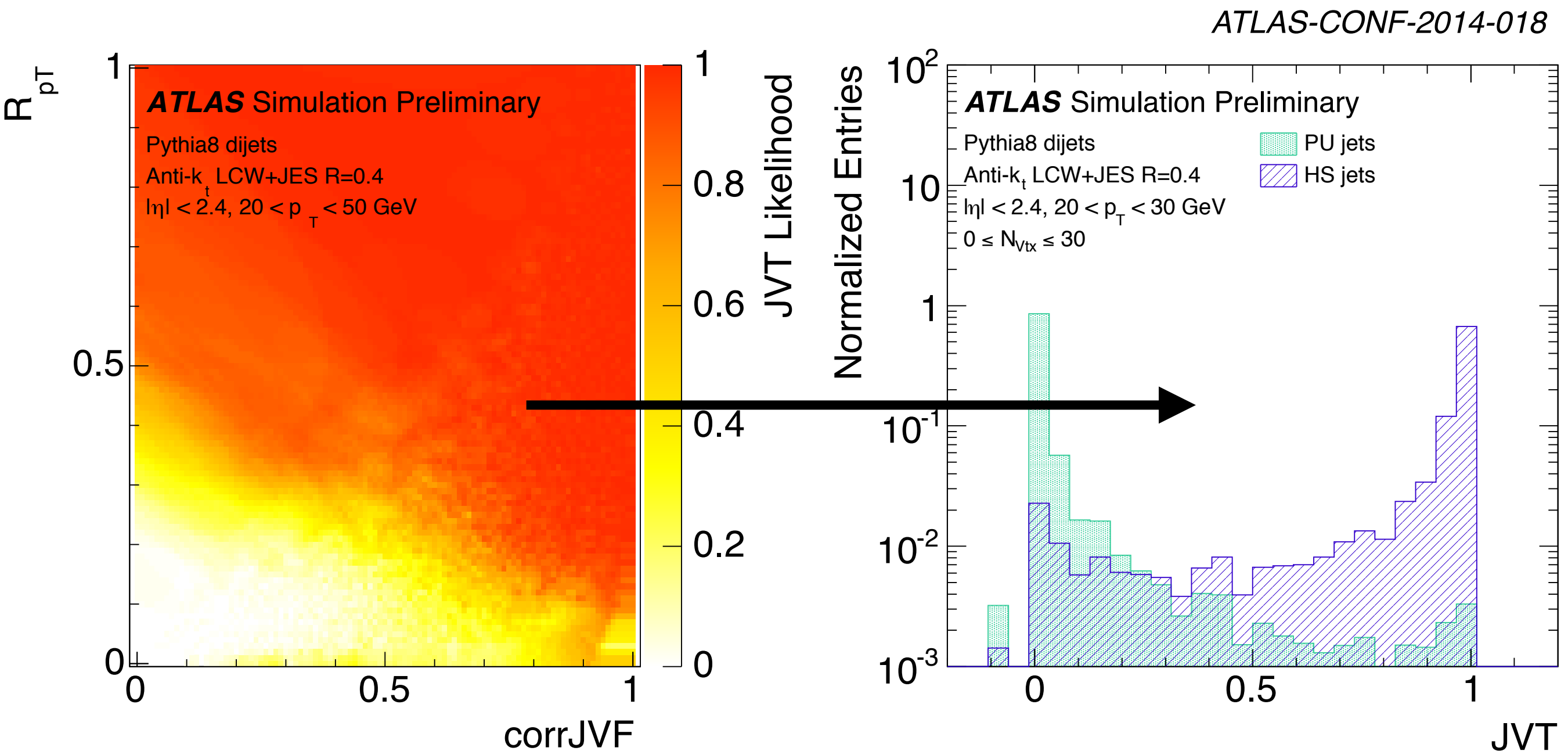
If these do not apply...look elsewhere.



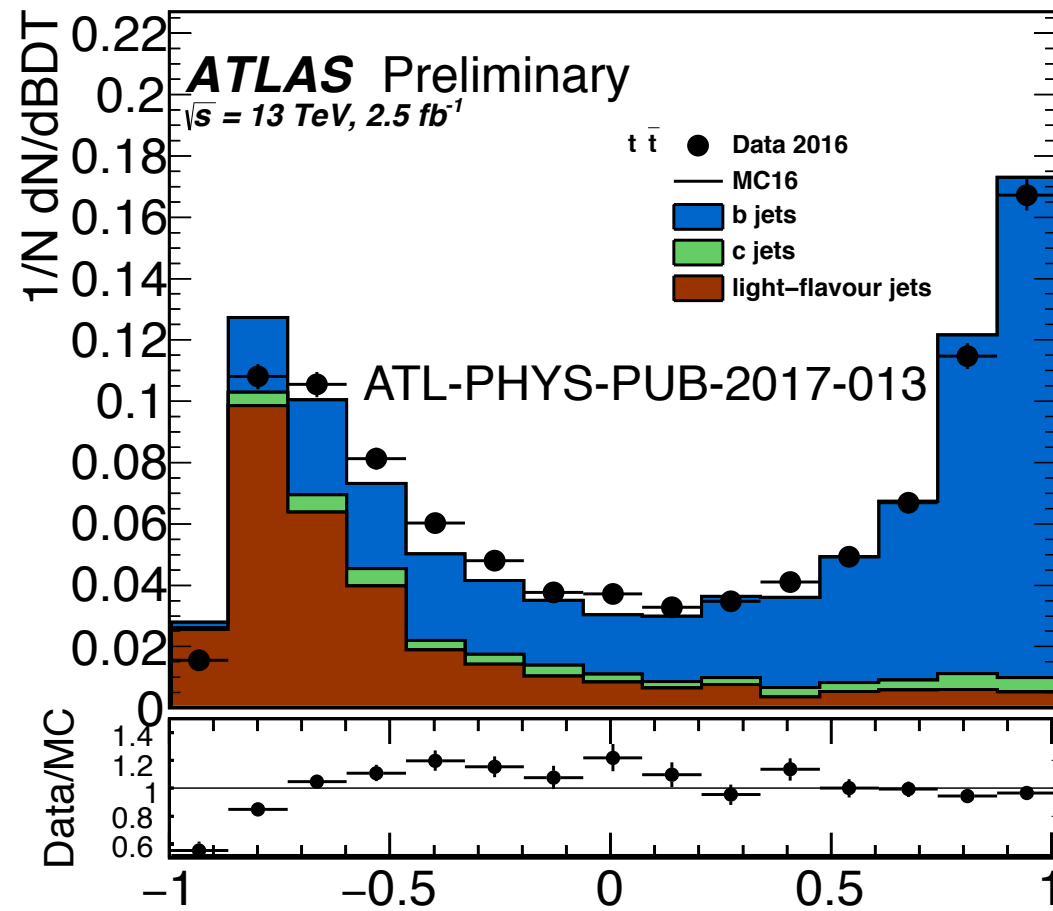
$$p(M, Q, B|V) = \sum_{\mathcal{F}} \Pr(\mathcal{F}|V) p(M|\mathcal{F}, V) p(Q|\mathcal{F}, V) \Pr(B|\mathcal{F}, V),$$

# Nearest Neighbors

In 2D, a nice extension of histogramming is to estimate the likelihood ratio based on the number of S and B points nearby.



