Machine Learning for High Energy Physics

Daniel Whiteson, UC Irvine

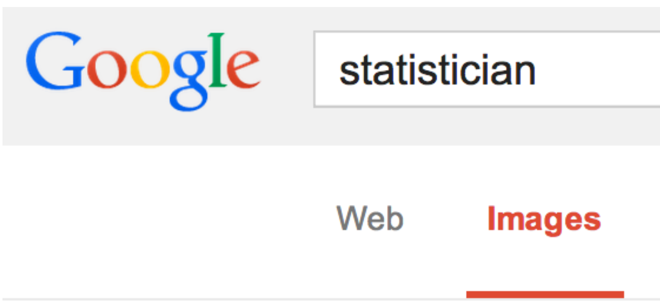
iSTEP 2016, Beijing

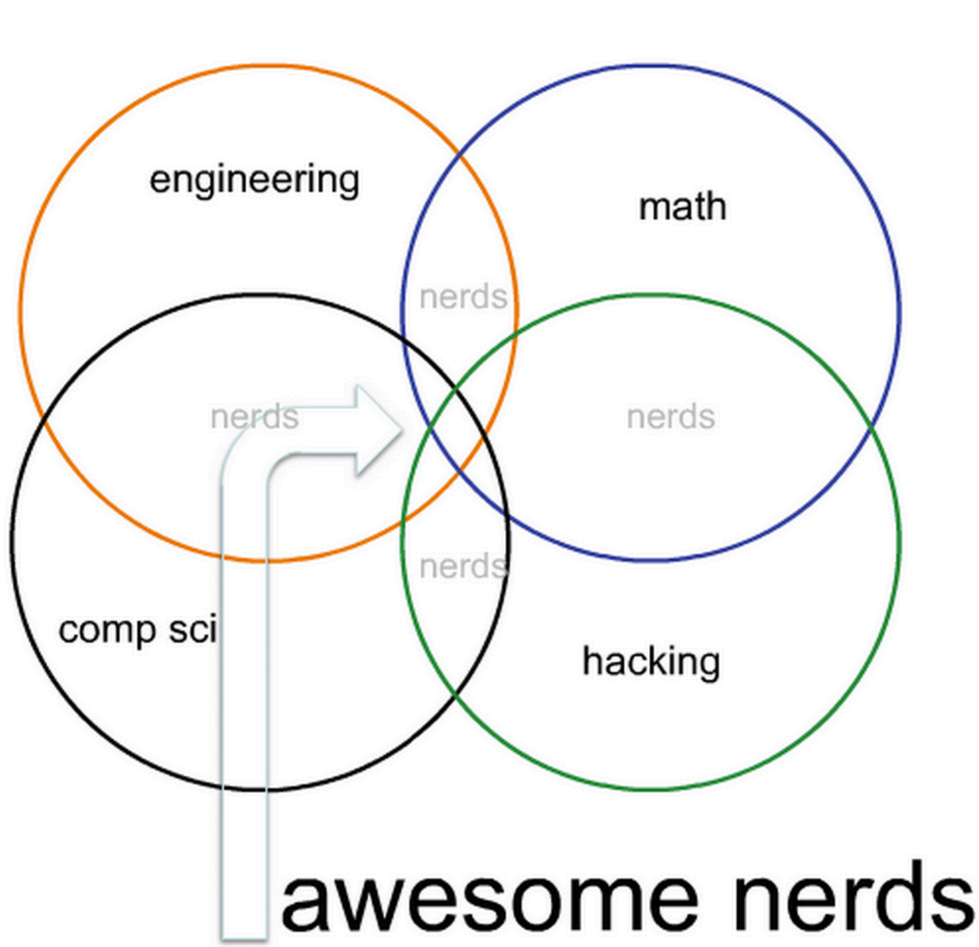
Caveat

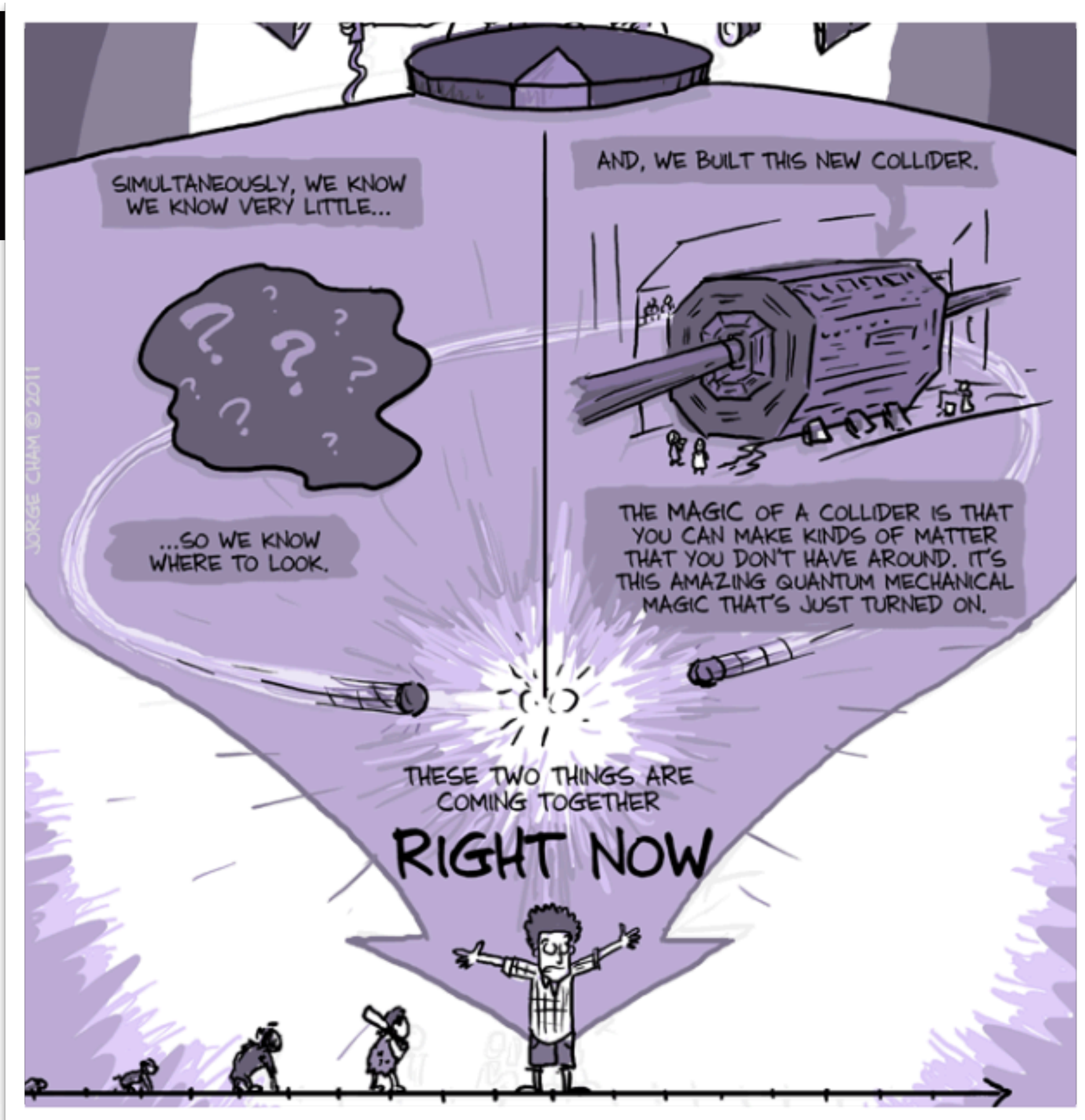
I am not a professional statistician!

Caveat

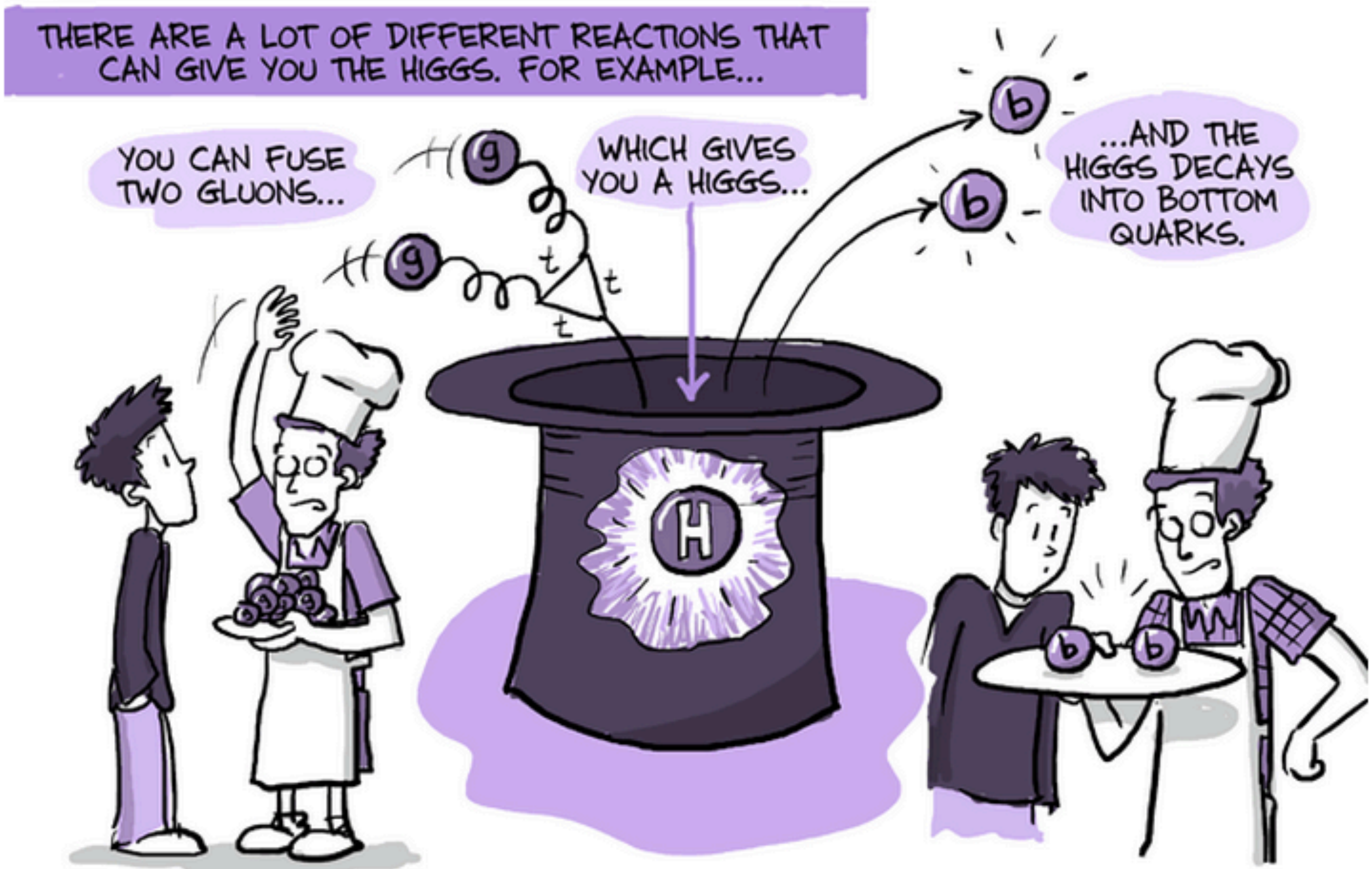
I am not a professional statistician!






SIMULTANEOUSLY, WE KNOW WE KNOW VERY LITTLE...
...SO WE KNOW WHERE TO LOOK.
AND, WE BUILT THIS NEW COLLIDER.
THE MAGIC OF A COLLIDER IS THAT YOU CAN MAKE KINDS OF MATTER THAT YOU DON'T HAVE AROUND. IT'S THIS AMAZING QUANTUM MECHANICAL MAGIC THAT'S JUST TURNED ON.
THESE TWO THINGS ARE COMING TOGETHER
RIGHT NOW
JORGE CHAM ©2011

Making a new particle

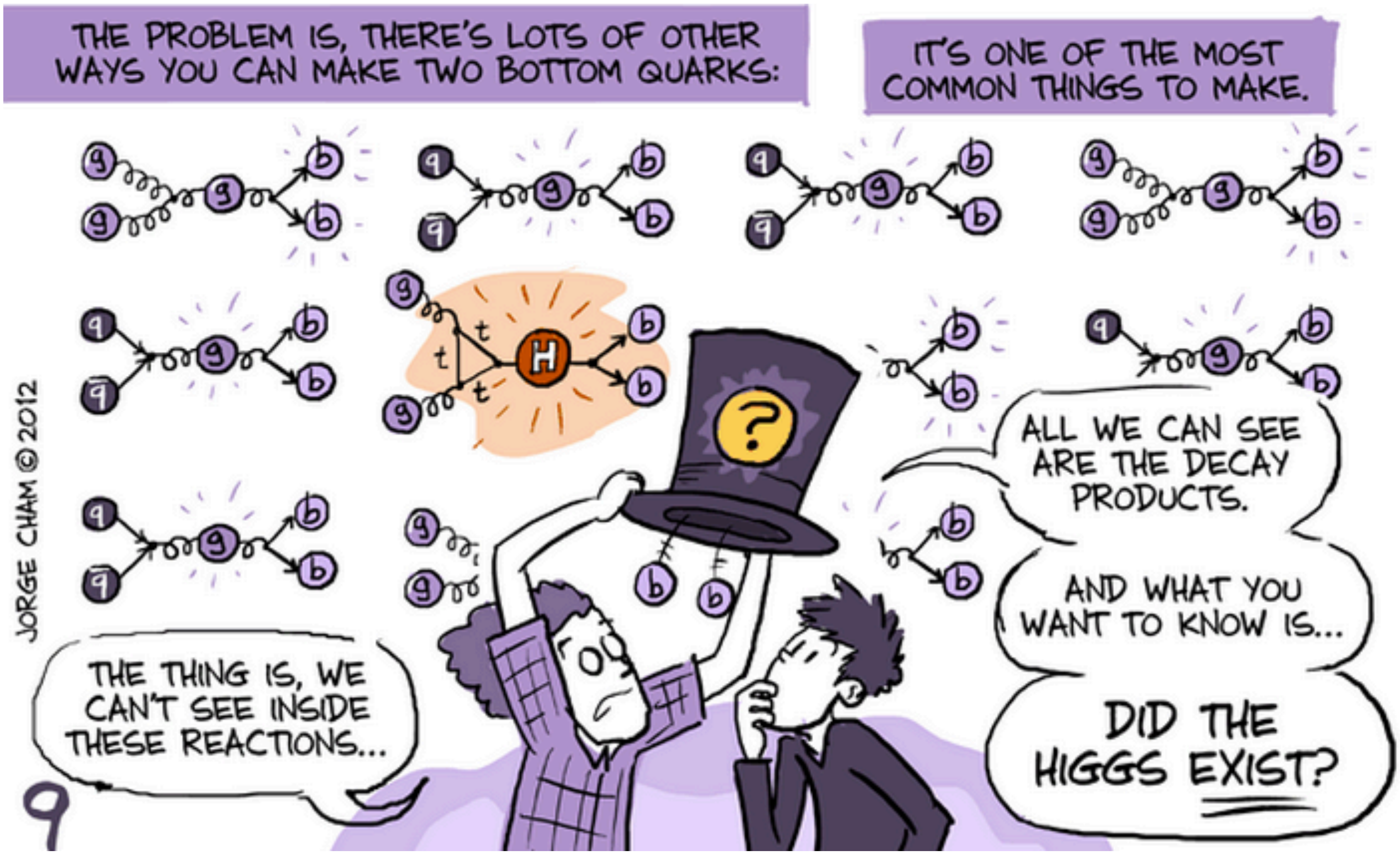


Unambiguous data

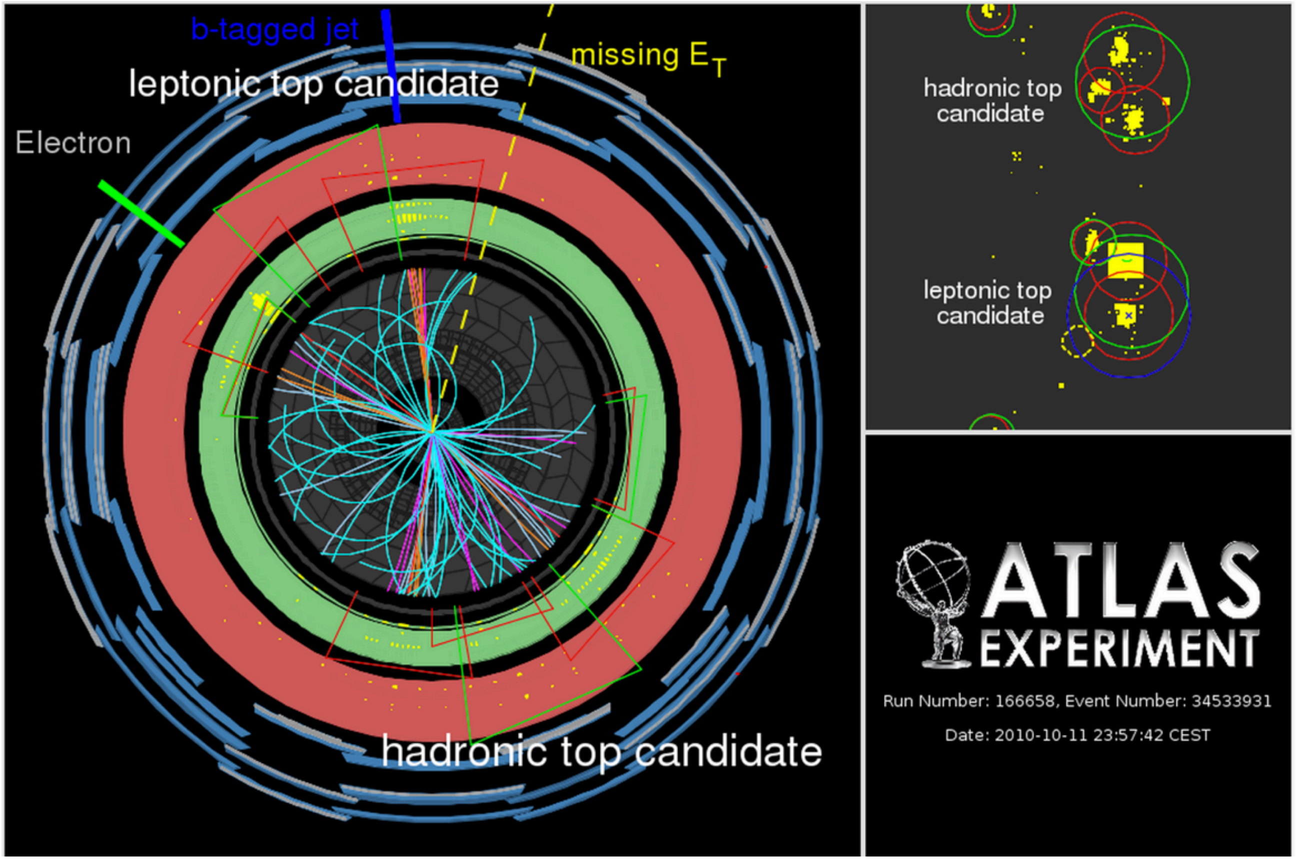
Ok, but see:

http://cerncourier.com/cws/article/cern/54388

Backgrounds

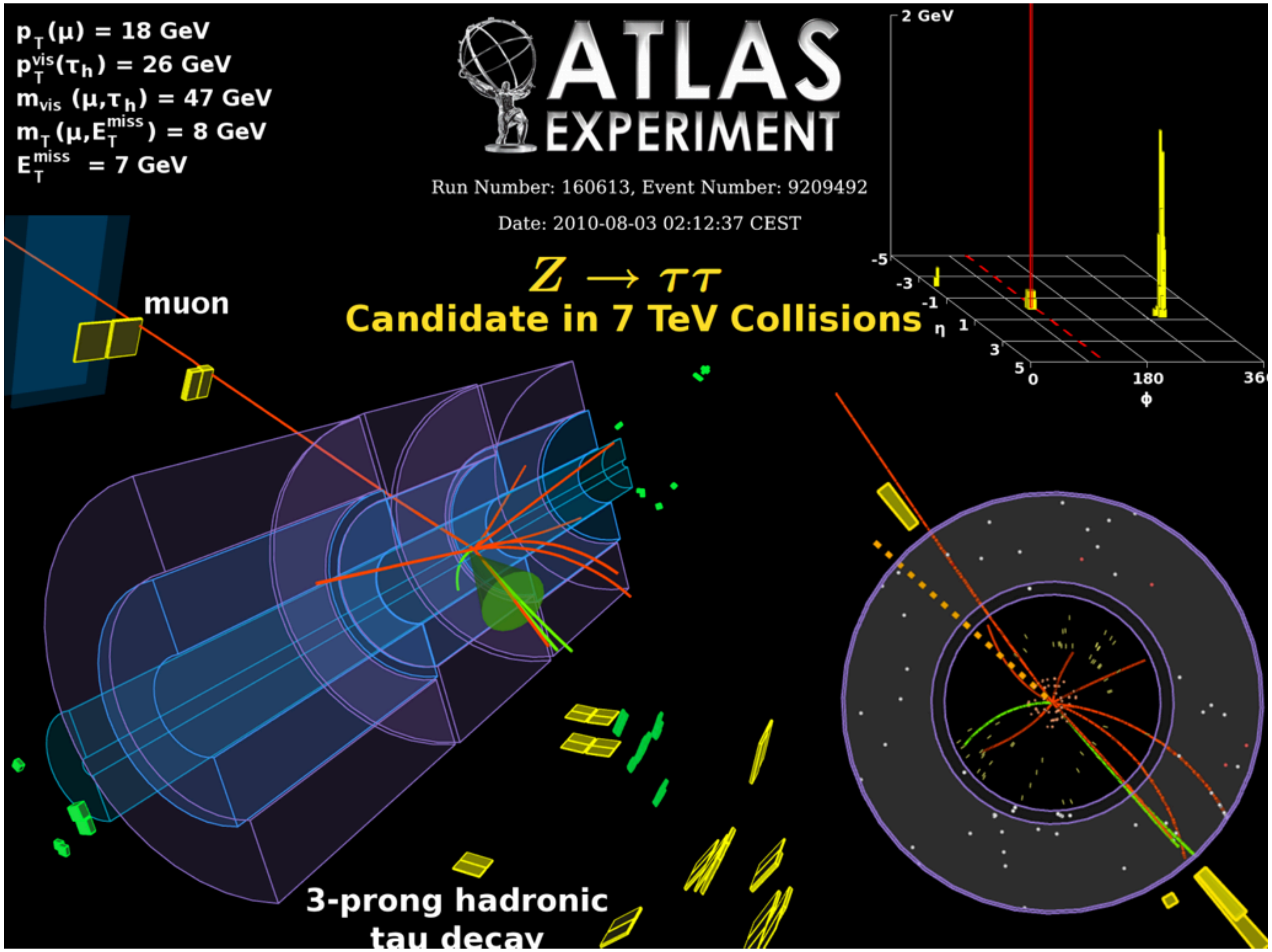


What's in an event?



No event can be unambiguously interpreted.

Why statistics?



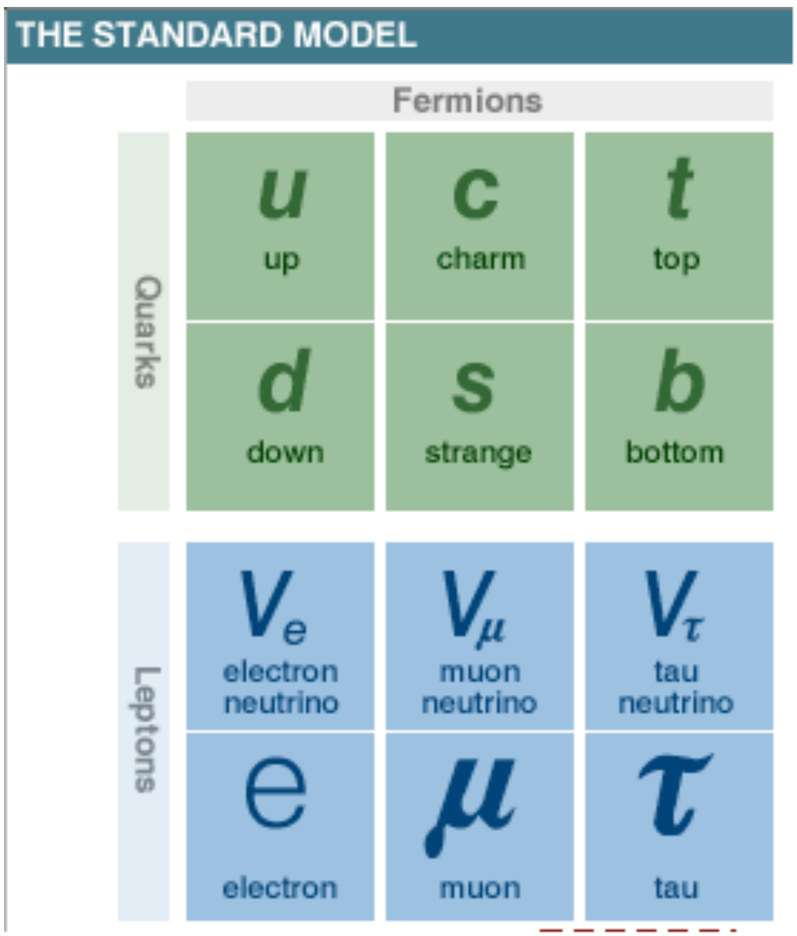
The nature of our data demands it.