# Boosted Decision Trees (BDTs)

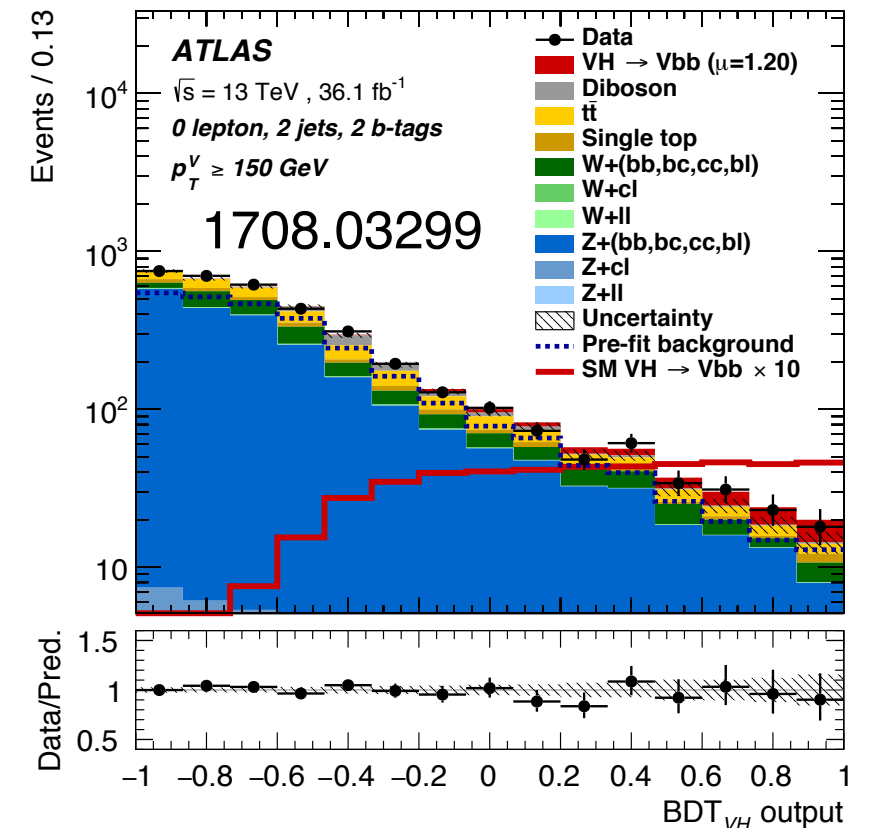
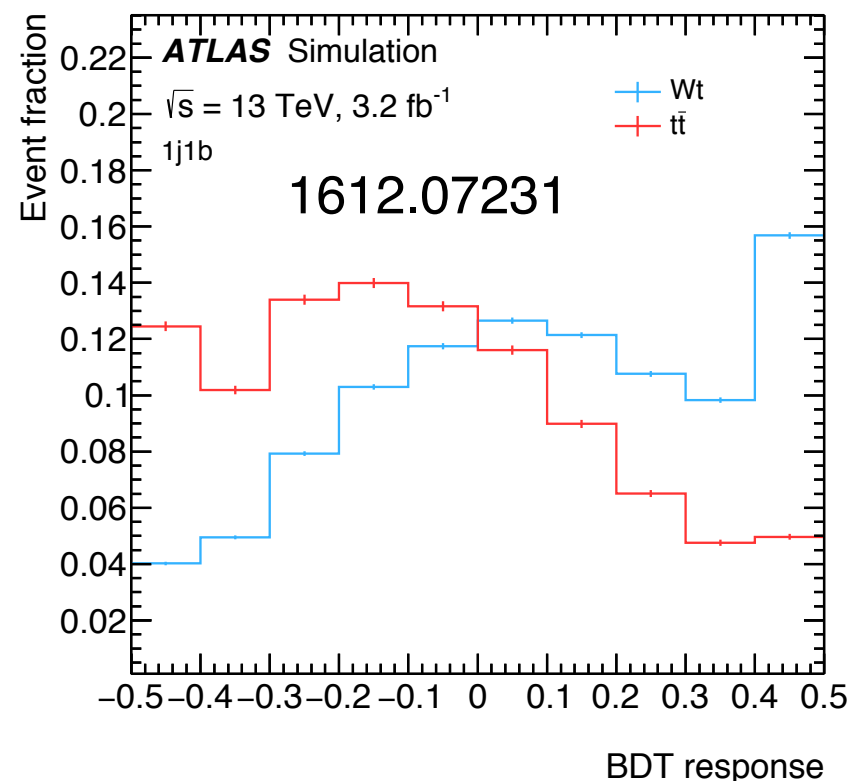
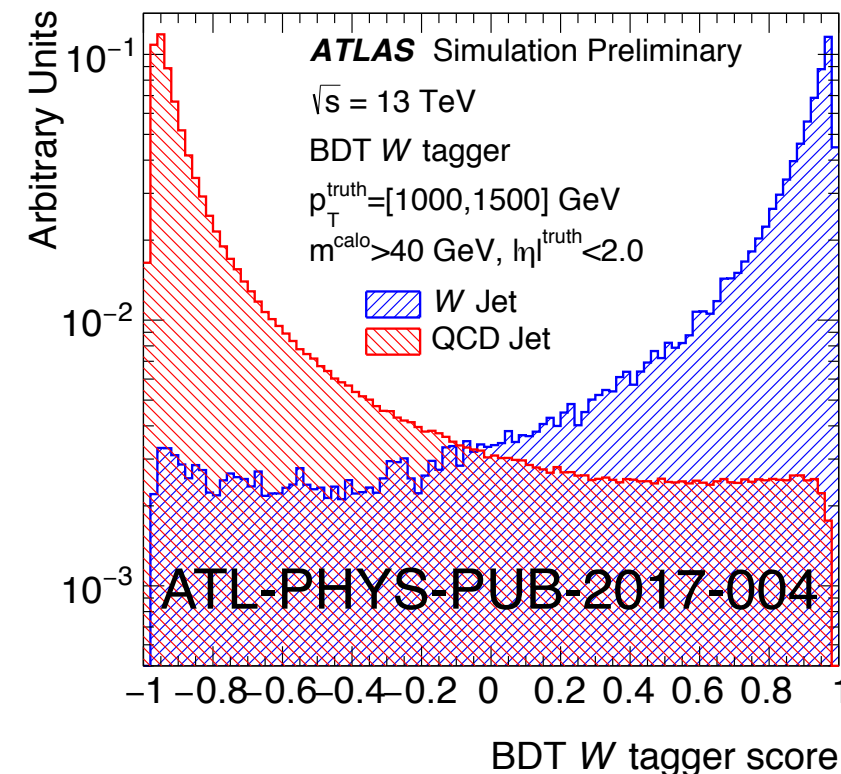
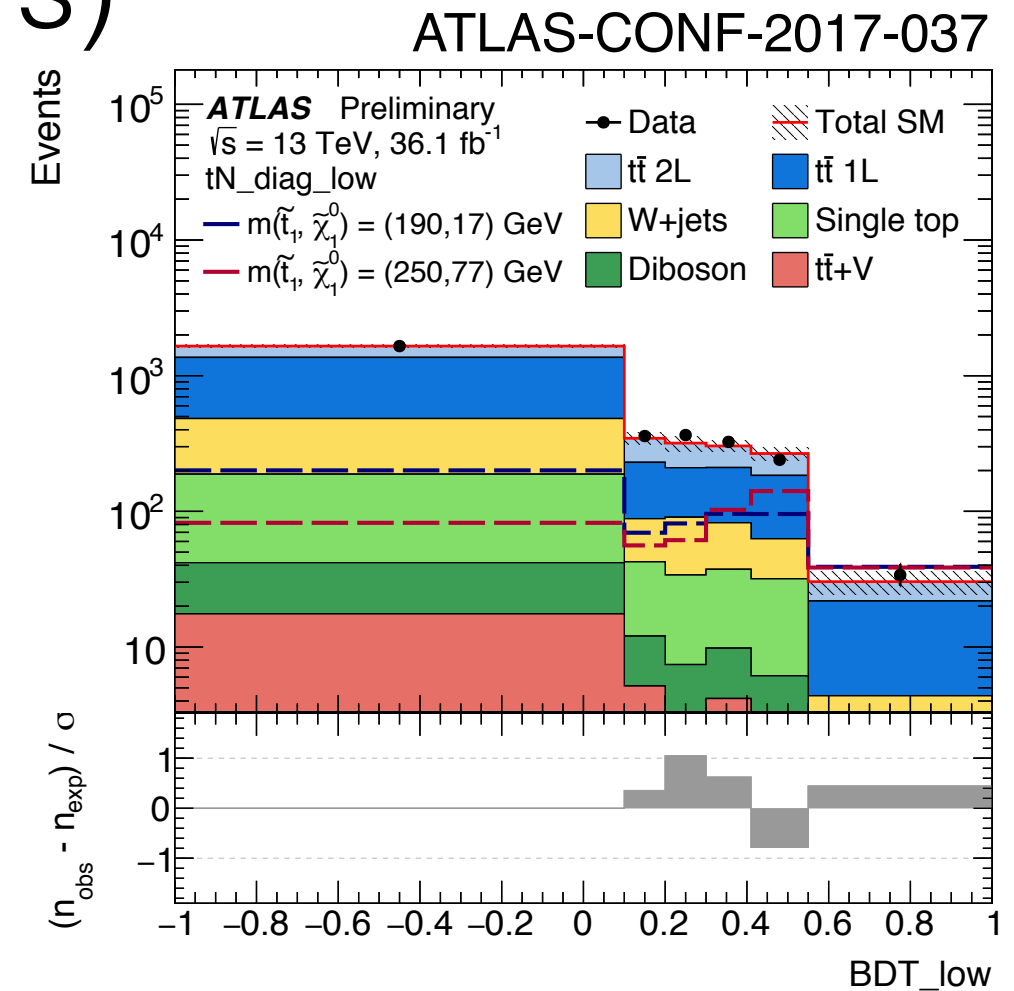


SMT BDT Discriminant

We love  
BDTs.

If  $3 < \dim(\text{feature vector}) < O(100)$

this is probably  
right for you!



# Boosted Decision Trees (BDTs)

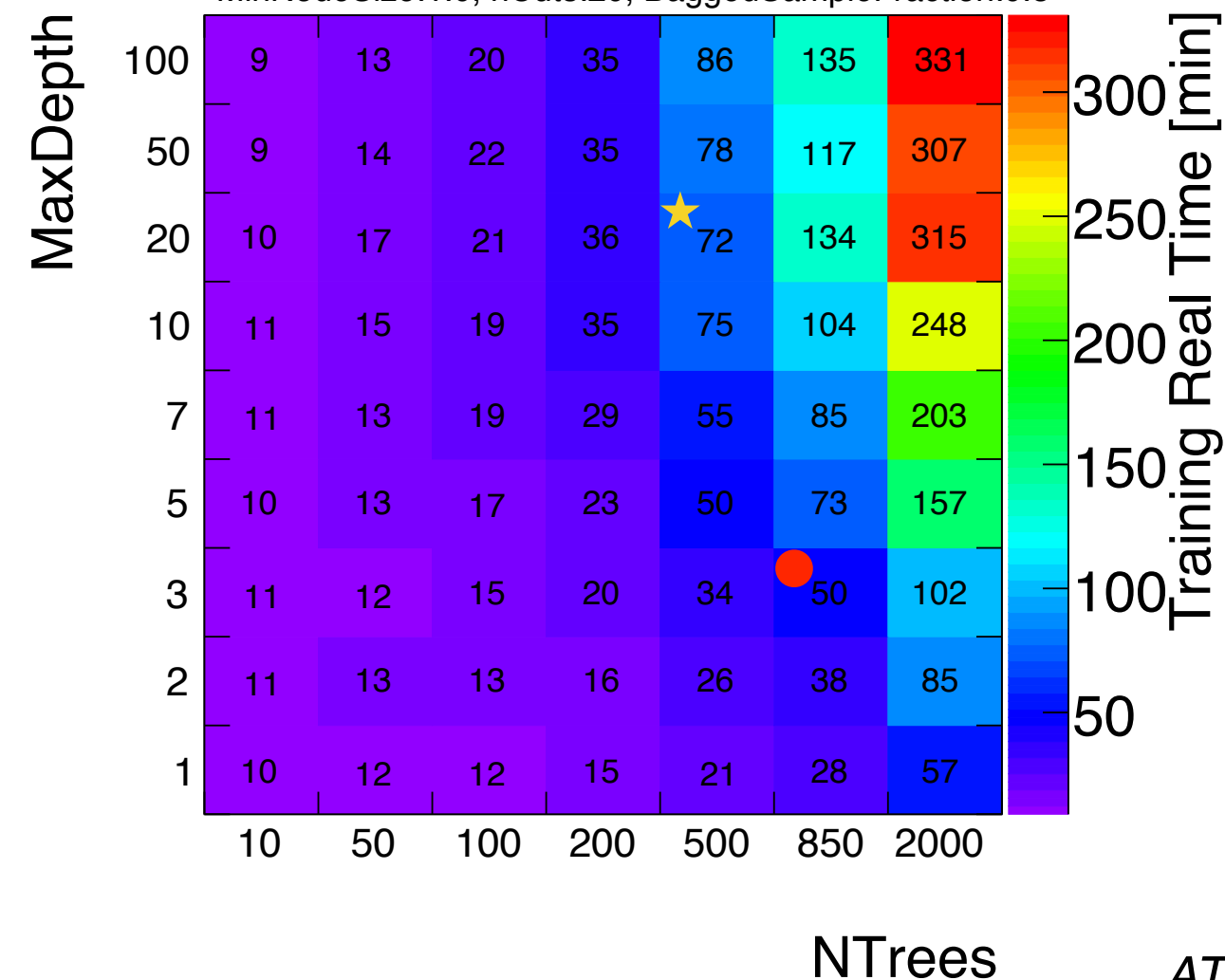
We love BDTs because they are fast to train  
and do not have very many parameters.  
They are also rather robust to *overtraining*.