Statistics for Discovery

Hypothesis testing

To search for a new particle, we compare the predictions of two hypotheses:

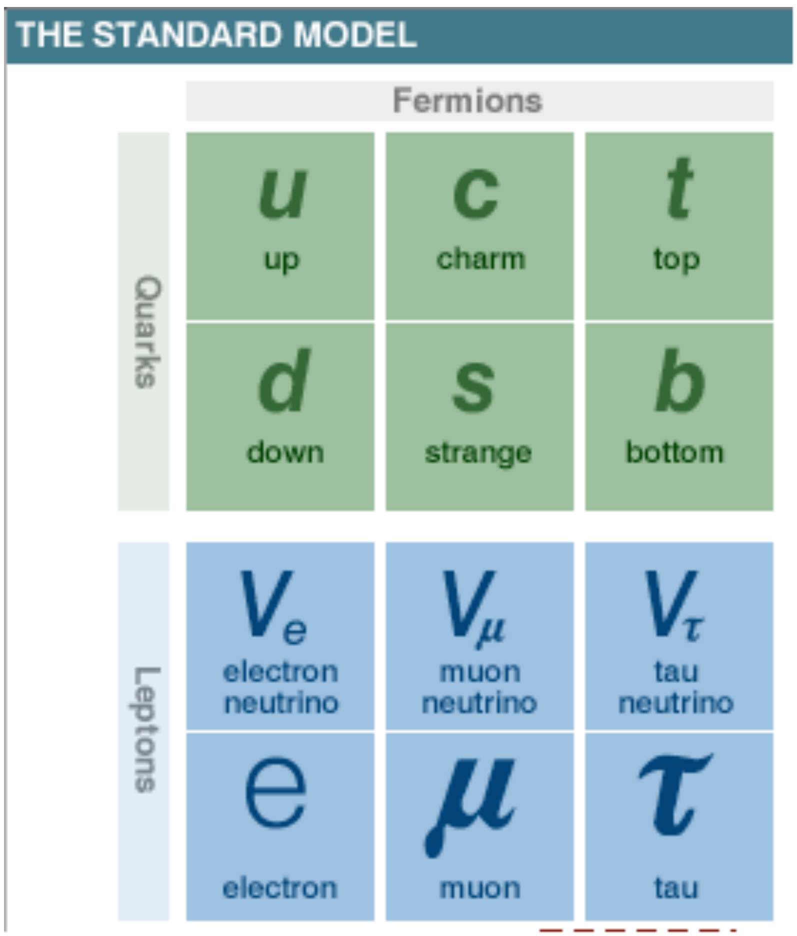
1.



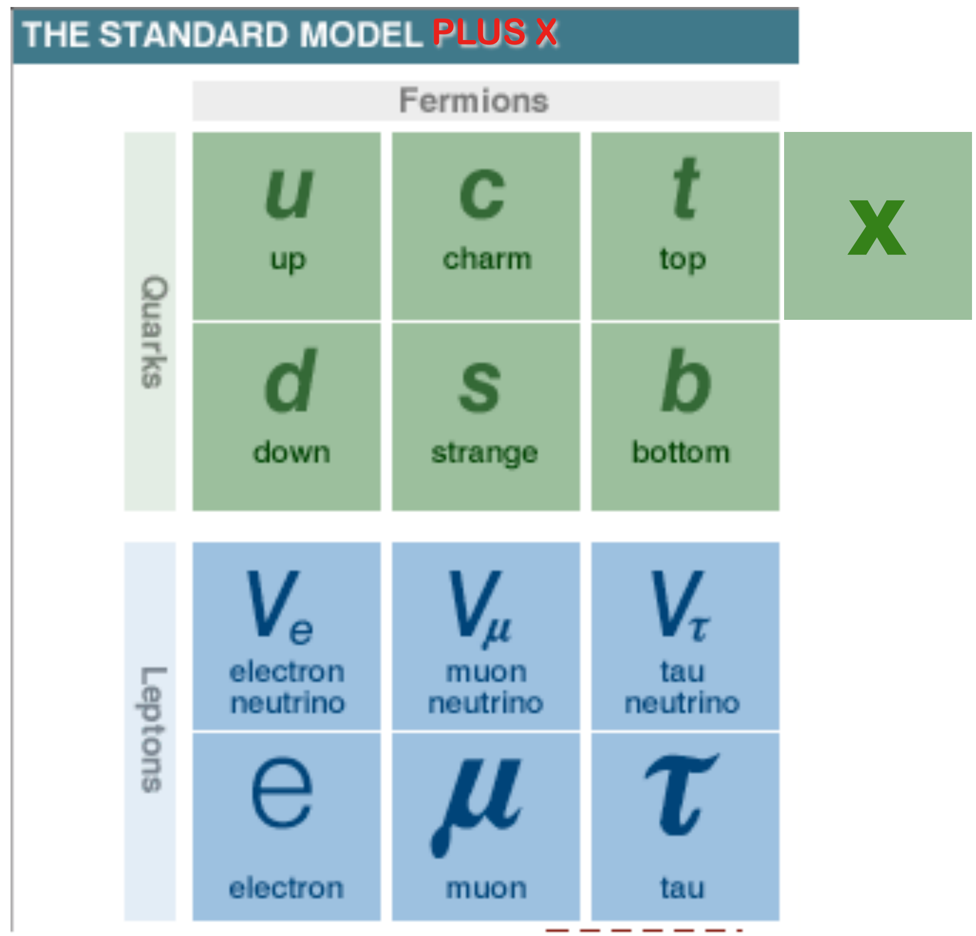
Hypothesis testing

To search for a new particle, we compare the predictions of two hypotheses:

1.



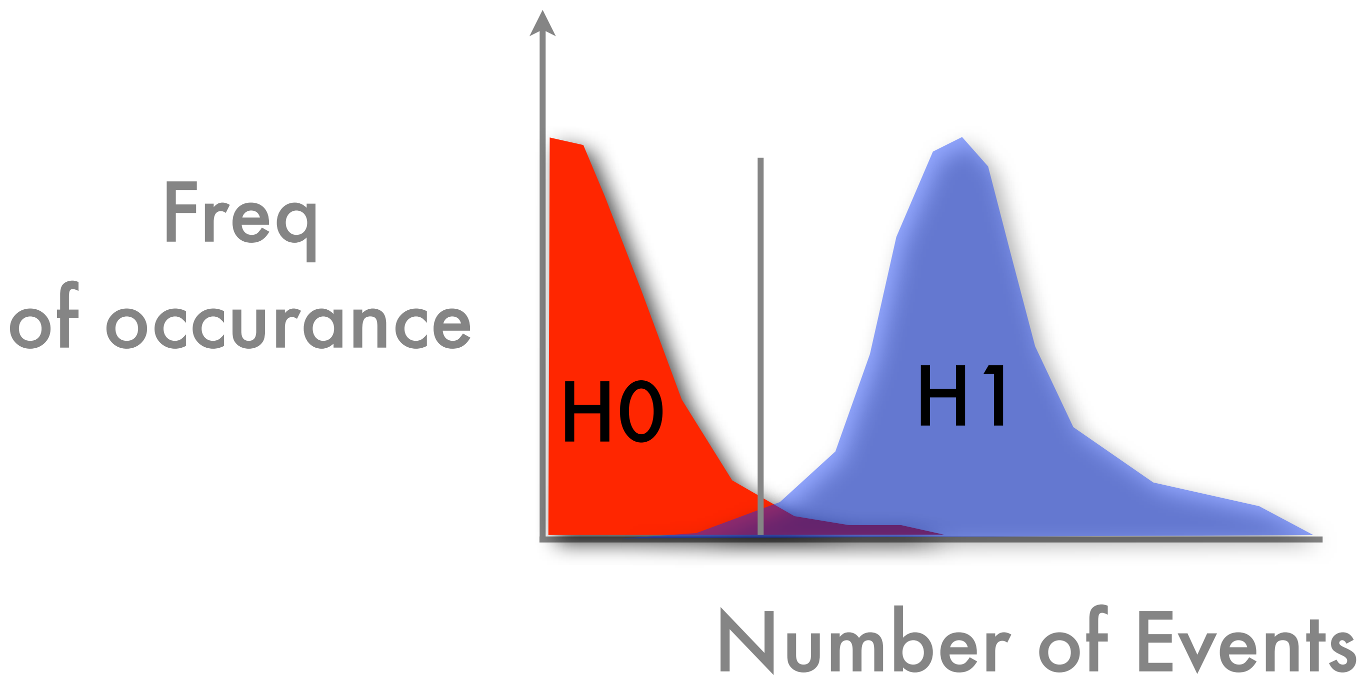
2.



Hypothesis Testing

	BSM Particle is real	Standard Model	
Claim BSM Discovery	True Positive	False Positive Type I error	α
No Claim of Discovery	False Negative Type II error	True Negative	
	β, power=1-β		