## **ATLAS** Simulation Preliminary

$\sqrt{s}=13\text{TeV}$ , BDT W Tagging,  $\epsilon_{\text{sig}}^{\text{rel}}=50\%$

W Jet,  $p_{\text{T}}^{\text{truth}}=[200,2000]\text{ GeV}$ ,  $m^{\text{calo}}>40\text{ GeV}$ ,  $|\eta|^{\text{truth}}<2.0$

MinNodeSize:1.0, nCuts:20, BaggedSampleFraction:0.5

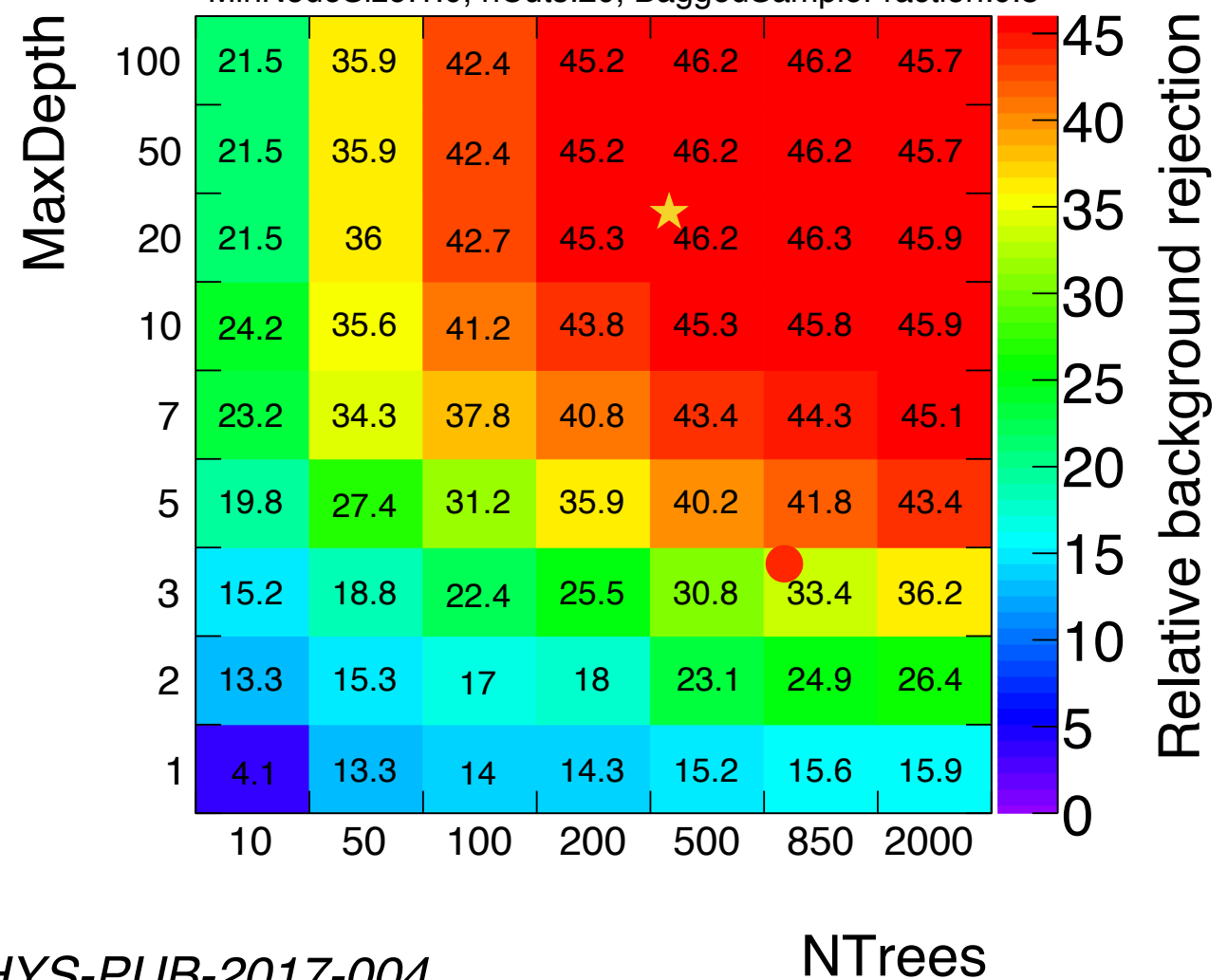


## **ATLAS** Simulation Preliminary

$\sqrt{s}=13\text{TeV}$ , BDT W Tagging,  $\epsilon_{\text{sig}}^{\text{rel}}=50\%$

W Jet,  $p_{\text{T}}^{\text{truth}}=[200,2000]\text{ GeV}$ ,  $m^{\text{calo}}>40\text{ GeV}$ ,  $|\eta|^{\text{truth}}<2.0$

MinNodeSize:1.0, nCuts:20, BaggedSampleFraction:0.5





# Boosted Decision Trees (BDTs)

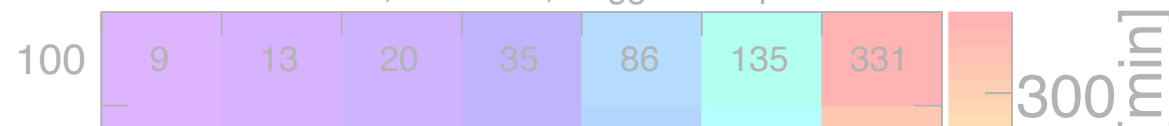
We love BDTs because they are fast to train and do not have very many parameters. They are also rather robust to *overtraining*.

**ATLAS** Simulation Preliminary

$\sqrt{s}=13\text{TeV}$ , BDT W Tagging,  $\epsilon_{\text{sig}}^{\text{rel}}=50\%$

W Jet,  $p_{\text{T}}^{\text{truth}}=[200,2000]\text{ GeV}$ ,  $m^{\text{calo}}>40\text{ GeV}$ ,  $|\eta|^{\text{truth}}<2.0$

MinNodeSize:1.0, nCuts:20, BaggedSampleFraction:0.5

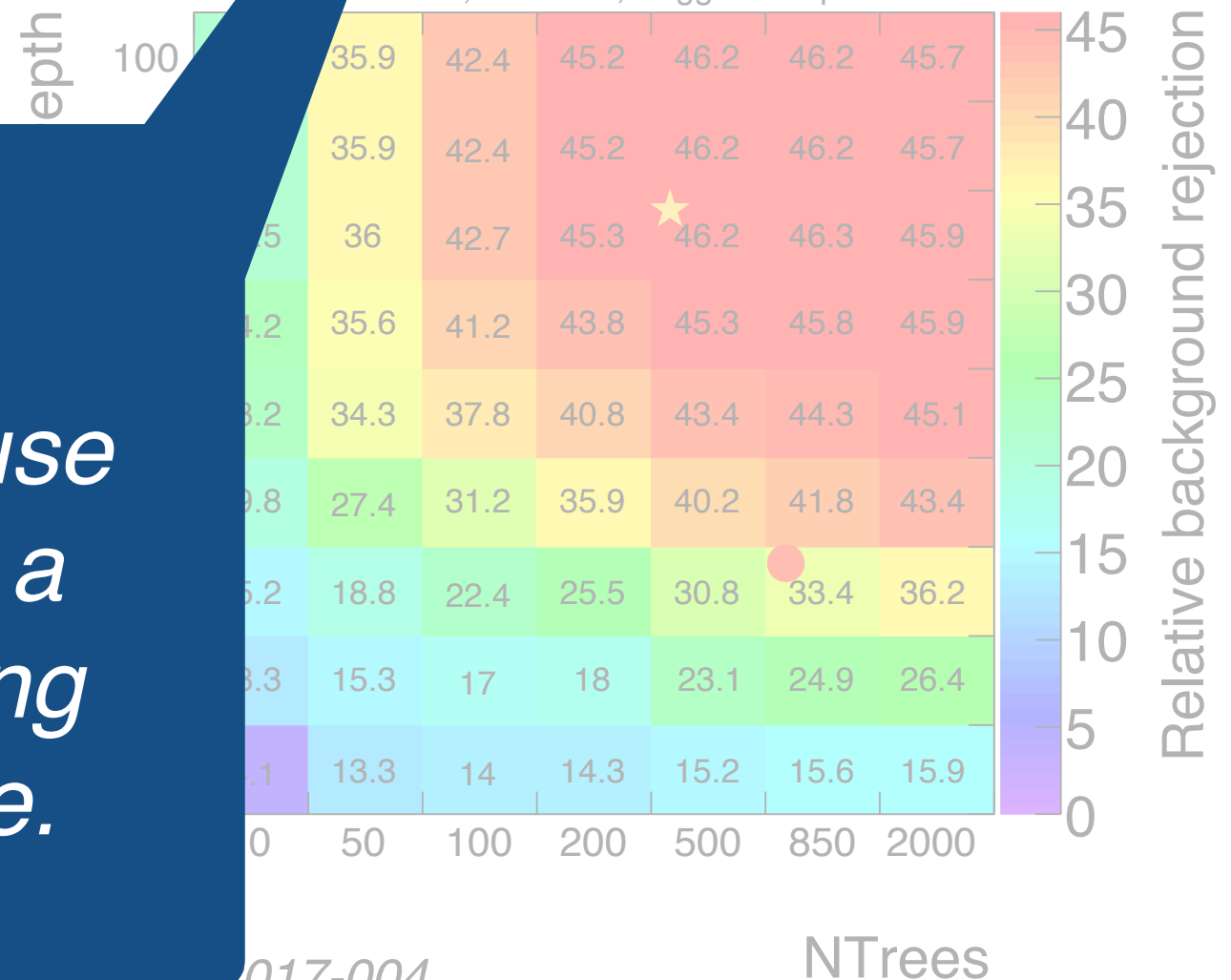


**ATLAS** Simulation Preliminary

$\sqrt{s}=13\text{TeV}$ , BDT W Tagging,  $\epsilon_{\text{sig}}^{\text{rel}}=50\%$

W Jet,  $p_{\text{T}}^{\text{truth}}=[200,2000]\text{ GeV}$ ,  $m^{\text{calo}}>40\text{ GeV}$ ,  $|\eta|^{\text{truth}}<2.0$

MinNodeSize:1.0, nCuts:20, BaggedSampleFraction:0.5

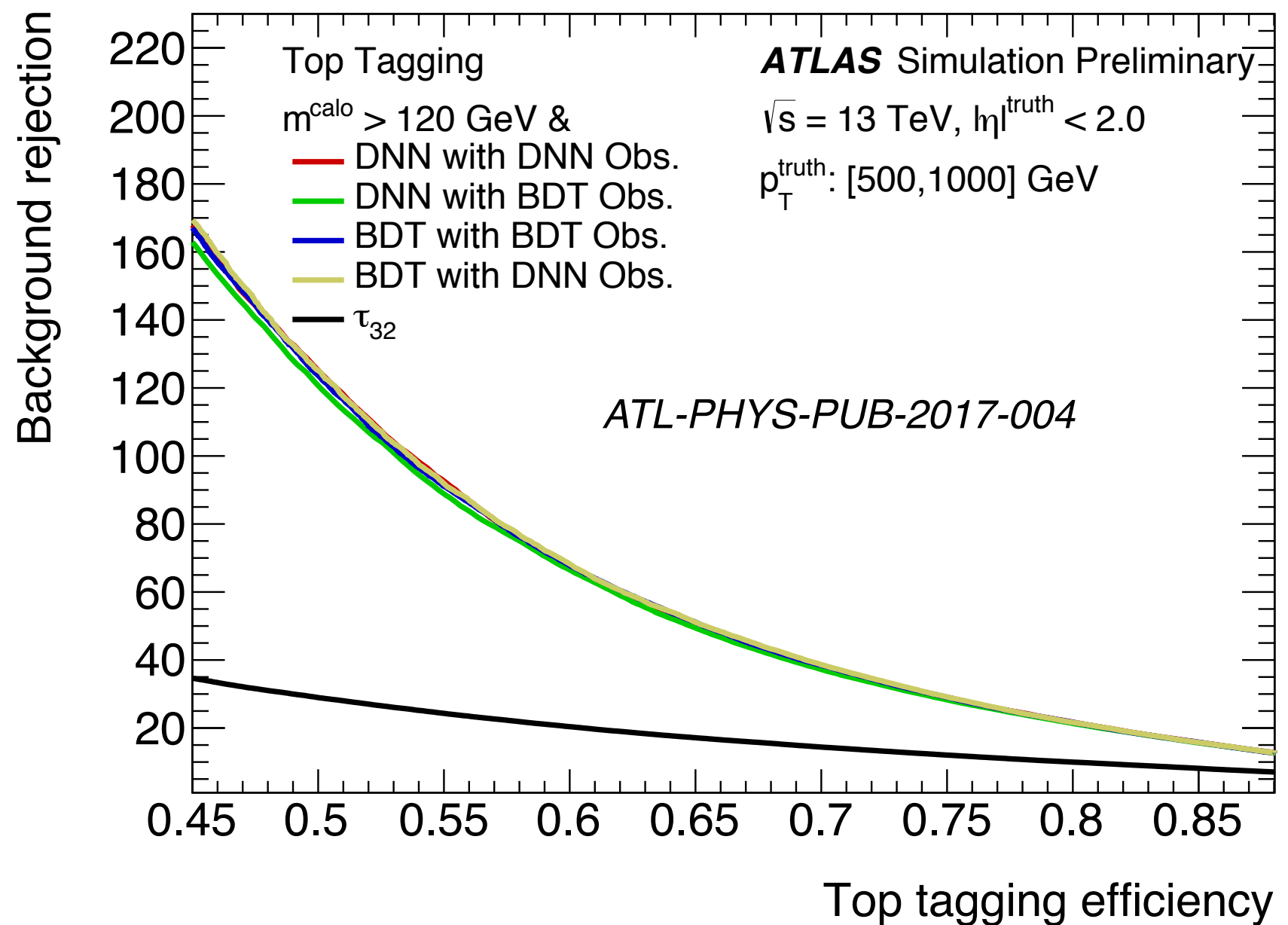


*Unless you have a lot of training data, it is better to use cross-validation instead of a single hold-out for evaluating out-of-sample performance.*

# Boosted Decision Trees (BDTs)

There is really not a good reason to use a DNN with  $\ll O(100)$  dimensions.

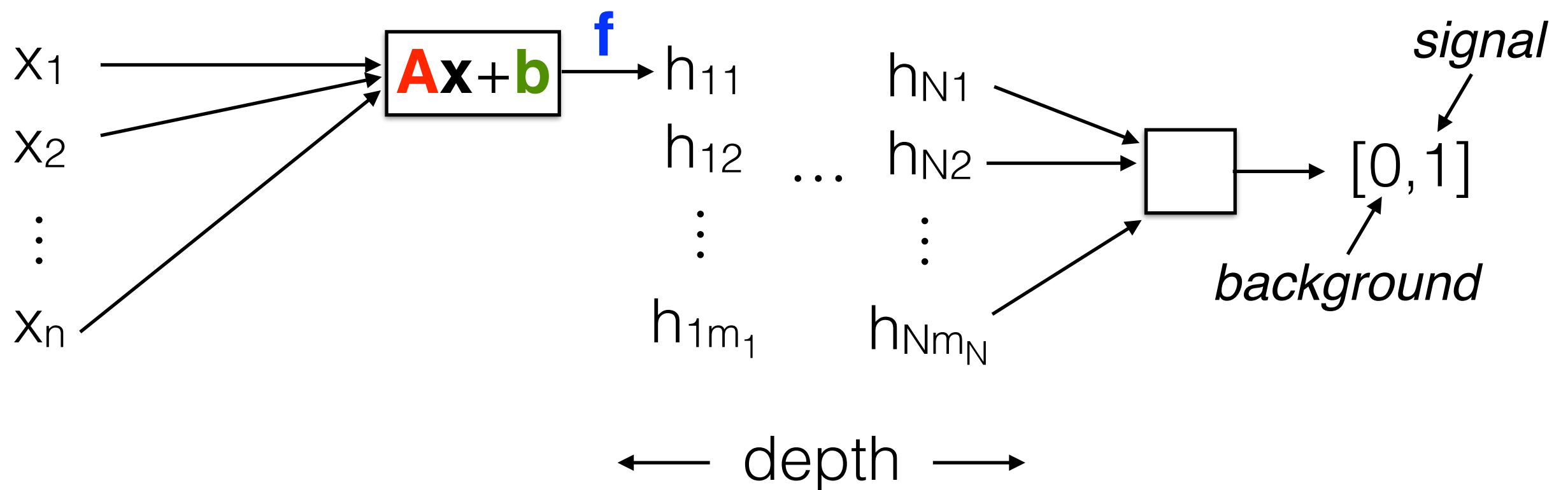
However, they are becoming increasingly easy to train ...



# Modern Deep NN's for Classification

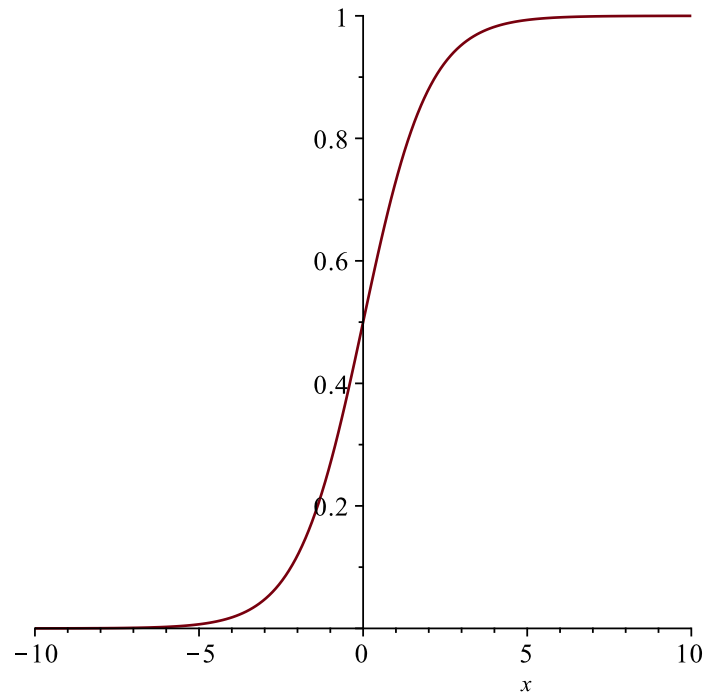
**Neural Network:** composition of functions  $f(\mathbf{Ax} + \mathbf{b})$  for inputs  $\mathbf{x}$  (features) matrix  $\mathbf{A}$  (weights), bias  $\mathbf{b}$ , non-linearity  $f$ .

N.B. I'm not mentioning biology - there may be a vague resemblance to parts of the brain, but that is not what modern NN's are about.



*Fact: NN's can approximate "any" function.*

# Choosing the non-linearity (activation function) **f**

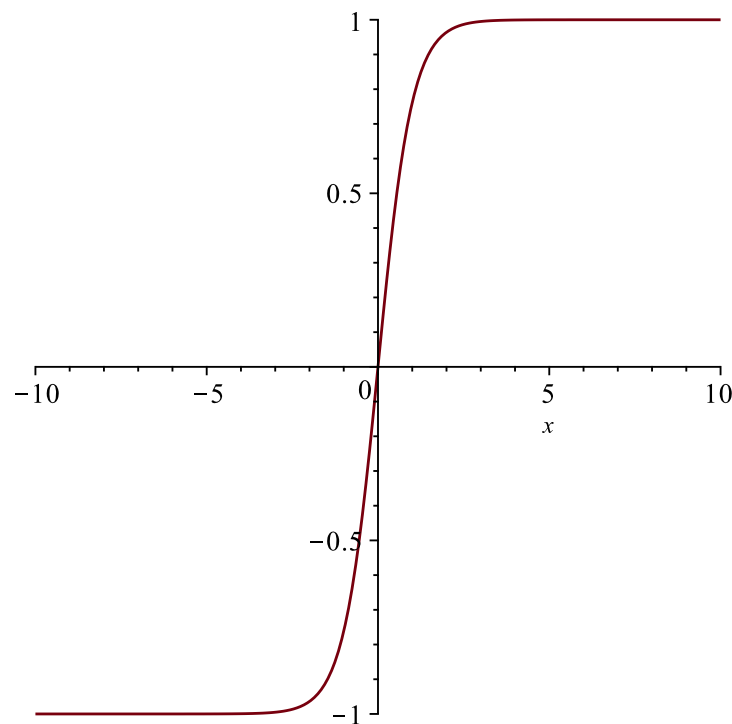


**Logistic (aka Sigmoid)**: one of the most widely-used functions in the past, now basically only used for the last layer.

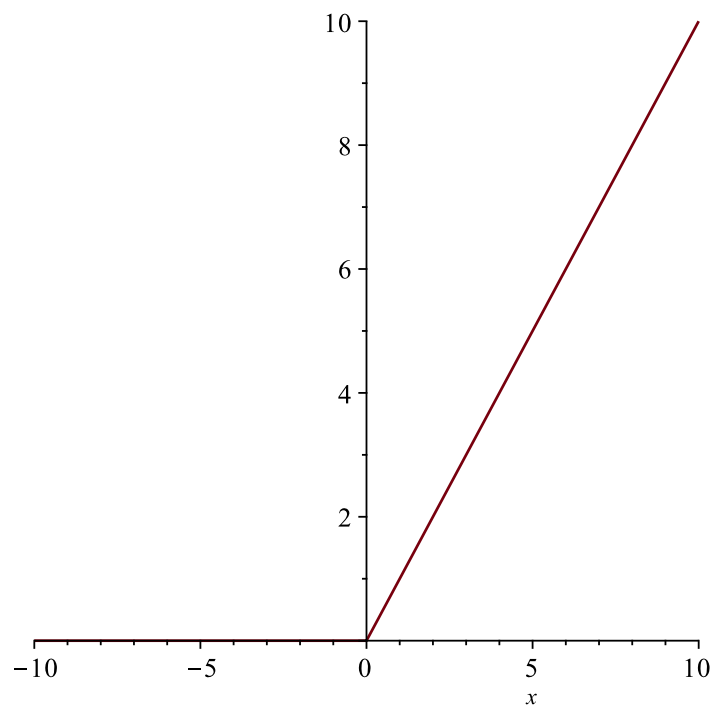
*generalization to multi-dimensional input: softmax*

$$\mathbf{f}(\vec{x}) = e^{x_i} / \sum_i e^{x_i}$$

**tanh**: similar story to sigmoid.

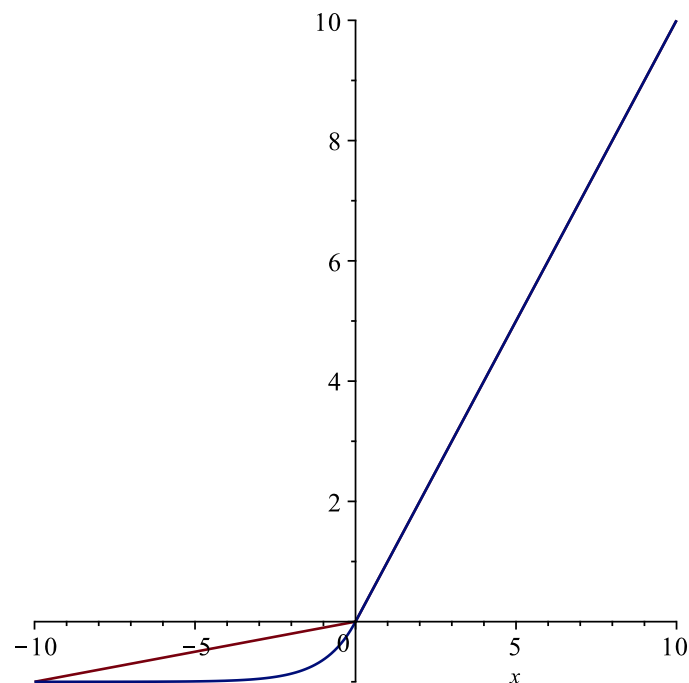


# Choosing the non-linearity (activation function) **f**



Rectified Linear Unit **ReLU**: one of the most widely-used functions now.