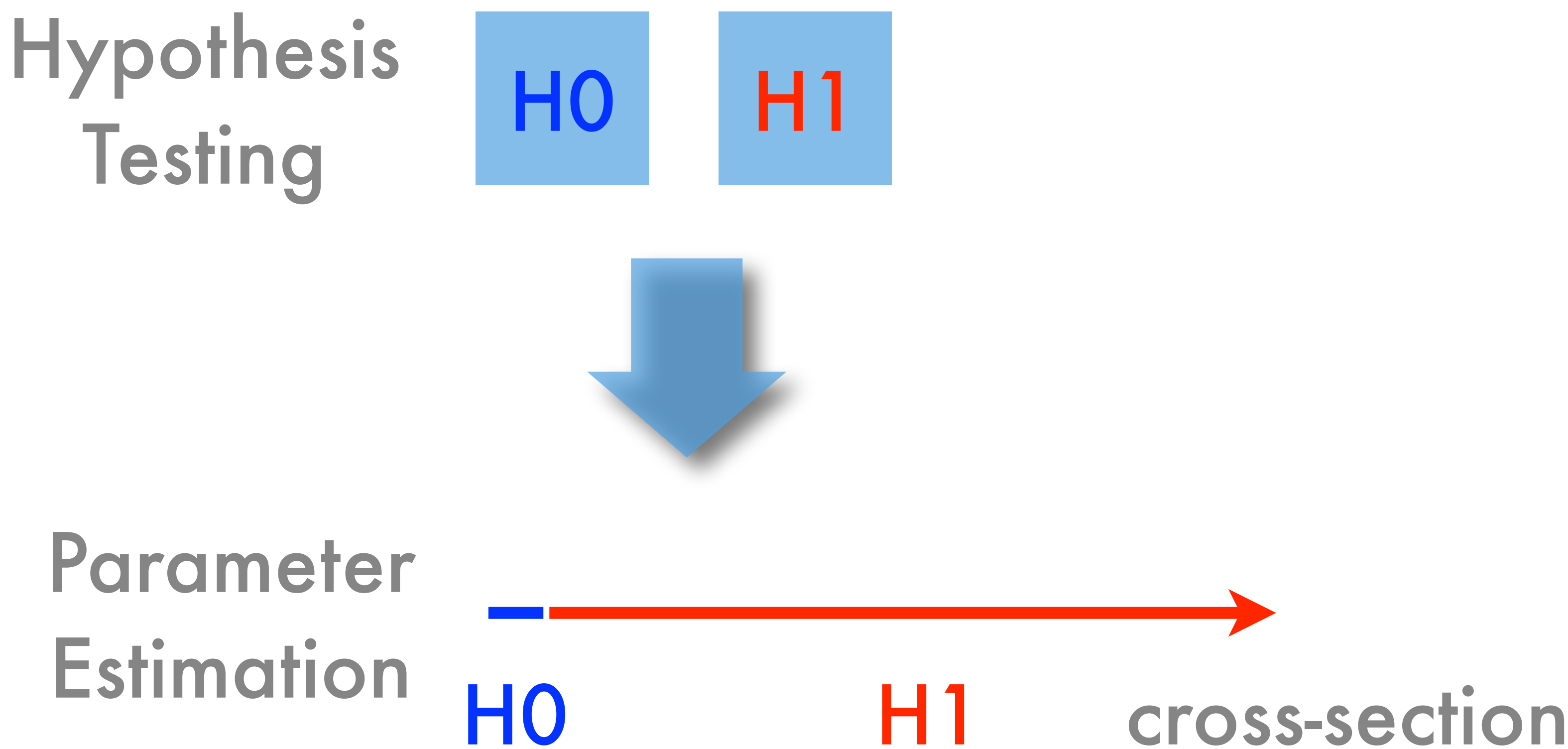
Example



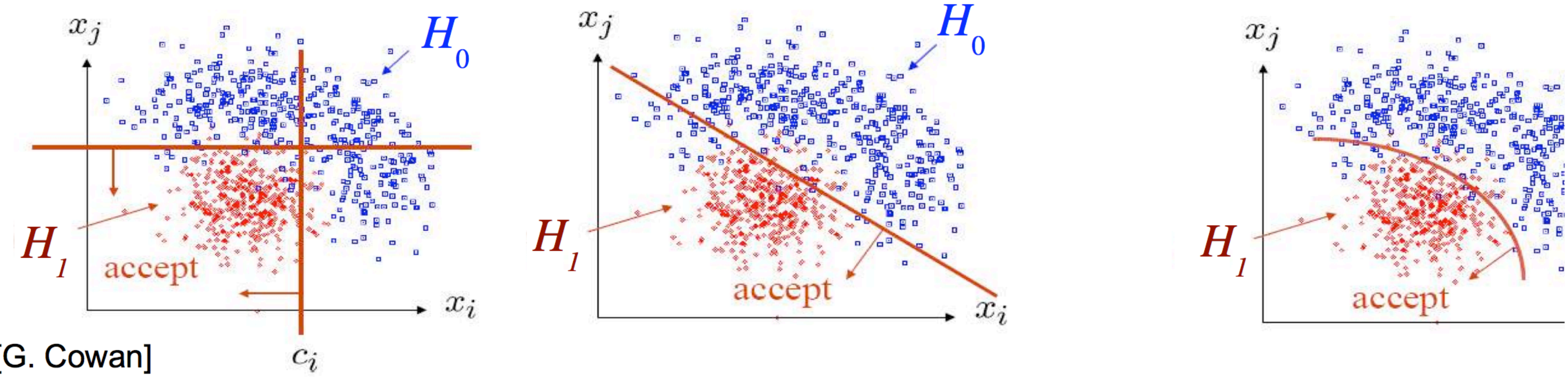
A threshold makes sense.
Choice of position balances
Type I/II errors

<u>Typically:</u>
fix α
minimize β

Generalize


Hypothesis Testing
H0
H1
Parameter Estimation
H0
H1
cross-section

More complicated



[G. Cowan]

Neyman-Pearson

Statement of the problem:

Given some prob that we wrongly reject the Null hypothesis

$$\alpha = P(x \notin W | H_0)$$

Find the region W (where we accept H_0) such that we minimize the prob

$$\beta = P(x \in W | H_1)$$

	BSM Particle is real	BSM Particle is not real	
Claim Discovery	True Positive	False Positive Type I error	α
No Claim of Discovery	False Negative Type II error	True Negative	
	β, power=1-β		

Neyman-Pearson

NP lemma says that the best
decision boundary is the likelihood ratio:

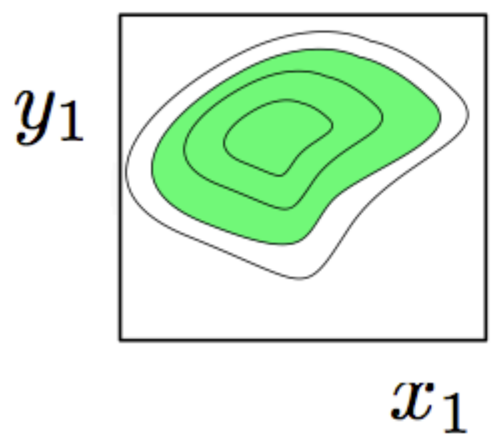
$$\frac{P(x|H_1)}{P(x|H_0)} > k_\alpha$$

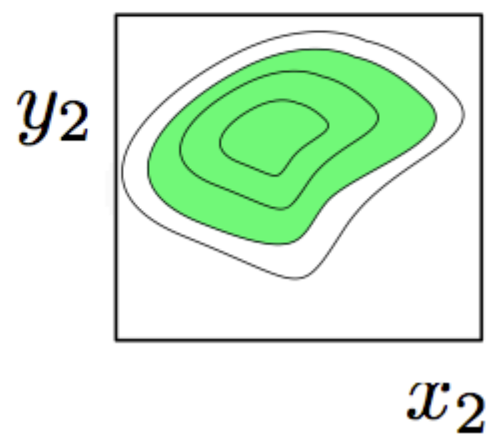
(Gives smallest β for fixed α)

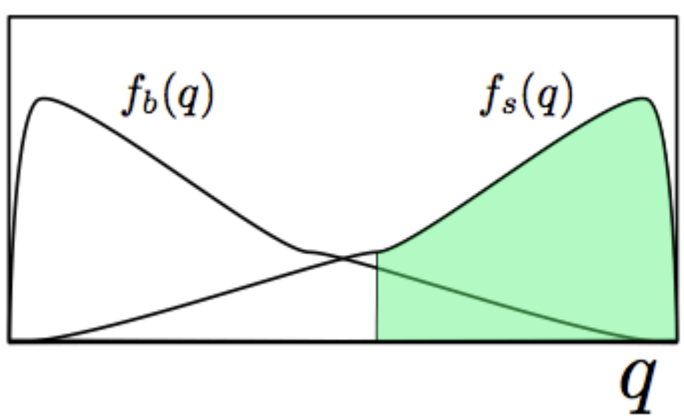
	BSM Particle is real	BSM Particle is not real	
Claim Discovery	True Positive	False Positive Type I error	α
No Claim of Discovery	False Negative Type II error	True Negative	
	β, power=1-β		

What does the TS do?

Finds a region in variable space



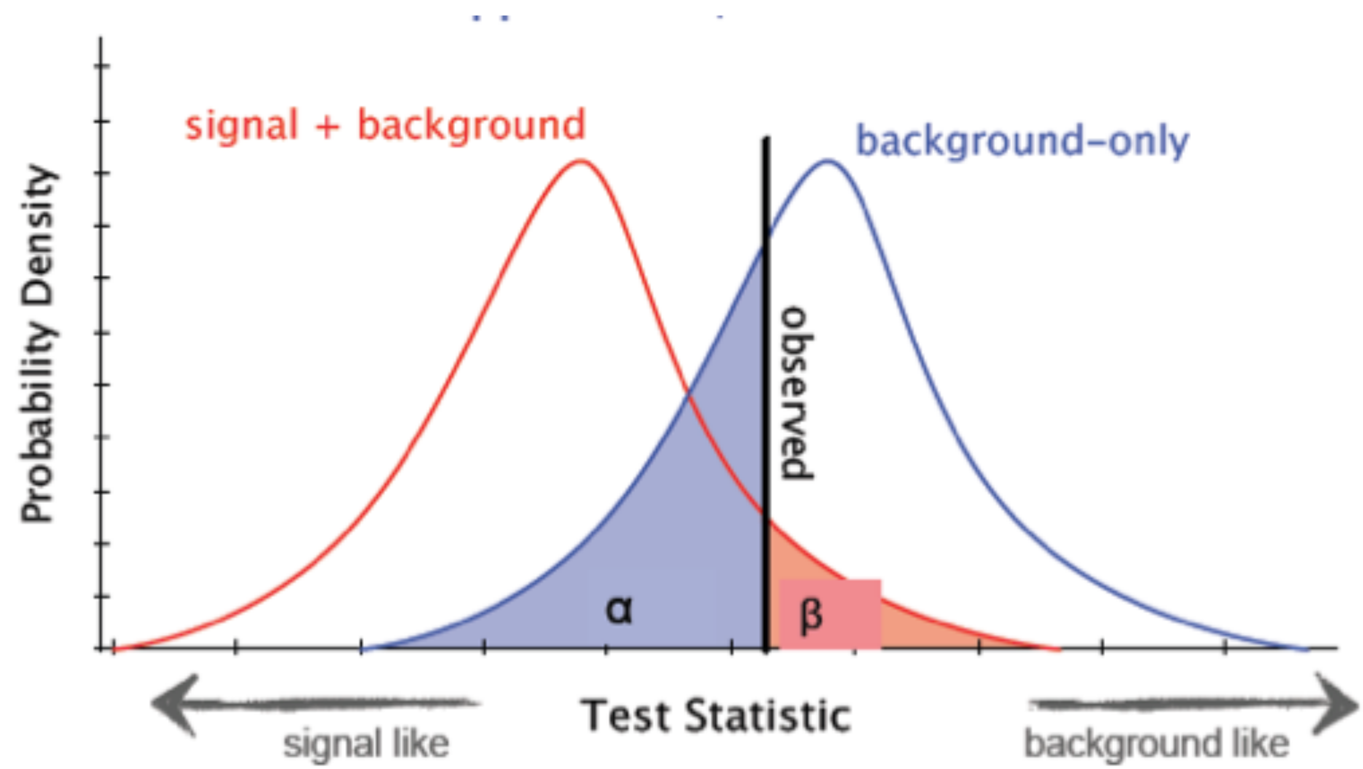




(K. Cranmer)

Test statistic

Reduce vector of observables to 1 number



How to choose TS?

(K. Cranmer)

No problem

Fairly easy to find test statistic

if you can calculate

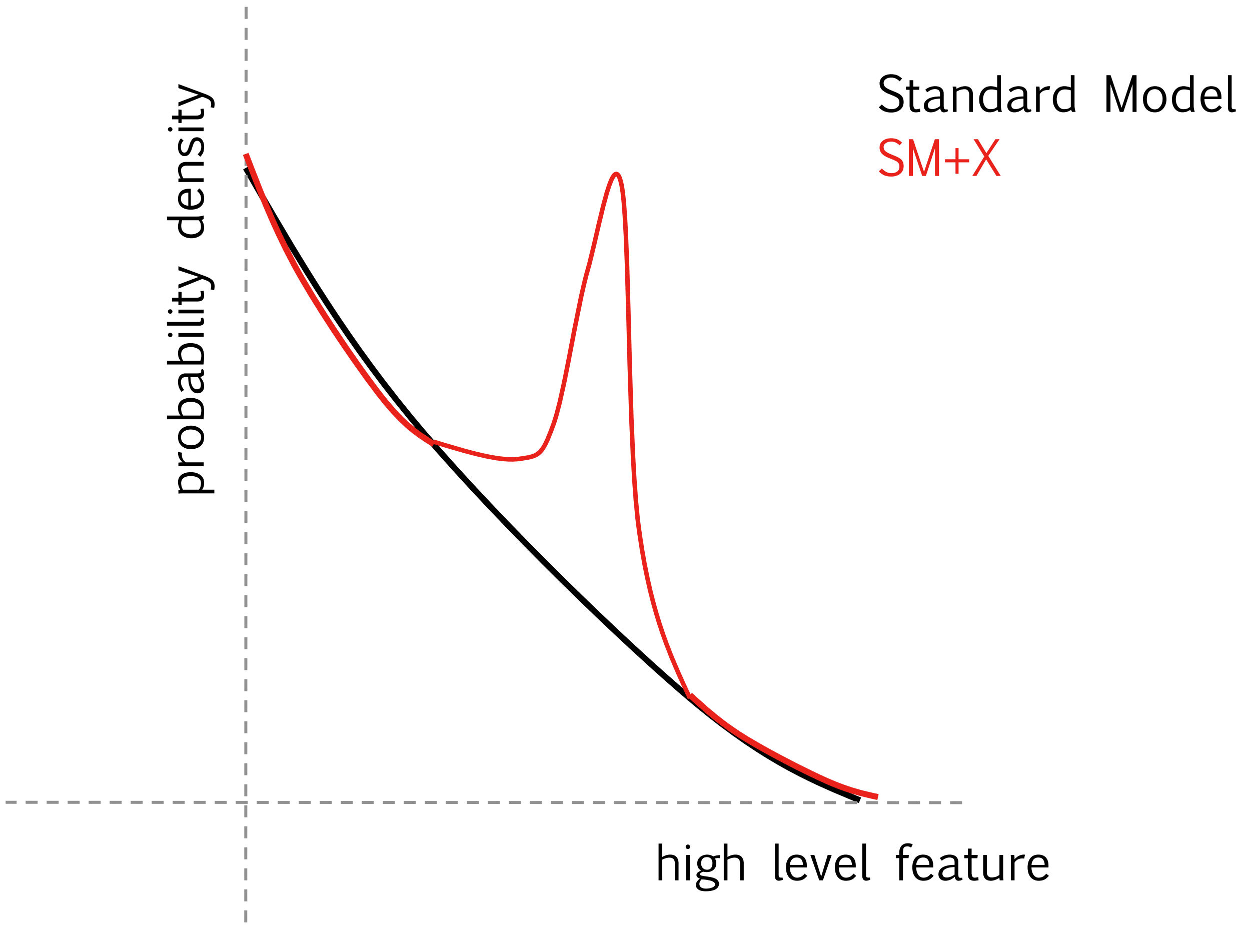
$P(x|H1)$, $P(x|H0)$

or generally

$P(data|theory)$

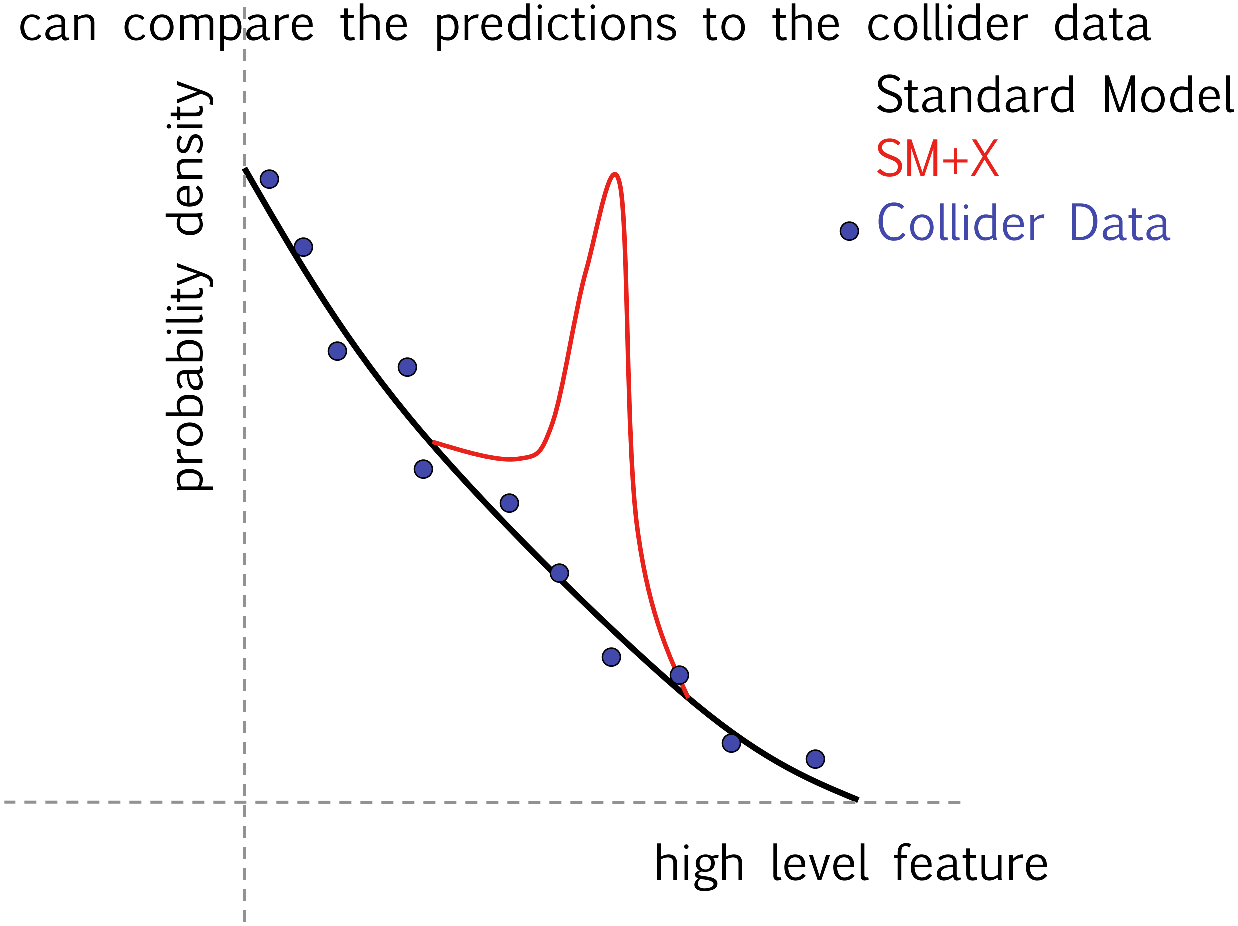
Hypothesis Testing

Sometimes this is easy



Hypothesis Testing

We can compare the predictions to the collider data



Which can tell us which hypothesis is preferred via a likelihood ratio:

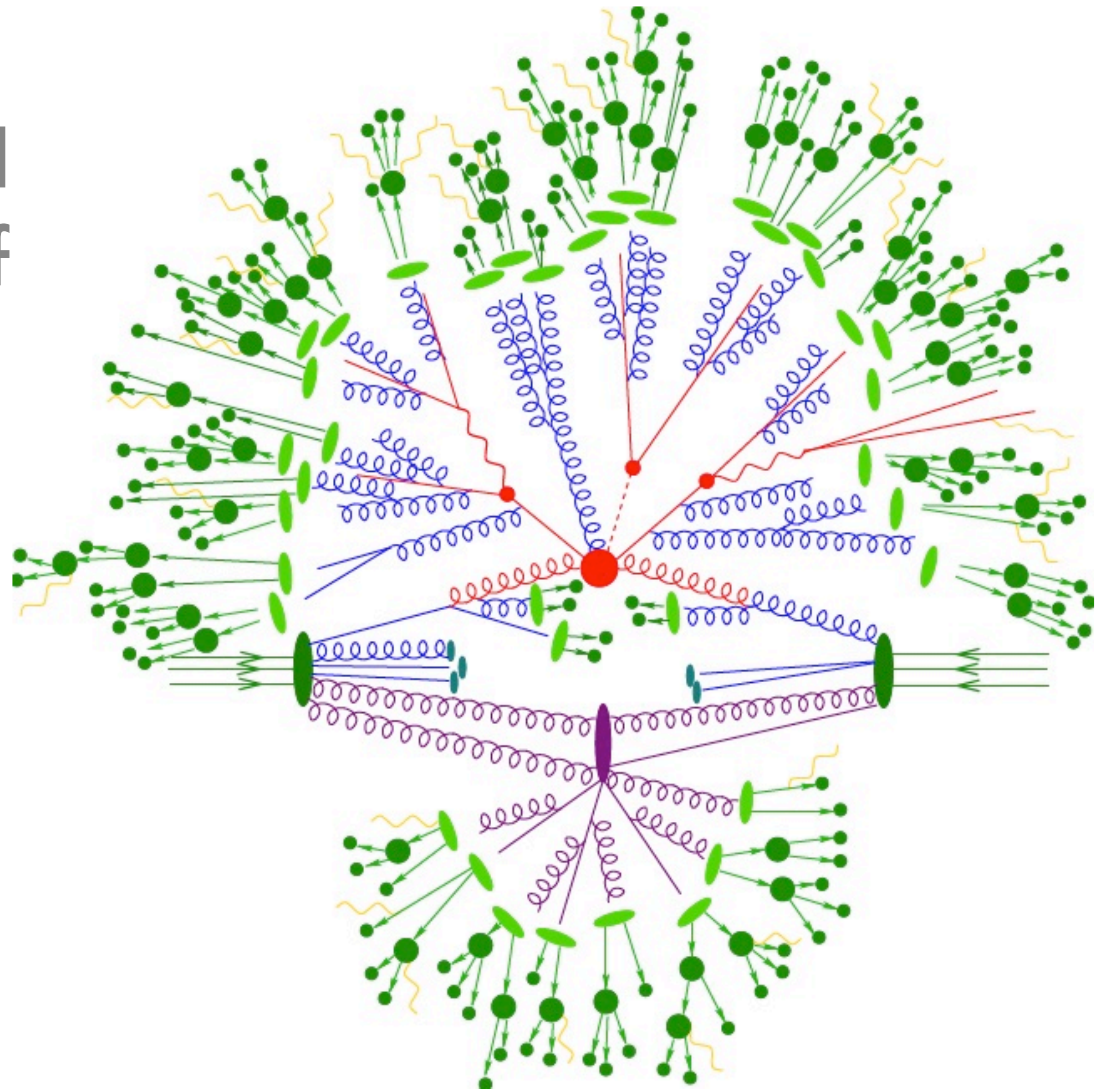
$$\frac{L_{SM+X}}{L_{SM}} = \frac{P(\text{data} \mid SM+X)}{P(\text{data} \mid SM)}$$

In general

We have a good understanding of of the pieces

Do we have

f(data|theory)?

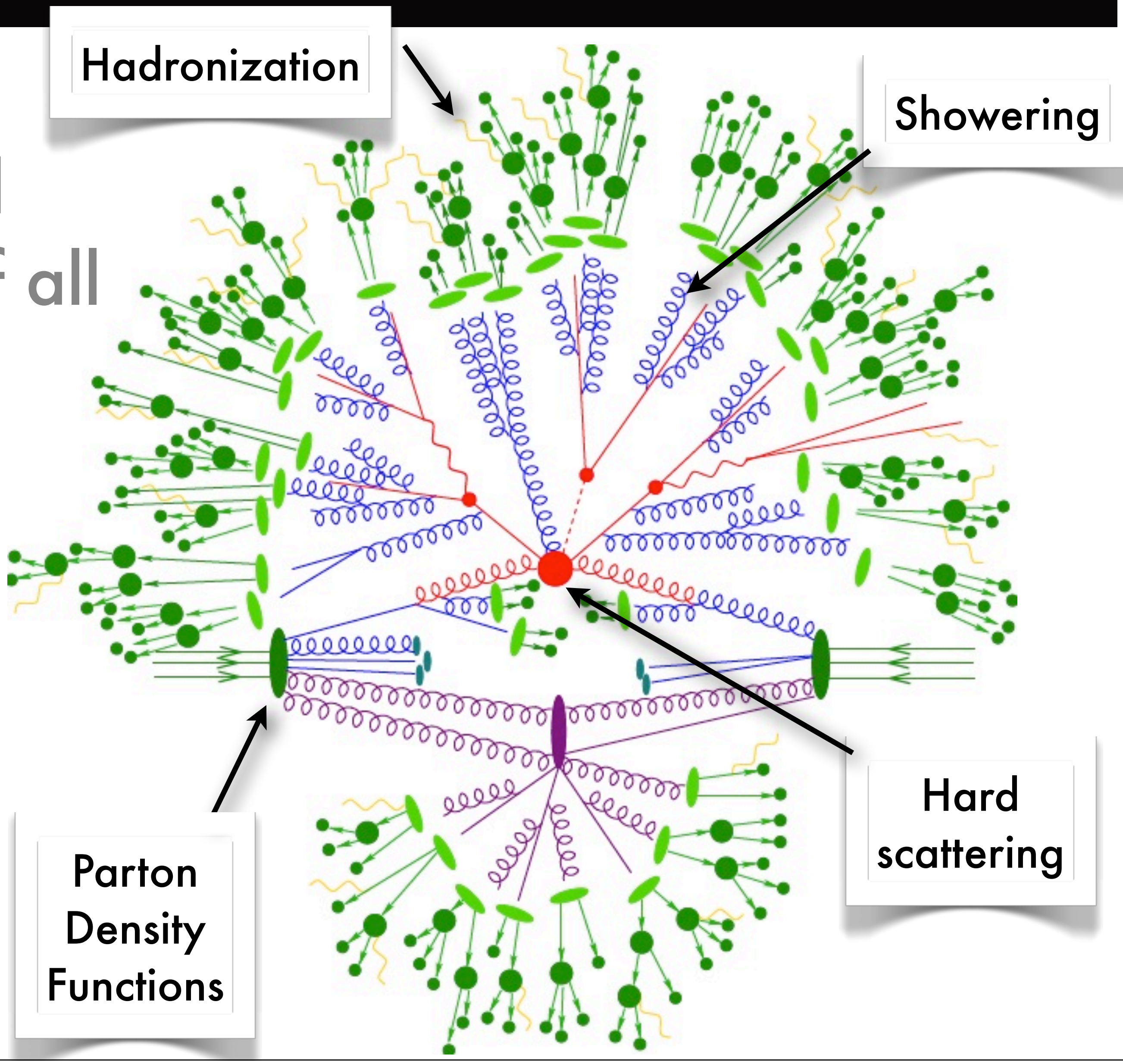


In general

We have a good understanding of all of the pieces

Do we have

f(data|theory)?