*do not suffer from the vanishing gradient problem*



**Leaky ReLU / Exponential LU (ELU):** variations on the ReLU that are popular.



Functions that act on multiple nodes in one layer

**MaxOut:** Take the maximum of multiple inputs

*reduces the dimensionality  
of a hidden layer*

**DropOut:** Randomly remove (for one forward/backward pass) nodes from a layer.

*helps with over-training*

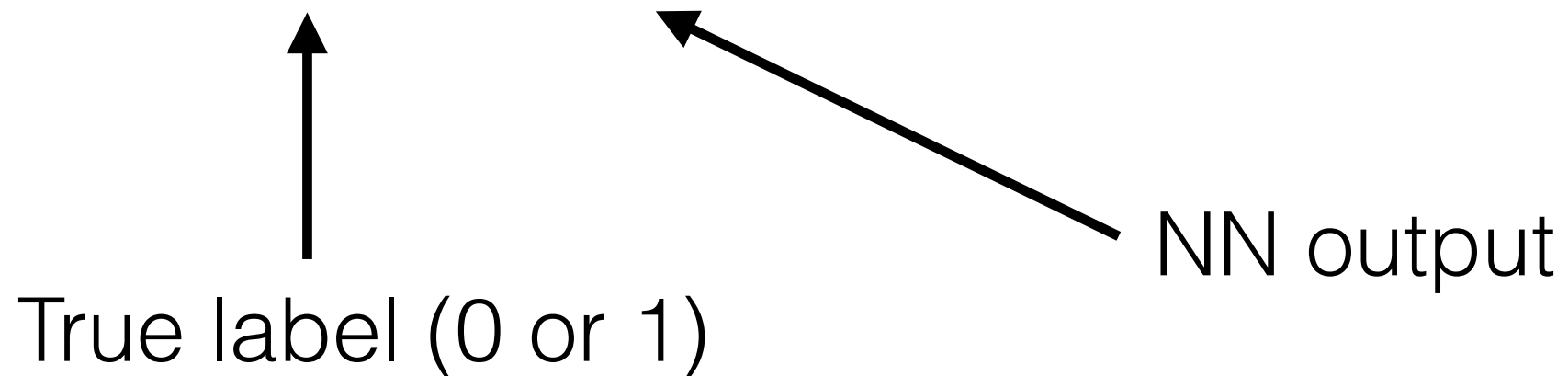
# (D)NN Training

Training proceeds by minimizing a loss function.

Typical loss functions

Squared error:  $(y_i - \hat{y}_i)^2$

Cross-entropy:  $-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$



True label (0 or 1)

NN output

## (D)NN Training

Objective function is minimized using stochastic gradient decent (almost exclusively with the Adam algorithm)

Stochastic gradient decent: Using single (or multiple “mini-batches”) examples, weights are updated:

$$A_{ij} \mapsto A_{ij} - \eta \nabla_{ij} \mathcal{L}$$

learning rate

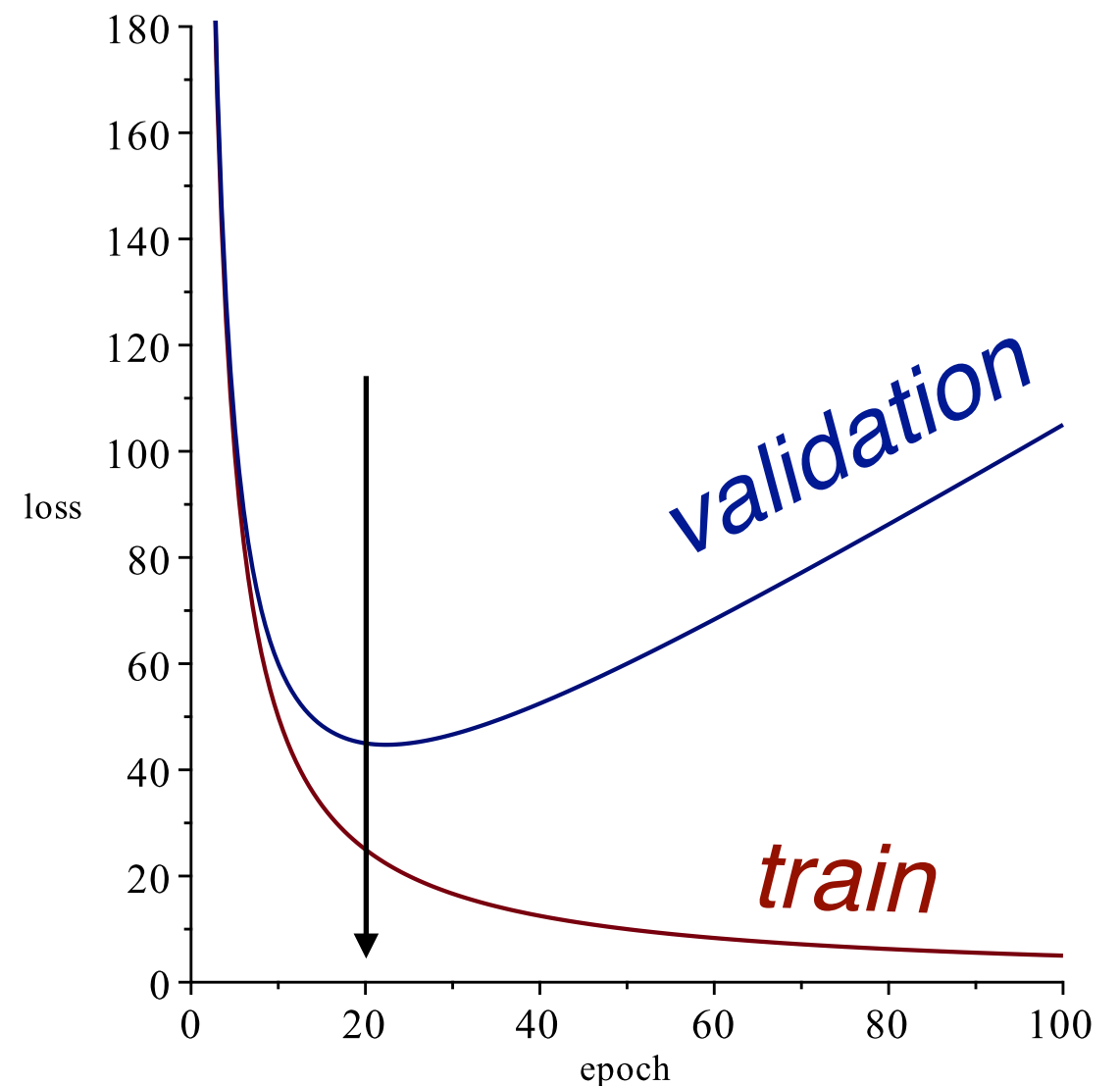
back-propagation: weights updated backwards and gradients are recycled.

N.B. a NN can do better than random **before any training!**  
For instance, if you initialize all the weights to 1 and the signal has generally higher values then the NN will beat random.

# (D)NN Training

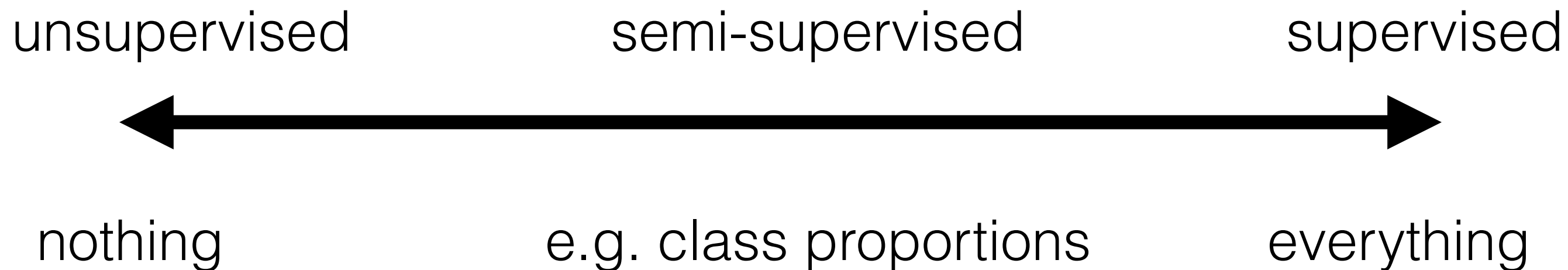
Training proceeds multiple times (epochs), reshuffling the data.