In general

What would

$f(data|theory)$

look like?

The dream

f(data | final-state particles P)

x f(final state particles P | showered particles S)

x f(showered particles S | hard scatter products M)

x f(hard scatter products M | theory)

Sum over all possible intermediate P,S,M

The dream

Detector Response

f(data | final-state particles P)

Hadronization

x f(final state particles P | showered particles S)

Showering

x f(showered particles S | hard scatter products M)

x f(hard scatter products M | theory)

Parton Density Functions

Hard scattering

Sum over all … e intermediate P,S,M

The dream

f(hard scatter products M | theory)

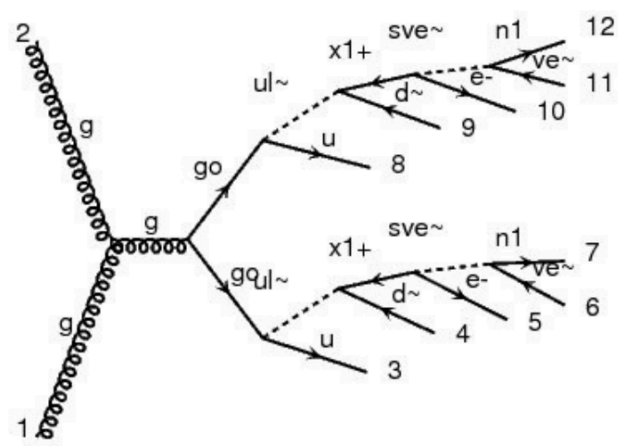


diagram 1

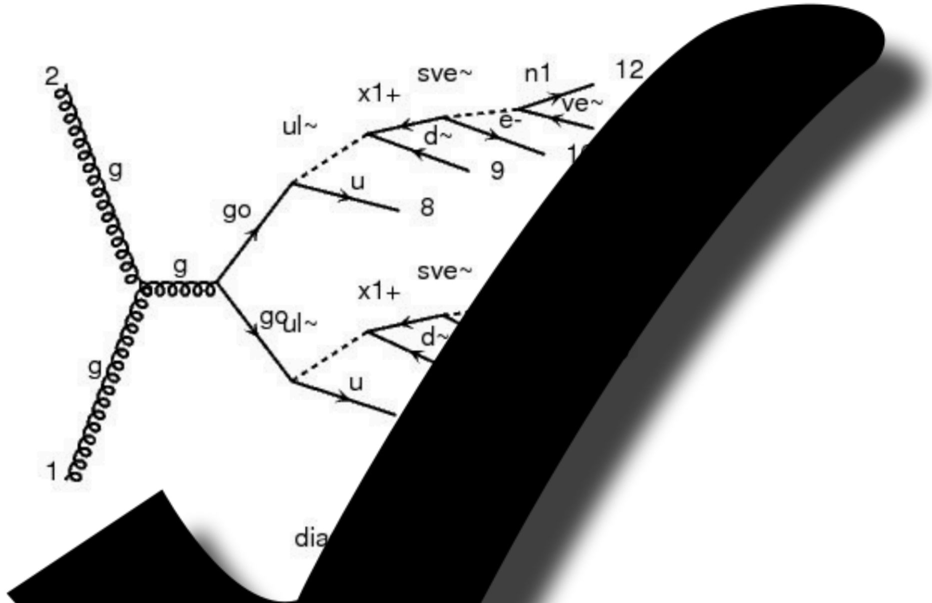
Theory well defined

automatic calculators exist

for almost any (B)SM theory

The dream

f(hard scatter products M | theory)



The ... defined

automat... ulators exist

for almost any (B)SM theory

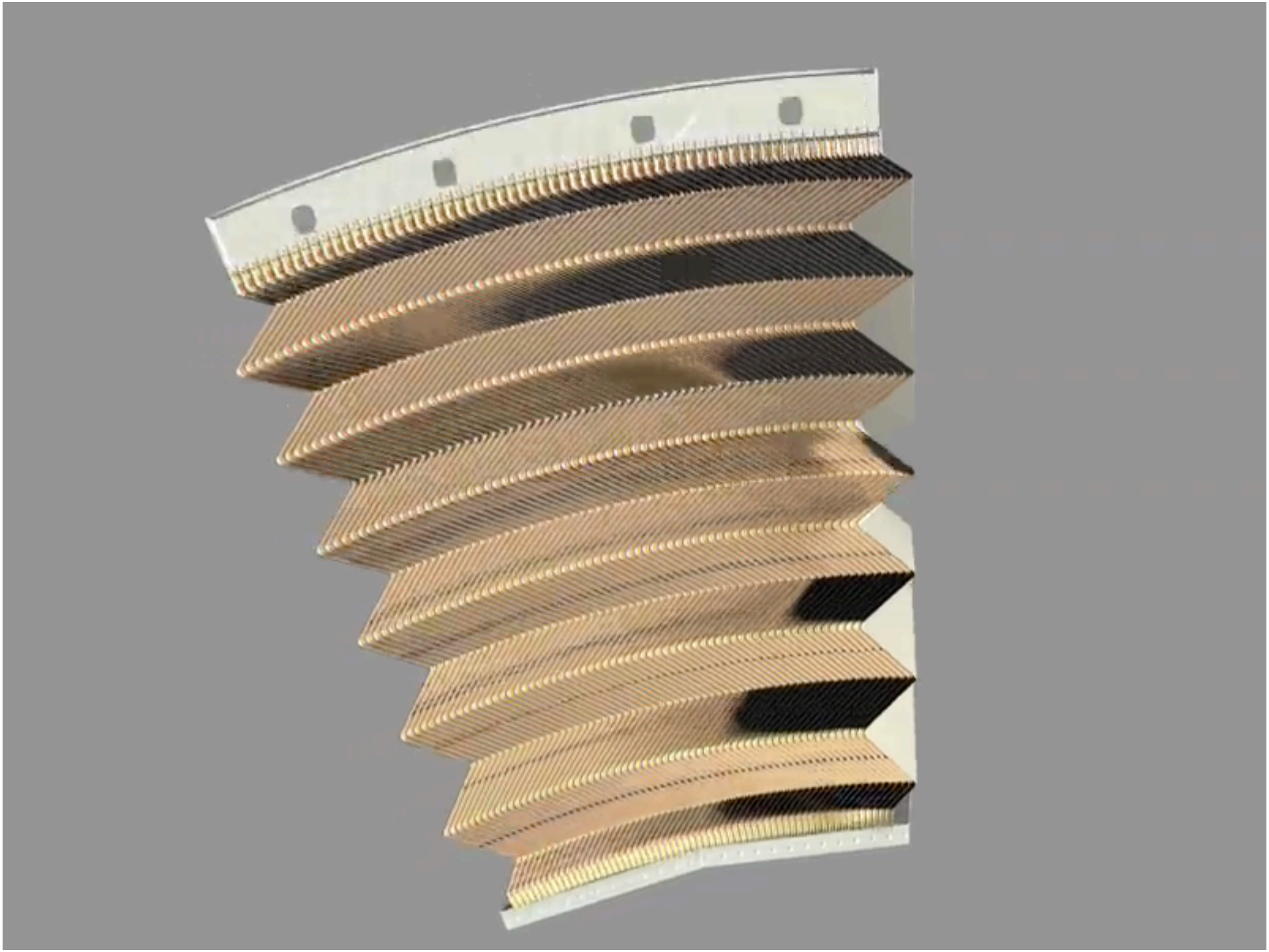
The nightmare

f(data | final-state particles P)

x f(final state particles P | showered particles S)

x f(showered particles S | hard scatter products M)

<u>We have</u>: solid understanding of microphysics
<u>We need</u>: analytic description of high-level physics



The solution

<u>We have</u>: solid understanding of microphysics
<u>We need</u>: analytic description of high-level physics
<u>But</u>: only heuristic lower-level approaches exist

Iterative simulation strategy, no overall PDF

<u>Iterative approach</u>
(1) Draw events from f(M|theory)
(2) add random showers
(3) do hadronization
(4) simulate detector

The solution

We have: solid understanding of microphysics
We need: analytic description of high-level physics
But: only heuristic lower-level approaches exist

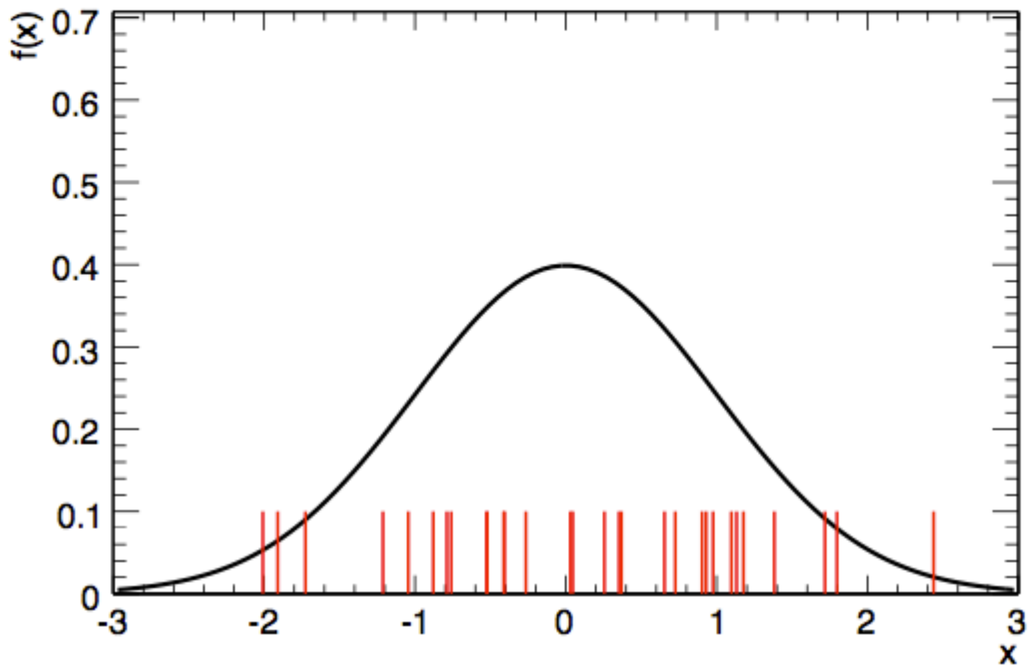
Iterative simulation strategy, no overall PDF

What do we get
Arbitrarily large samples of events drawn from f(data|theory), but not the PDF itself

The problem

Don't know PDF, have events drawn from PDF

$$f_{emp} = \frac{1}{N}\sum_{i}^{N}\delta(x - x_i)$$



(K. Cranmer)

Need to recreate PDF

What do we need?

Want:

our model of the expected results of the experiment

f(data | theory)

Provides:

- PDF for data as a function of POI, NPs
- generate pseudo-data
- fix data to get lhood

We have:

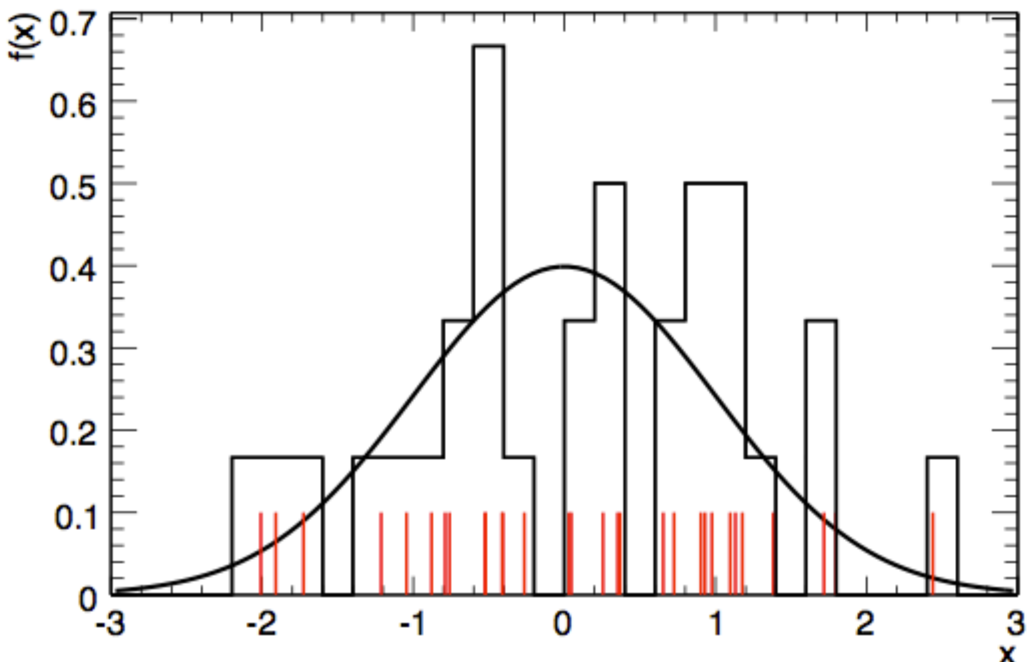
A tool that can generate sample event data

How do we use that to build our PDF?

MC events to PDF

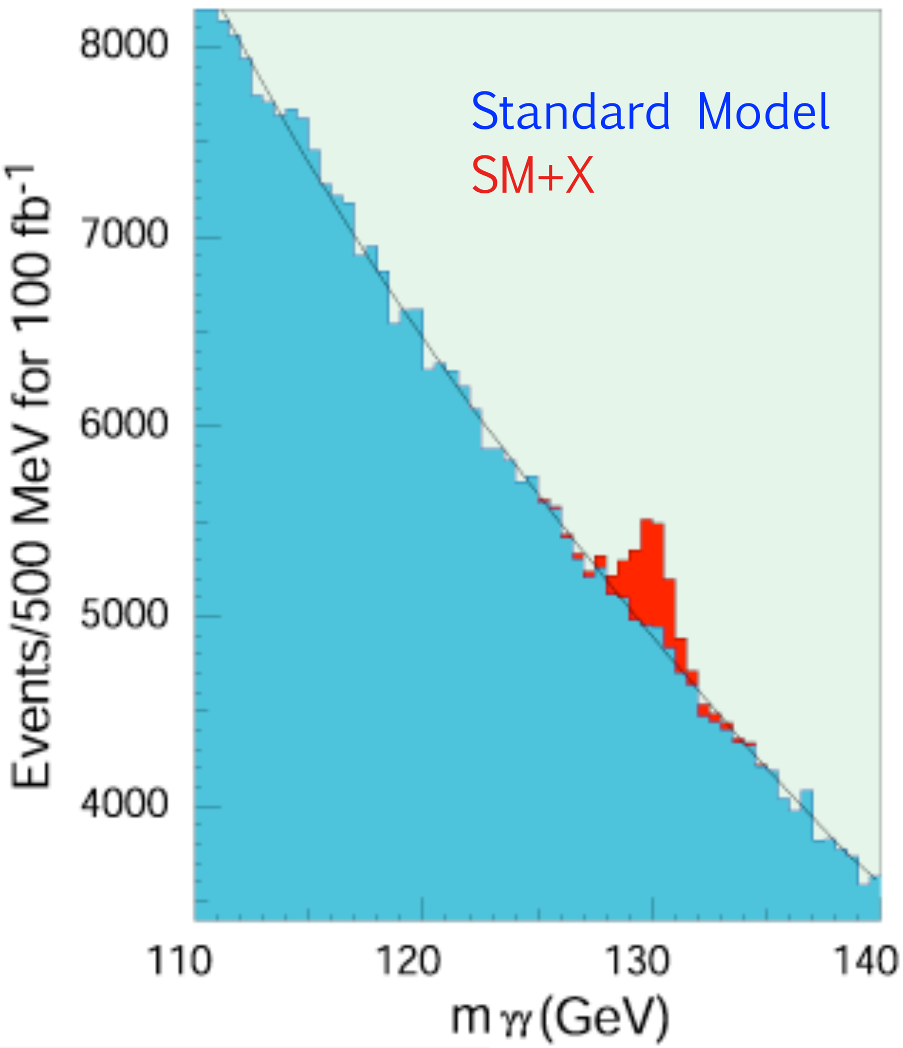
Simple approach : histogram

$$f^{w,s}_{hist}(x) = \frac{1}{N}\sum_i h^{w,s}_i$$



(K. Cranmer)

Example



Use histograms to define

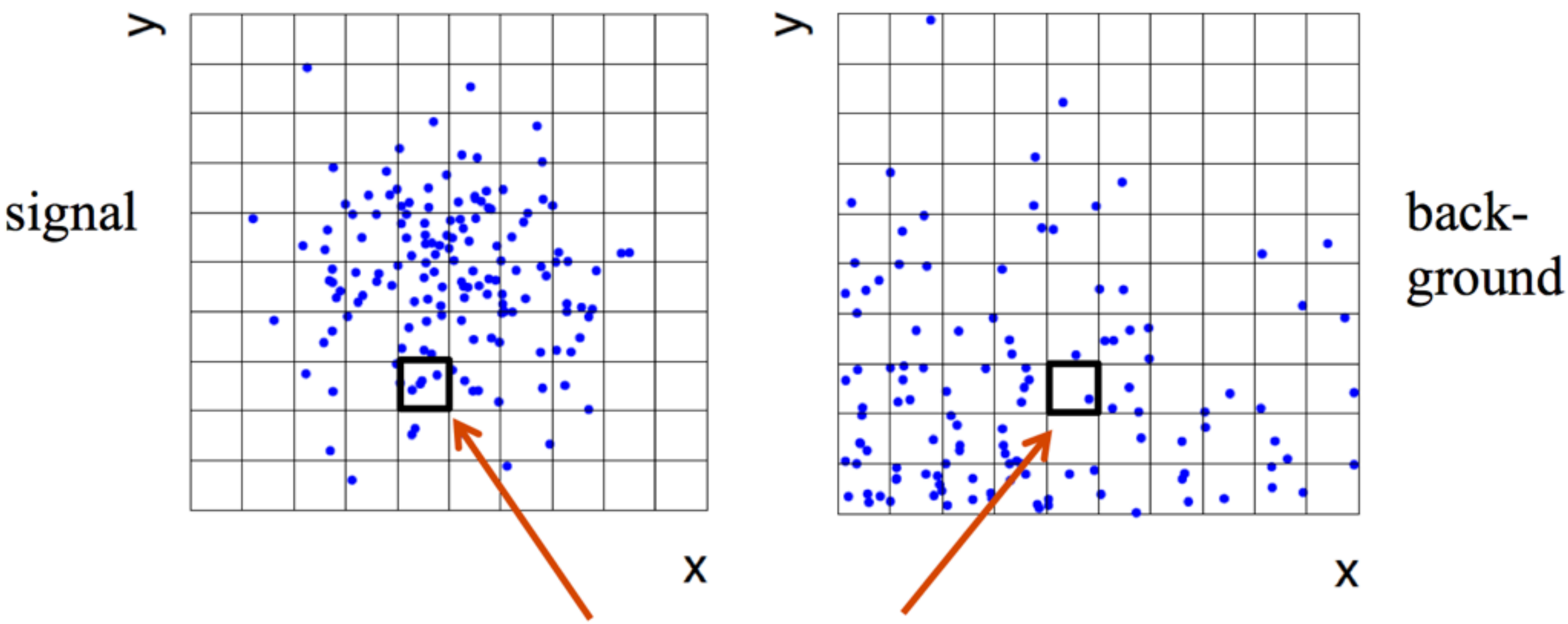
P(data | SM+X),

and

P(data | SM)

Approximate LR from 2D-histograms

Suppose problem has 2 variables. Try using 2-D histograms:



Approximate pdfs using $N(x,y|\mathrm{s})$, $N(x,y|\mathrm{b})$ in corresponding cells.

But if we want M bins for each variable, then in n-dimensions we have M^n cells; can't generate enough training data to populate.

$\rightarrow$ Histogram method usually not usable for $n > 1$ dimension.

Curse of Dimensionality

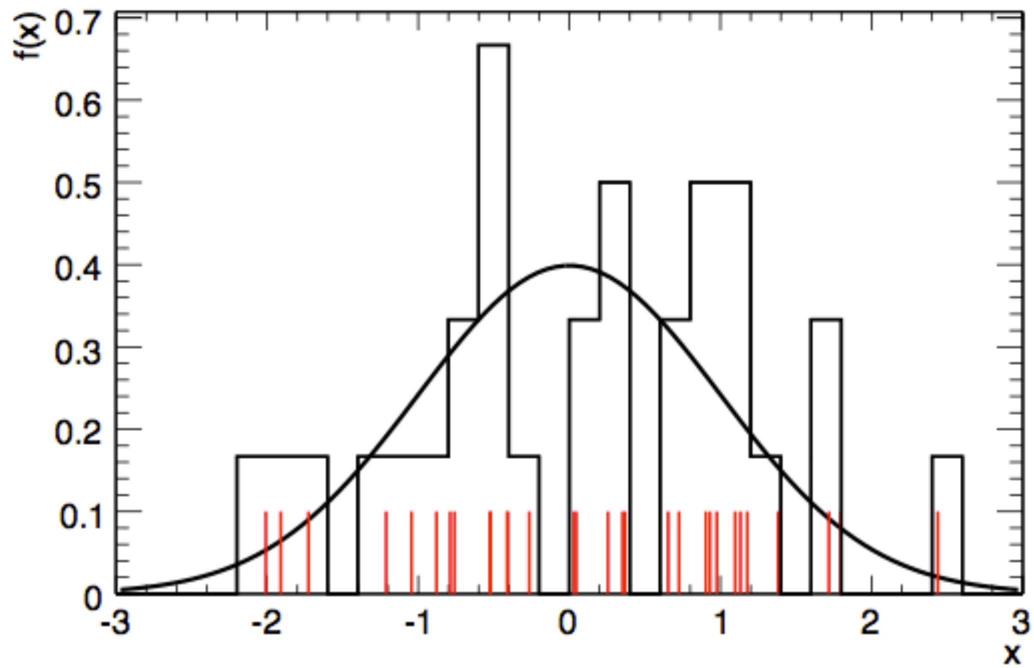
How many events
do you need
to describe a 1D
distribution? O(100)

An n-D distribution?

$O(100^n)$

!!

$$f_{hist}^{w,s}(x) = \frac{1}{N}\sum_i h_i^{w,s}$$



(K. Cranmer)

The nightmare

f(data | final-state particles P)

x f(final state particles P | showered particles S)

x f(showered particles S | hard scatter products M)

“data” is a 100M-d vector!

The nightmare

f(data | final-state particles P)

x f(final stat

red particles S)

x f(showered

atter products M)

"

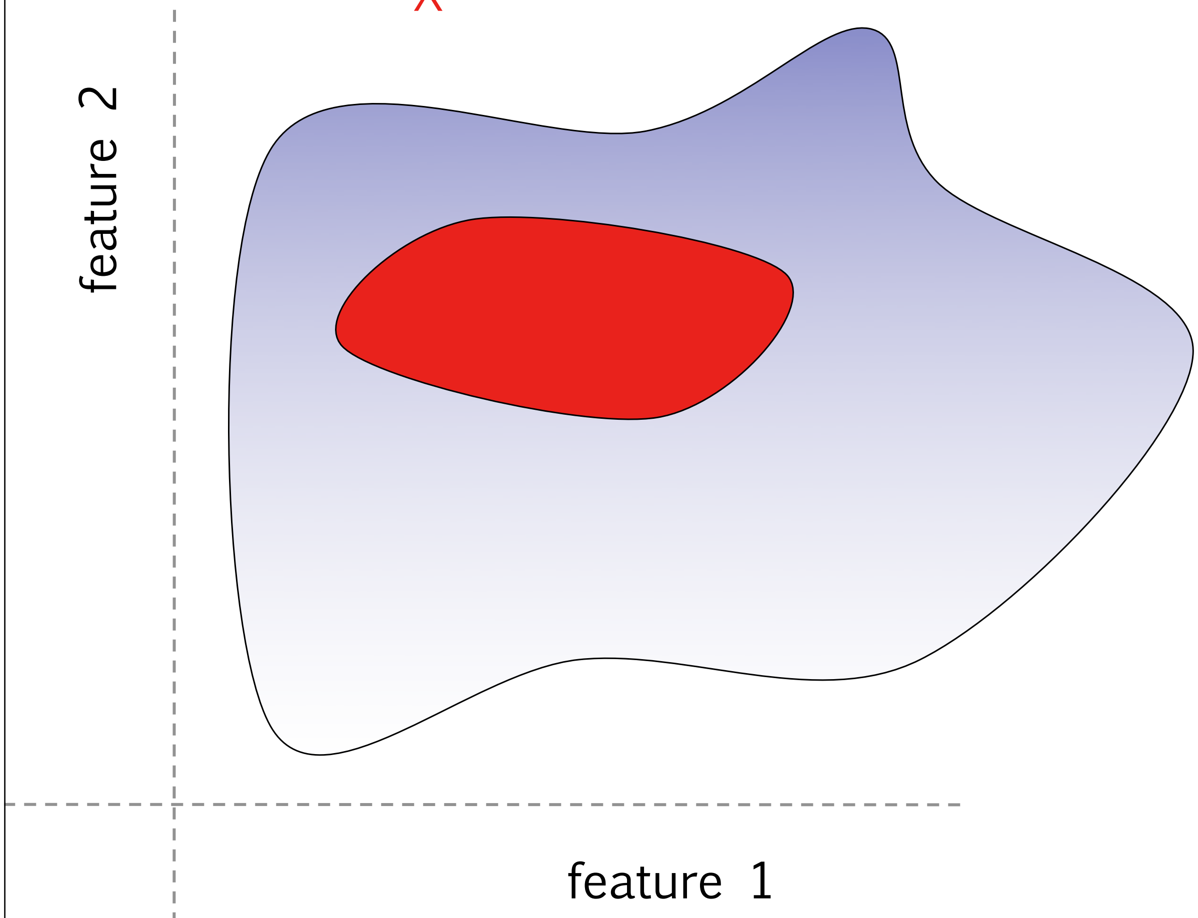
vector!