Early stopping: stop at the epoch where the validation error starts to increase



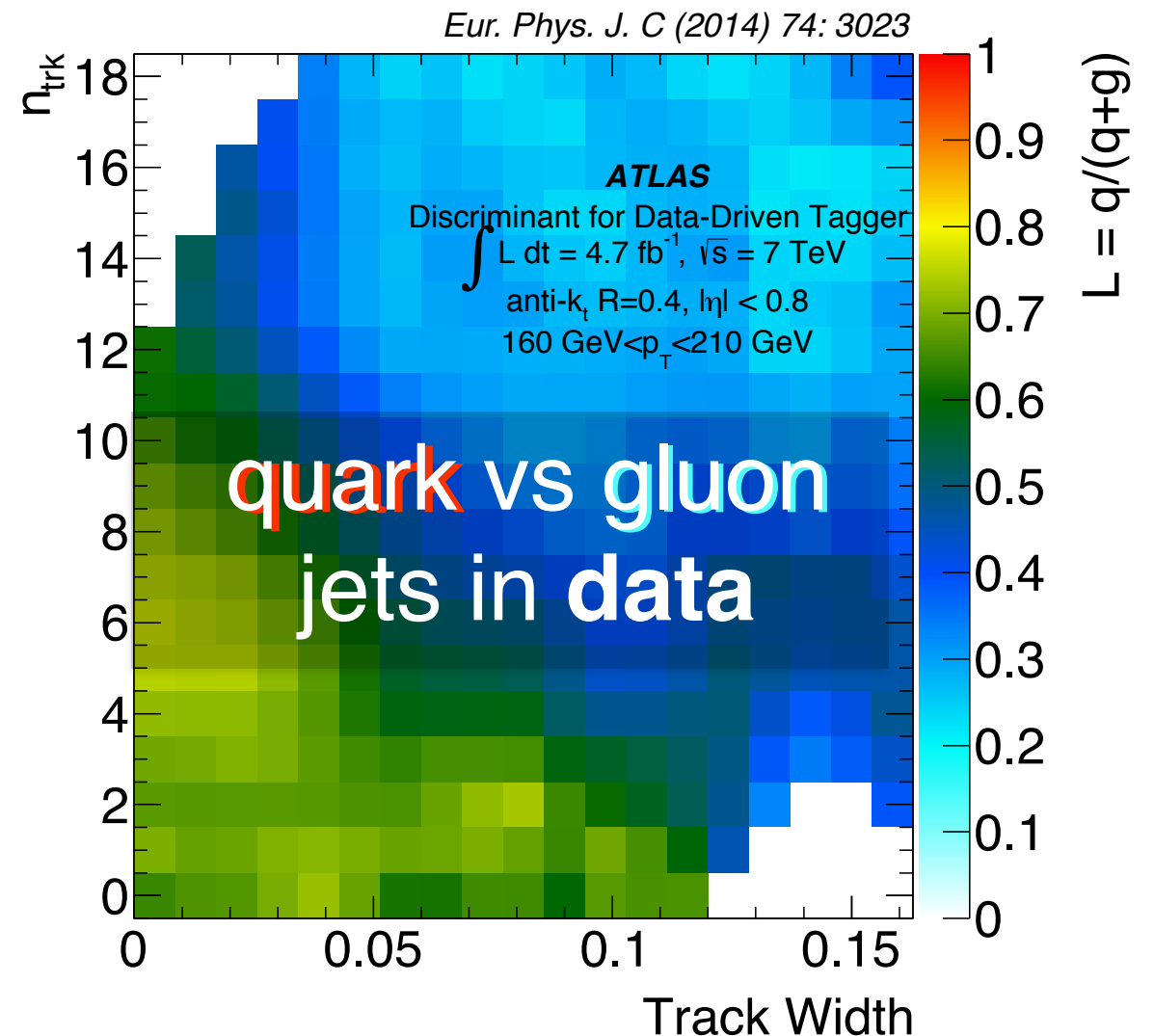
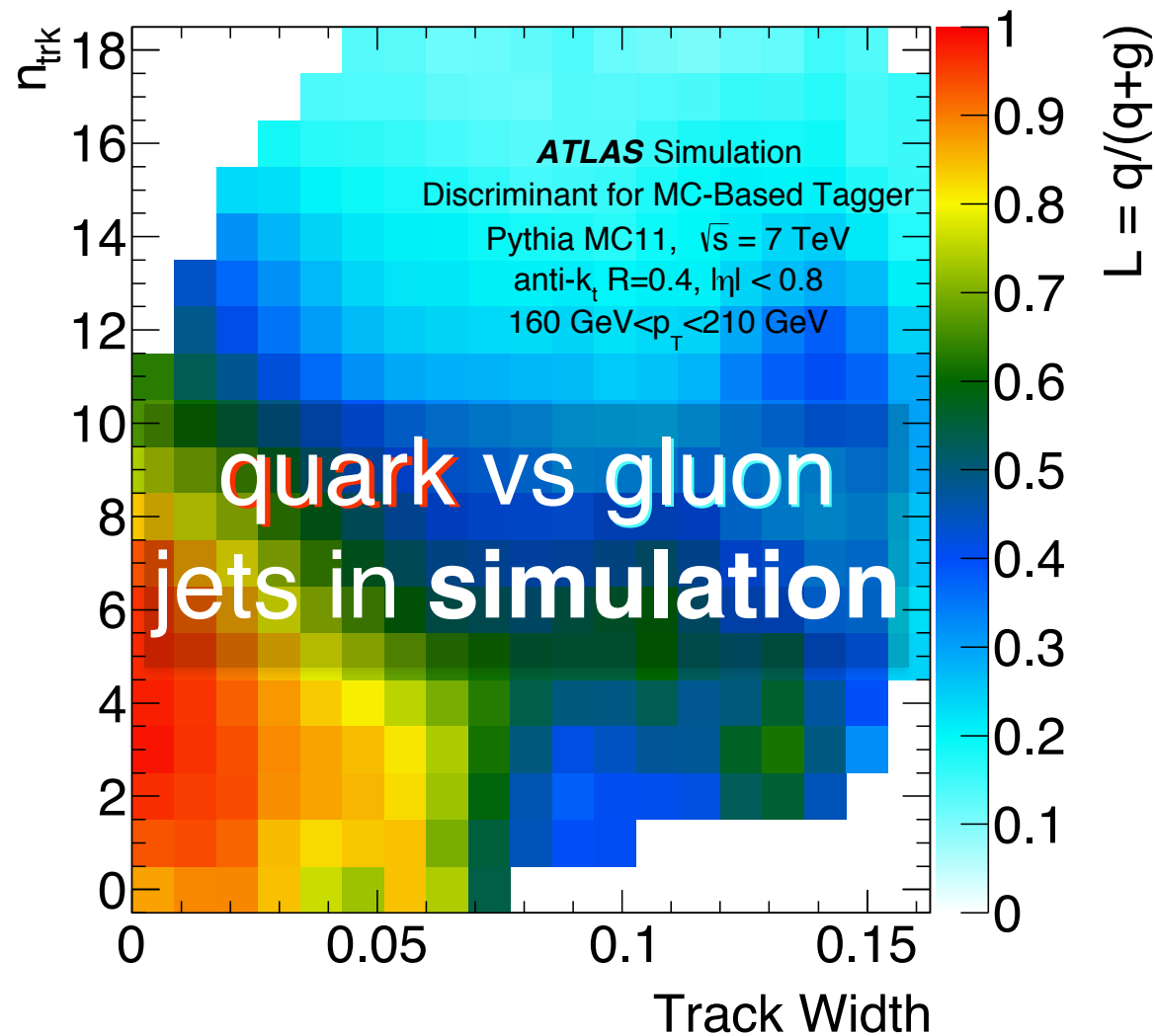
If you are new to these concepts, I would encourage you to take a look at [this notebook](#) which implements some of the ideas in practice.

Before closing, I'll leave you with one last concept: semi-supervised learning.



how much you know about per-example labels

For supervised learning, we depend on labels  
labels usually come from simulation



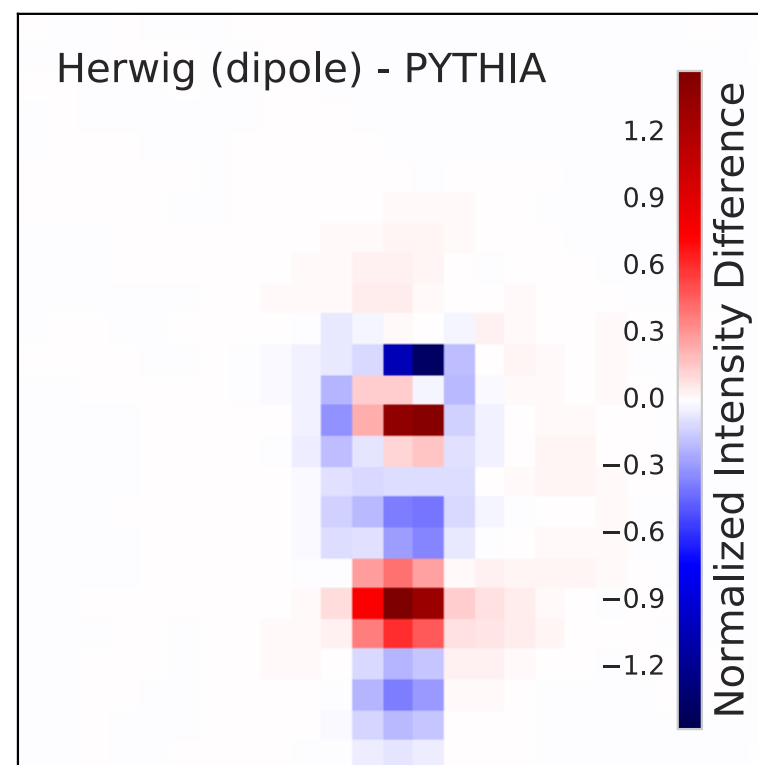
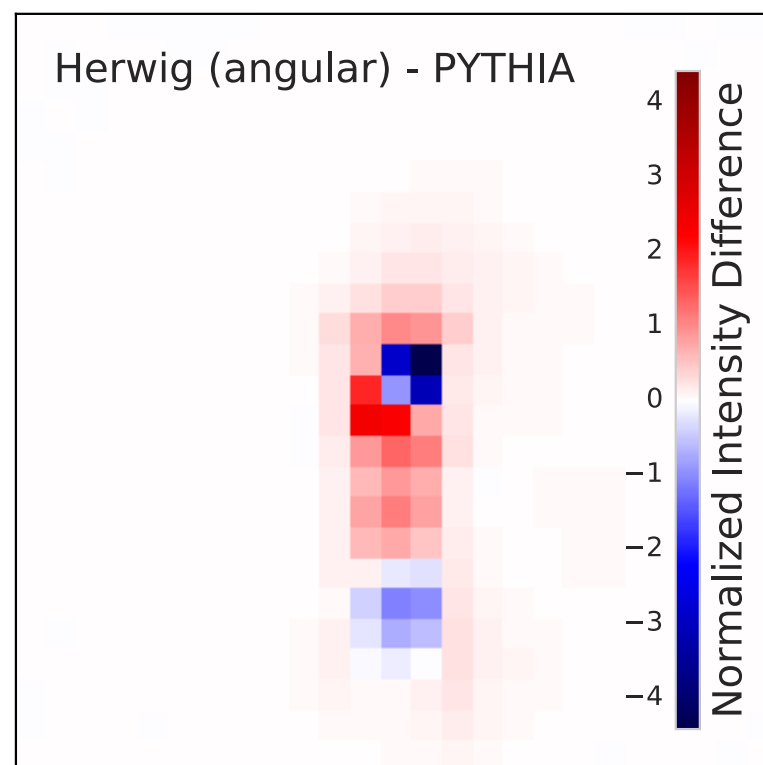
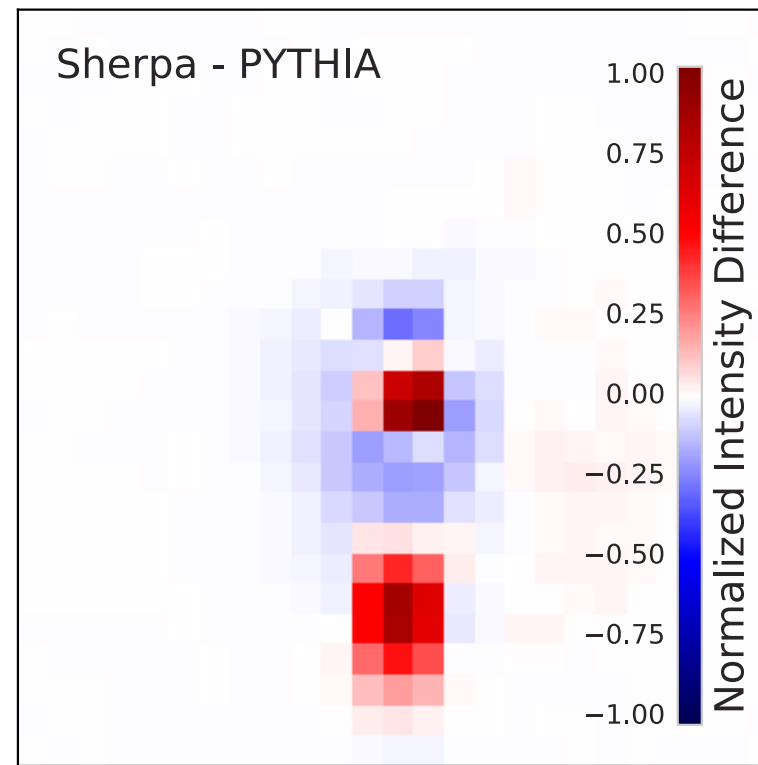
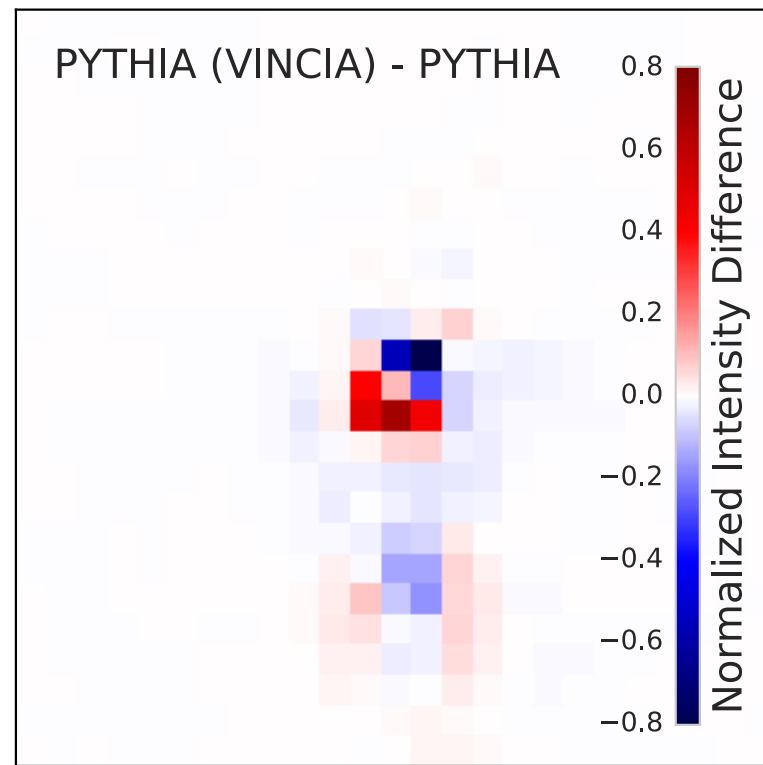
What if data and simulation are very different?  
...your classifier will be sub-optimal



# Boosted $W$ boson jets

*J. Barnard et al.*

Phys. Rev. D 95, 014018 (2017)



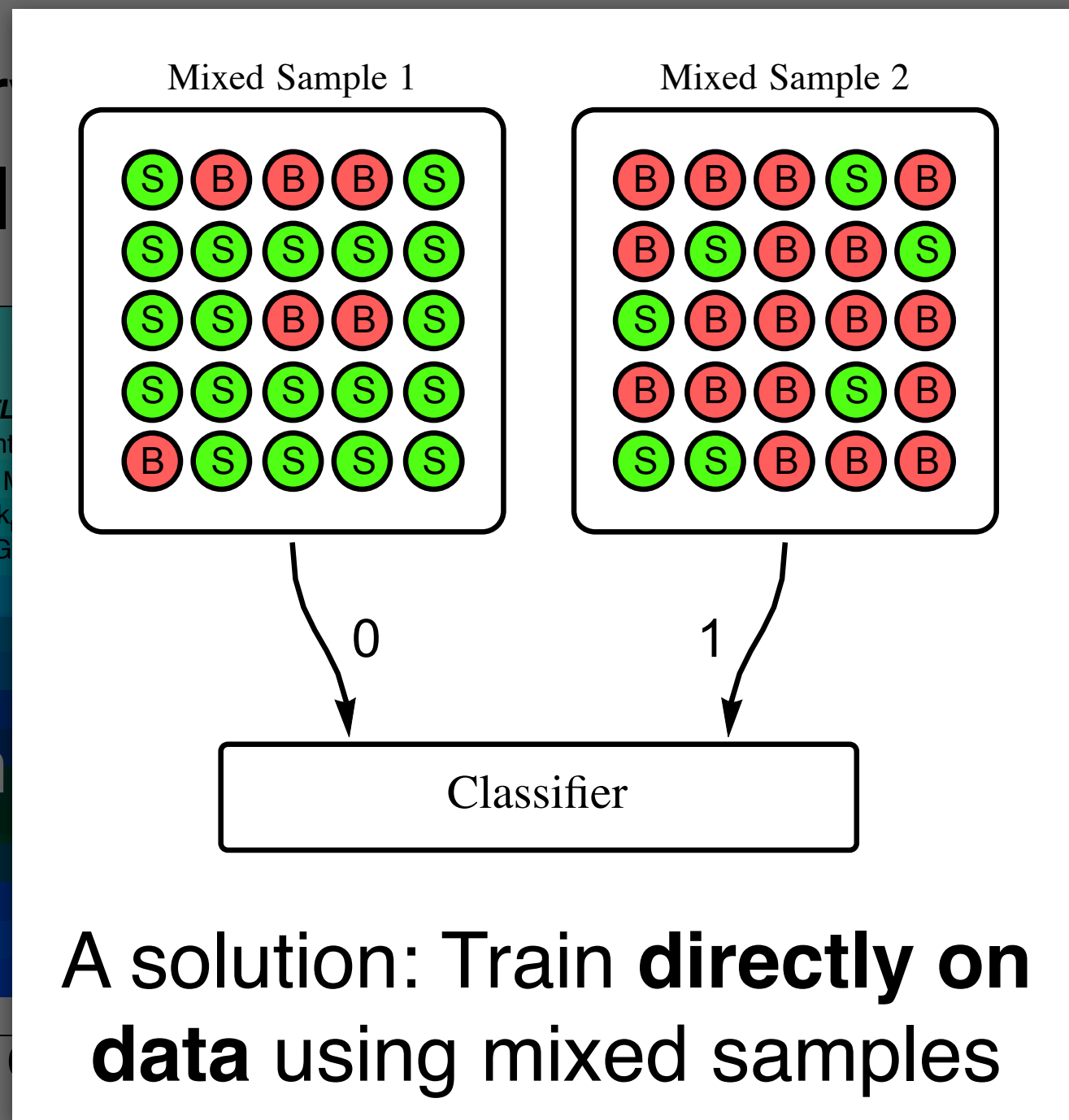
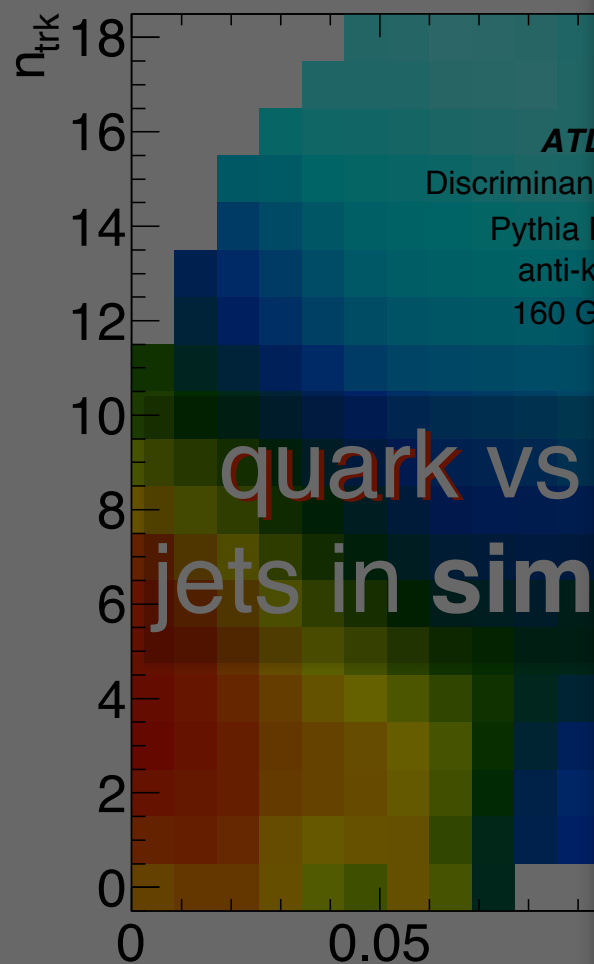
DNN classifiers  
can **exploit**  
subtle features

subtle features are  
**hard to model !**

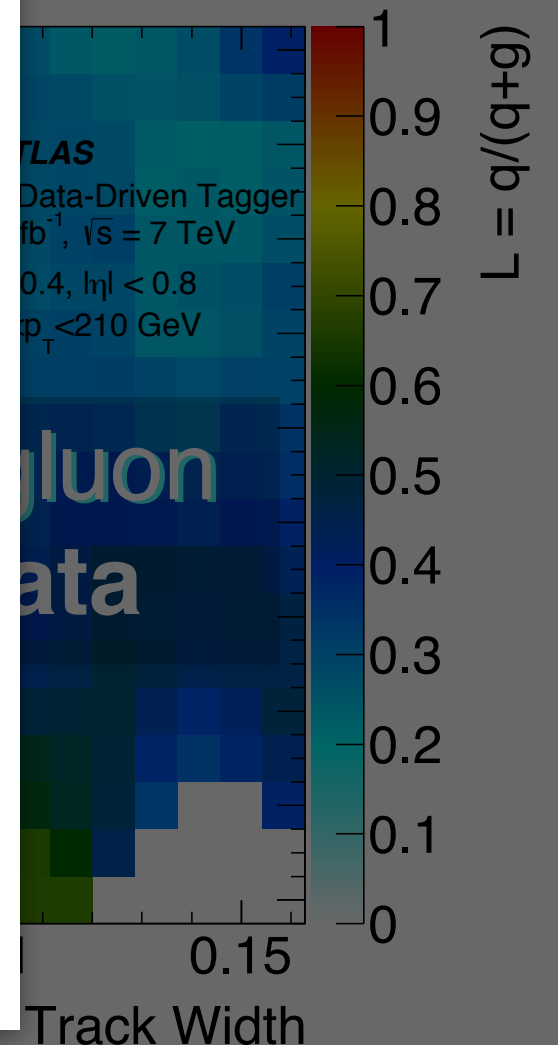
we need to be  
careful about which  
models we use -  
**only data is correct**

N.B. not all of these have been tuned to the same data

For super  
label



n labels  
ion



What if data and simulation are very different?

...your classifier will be sub-optimal

E. Metodiev, **BPN**, J. Thaler, JHEP 10 (2017) 174

related ideas: L. Dery, **BPN**, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145

and T. Cohen, M. Freytsis, B. Ostdiek, JHEP 02 (2018) 034

# Training on data:

learning when you know (basically) nothing

For supervised learning, we depend on labels

labels usually come from simulation

$$L_{M_1/M_2} = \frac{p_{M_1}}{p_{M_2}} = \frac{f_1 p_S + (1 - f_1) p_B}{f_2 p_S + (1 - f_2) p_B} = \frac{f_1 L_{S/B} + (1 - f_1)}{f_2 L_{S/B} + (1 - f_2)},$$

optimal classifier  
for  $M_1$  versus  $M_2$

optimal classifier  
for S versus B

What if data and simulation are very different?

...your classifier will be sub-optimal

E. Metodiev, **BPN**, J. Thaler, JHEP 10 (2017) 174

related ideas: L. Dery, **BPN**, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145

and T. Cohen, M. Freytsis, B. Ostdiek, JHEP 02 (2018) 034

# Training on data:

learning when you know (basically) nothing

For supervised learning, we depend on labels

labels usually come from simulation

$$L_{M_1/M_2} = \frac{p_{M_1}}{p_{M_2}} = \frac{f_1 p_S + (1 - f_1) p_B}{f_2 p_S + (1 - f_2) p_B} = \frac{f_1 L_{S/B} + (1 - f_1)}{f_2 L_{S/B} + (1 - f_2)},$$

optimal classifier  
for  $M_1$  versus  $M_2$

optimal classifier  
for S versus B

$$\partial_{L_{S/B}} L_{M_1/M_2} = (f_1 - f_2) / (f_2 L_{S/B} - f_2 + 1)^2 > 0$$

What if data and simulation are very different?

...your classifier will be sub-optimal

E. Metodiev, **BPN**, J. Thaler, JHEP 10 (2017) 174

related ideas: L. Dery, **BPN**, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145

and T. Cohen, M. Freytsis, B. Ostdiek, JHEP 02 (2018) 034

# Training on data:

learning when you know (basically) nothing

For supervised learning, we depend on labels

labels usually come from simulation

$$L_{M_1/M_2} = \frac{p_{M_1}}{p_{M_2}} = \frac{f_1 p_S + (1 - f_1) p_B}{f_2 p_S + (1 - f_2) p_B} = \frac{f_1 L_{S/B} + (1 - f_1)}{f_2 L_{S/B} + (1 - f_2)},$$

*aka optimal classifiers are the same !  
(one of the few analytic results we have...)*

$$\partial_{L_{S/B}} L_{M_1/M_2} = (f_1 - f_2) / (f_2 L_{S/B} - f_2 + 1)^2 > 0$$

What if data and simulation are very different?

...your classifier will be sub-optimal

E. Metodiev, **BPN**, J. Thaler, JHEP 10 (2017) 174

related ideas: L. Dery, **BPN**, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145

and T. Cohen, M. Freytsis, B. Ostdiek, JHEP 02 (2018) 034

Training on data:

learning when you know (basically) nothing

For supervised learning, we depend on labels

labels usually come from simulation

$$L_{M_1/M_2} = \frac{p_{M_1}}{p_{M_2}} = \frac{f_1 p_S + (1 - f_1) p_B}{f_2 p_S + (1 - f_2) p_B} = \frac{f_1 L_{S/B} + (1 - f_1)}{f_2 L_{S/B} + (1 - f_2)}$$

**Does it work? See Bryan's talk on Friday!**

classifier  
for  $M_1$  versus  $M_2$

optimal classifier  
for S versus B

$$\partial_{L_{S/B}} L_{M_1/M_2} = (f_1 - f_2) / (f_2 L_{S/B} - f_2 + 1)^2 > 0$$

What if data and simulation are very different?

...your classifier will be sub-optimal

E. Metodiev, **BPN**, J. Thaler, JHEP 10 (2017) 174

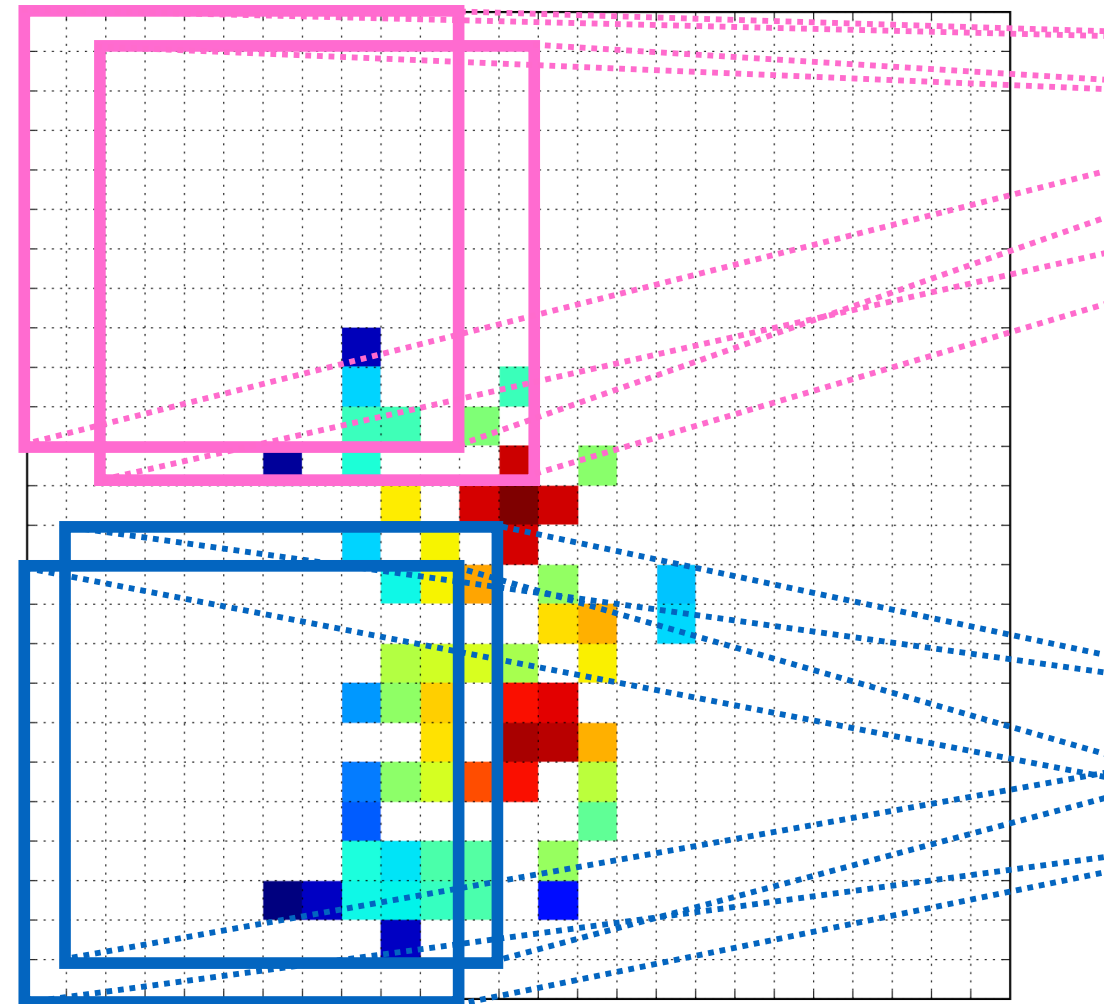
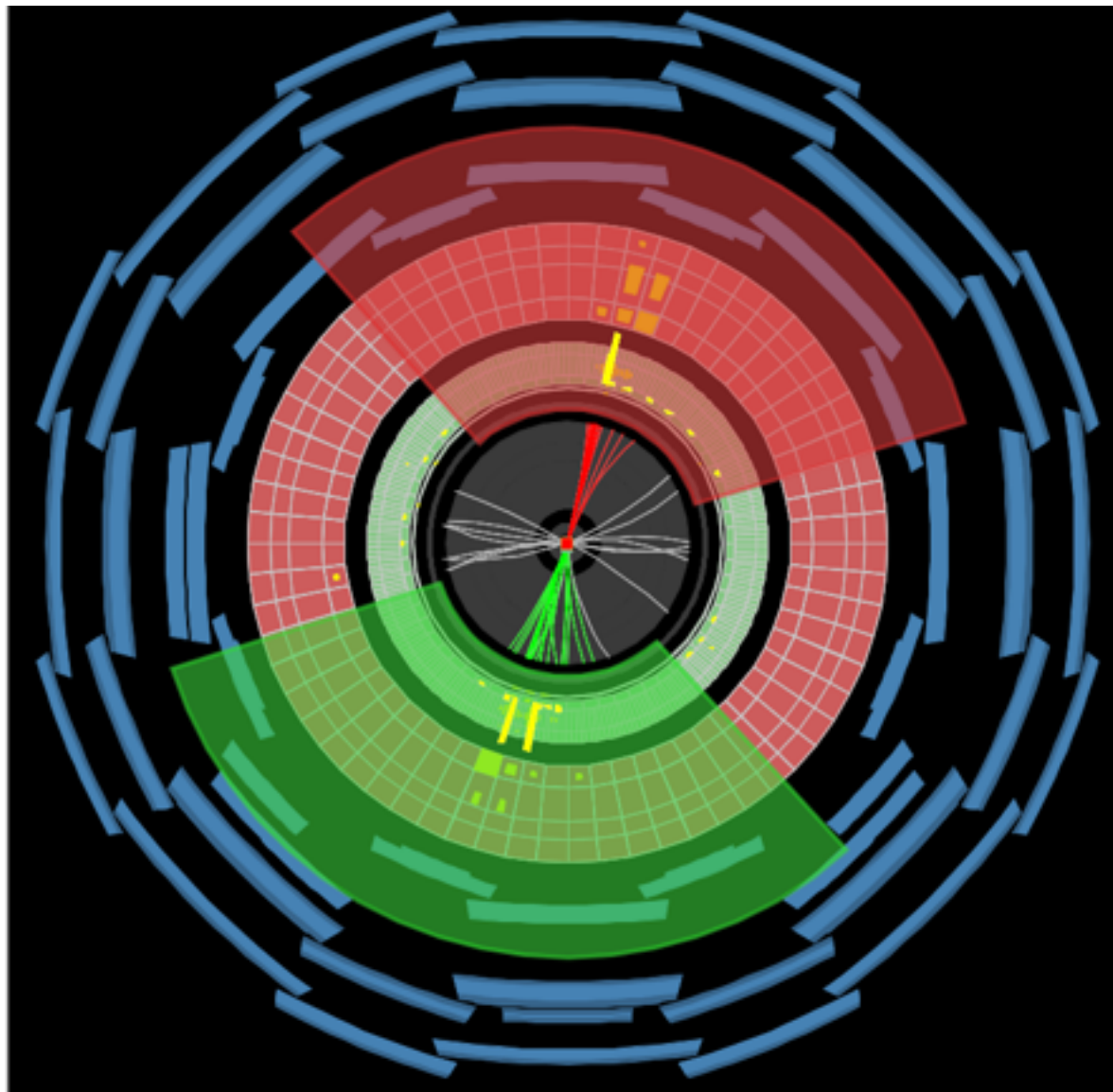
related ideas: L. Dery, **BPN**, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145

and T. Cohen, M. Freytsis, B. Ostdiek, JHEP 02 (2018) 034



# The future

(D)NN's are powerful tools that will help us fully exploit the physics potential of our experiments.



**We must be cautious to apply the right tool for the right job.**  
The more you know, the less black the boxes will be...