Machine Learning

Standard Model

X

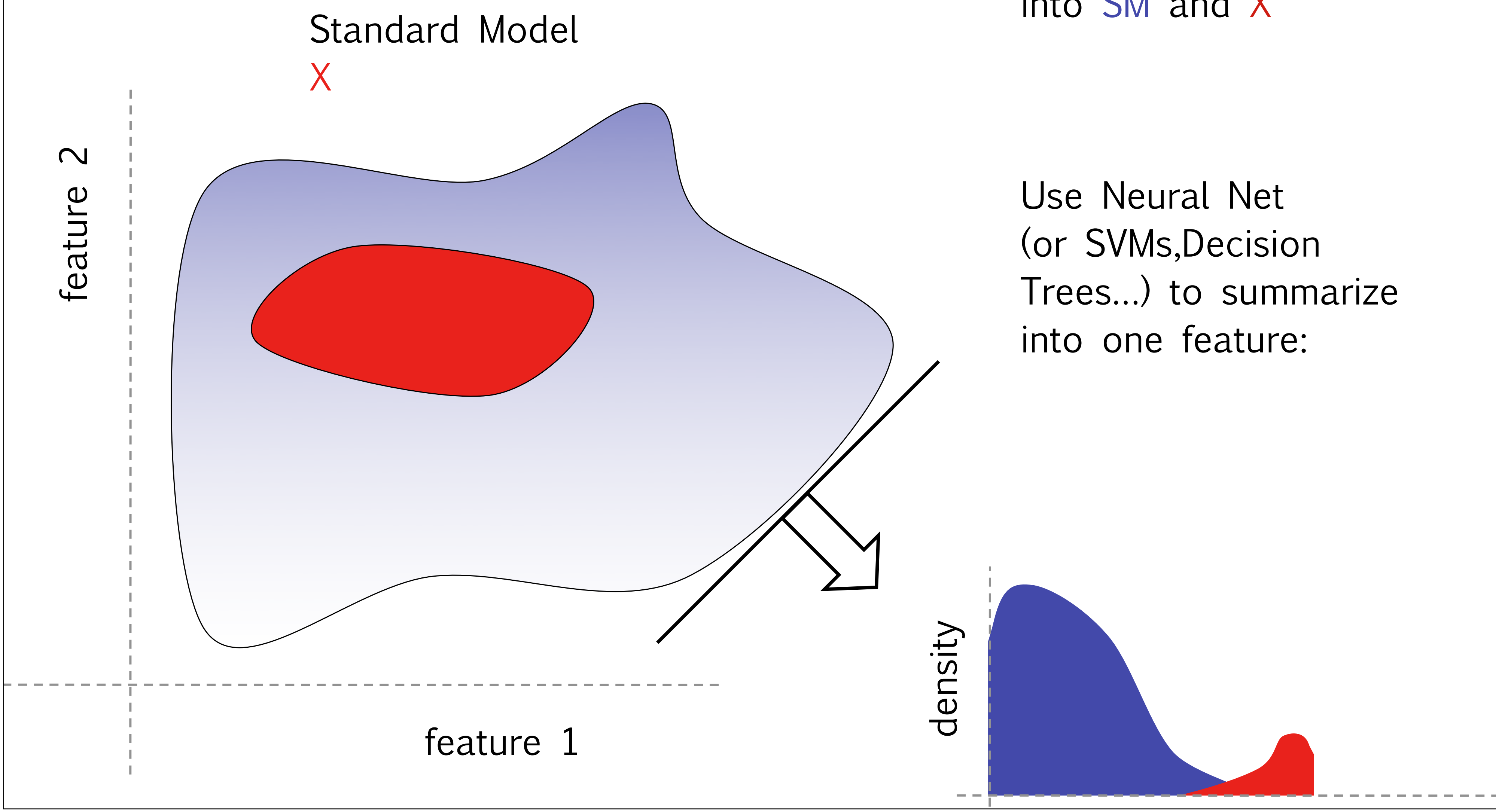
Separate SM+X
into SM and X



Machine Learning

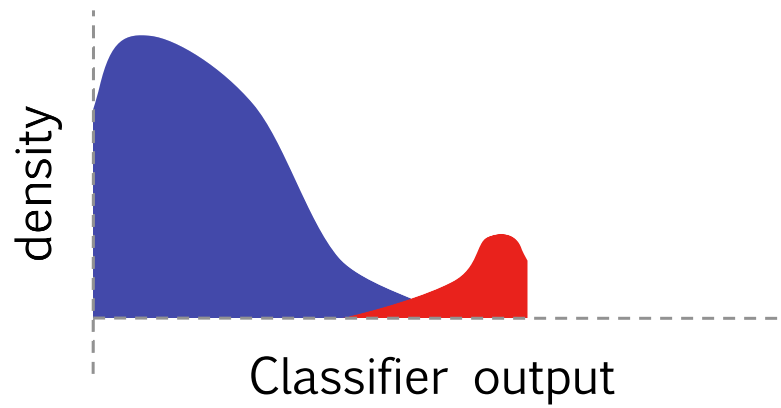
Separate SM+X
into SM and X

Use Neural Net
(or SVMs,Decision
Trees...) to summarize
into one feature:



Dimensional Reduction

This dimensional reduction can be very helpful.



Summarize the differences between the hypotheses

$$\frac{L_{SM+X}}{L_{SM}} = \frac{P(\text{data} \mid SM+X)}{P(\text{data} \mid SM)}$$

And require a histogram in only one dimension

No problem

Fairly easy to find test statistic
if you can calculate

$P(x|H1)$, $P(x|H0)$

or generally

$P(data|theory)$

or: use ML to reduce data to 1-dimension

Task for ML

Find a function:

$$f(\bar{x}) : \mathbb{R}^N \to \mathbb{R}^1$$

which contains the same hypothesis testing power as

$$\frac{P(x|H_1)}{P(x|H_0)} > k_\alpha$$

ML approaches

1. Kernel methods

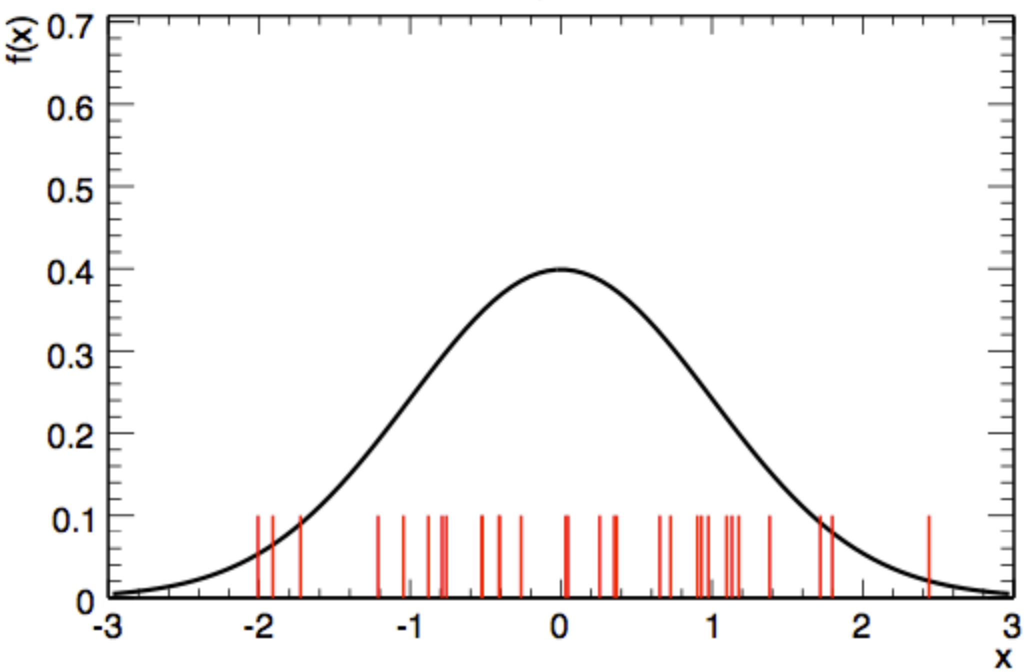
2. Neural Networks

3. Support Vectors

Kernel Methods

Generalized histogram

$$f_{emp} = \frac{1}{N}\sum_{i}^{N}\delta(x - x_i)$$



Revisit

Can we be smarter than this?

Rather than use a delta function, use a smoother blob

Kernel density estimate

Consider d dimensions, N training events, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, estimate $f(\boldsymbol{x})$ with

$\boldsymbol{x}$ where we want to know pdf

$\boldsymbol{x}$ of i^{th} training event

$$\hat{f}(\vec{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\vec{x} - \vec{x}_i}{h}\right)$$

kernel

bandwidth (smoothing parameter)

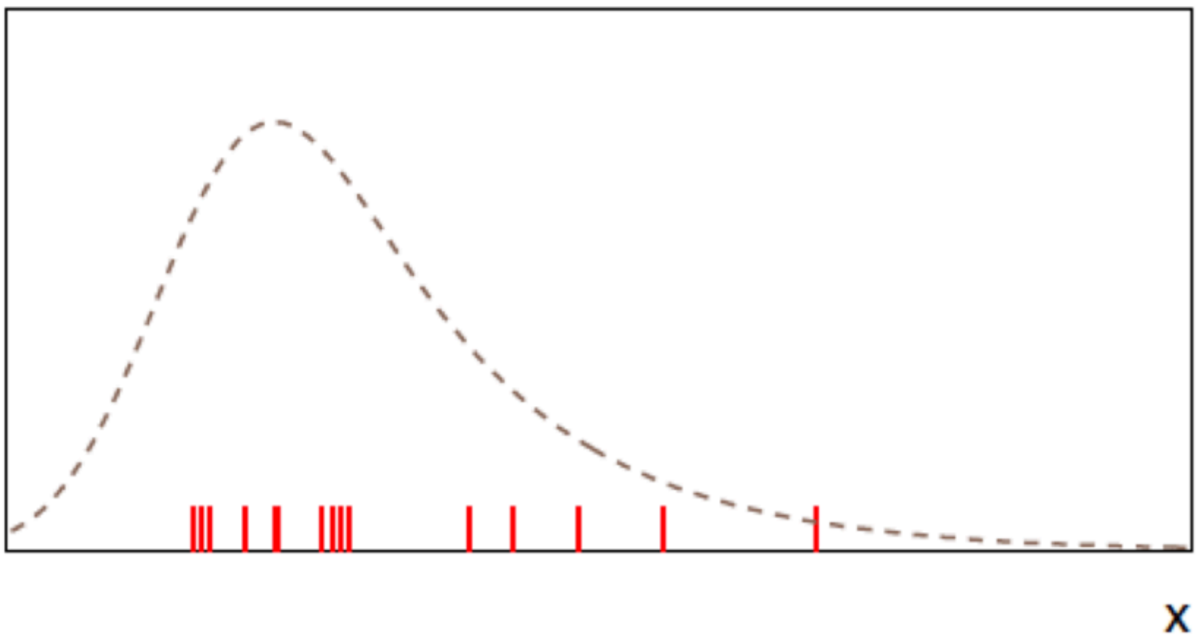
G Cowan

Use e.g. Gaussian kernel: $K(\vec{x}) = \frac{1}{(2\pi)^{d/2}} e^{-|\vec{x}|^2/2}$

This resmoothing effectively increases the power of an individual example event.

Density Estimation

Suppose the pdf (dashed line) below is not known in closed form, but we can generate events that follow it (the red tick marks):

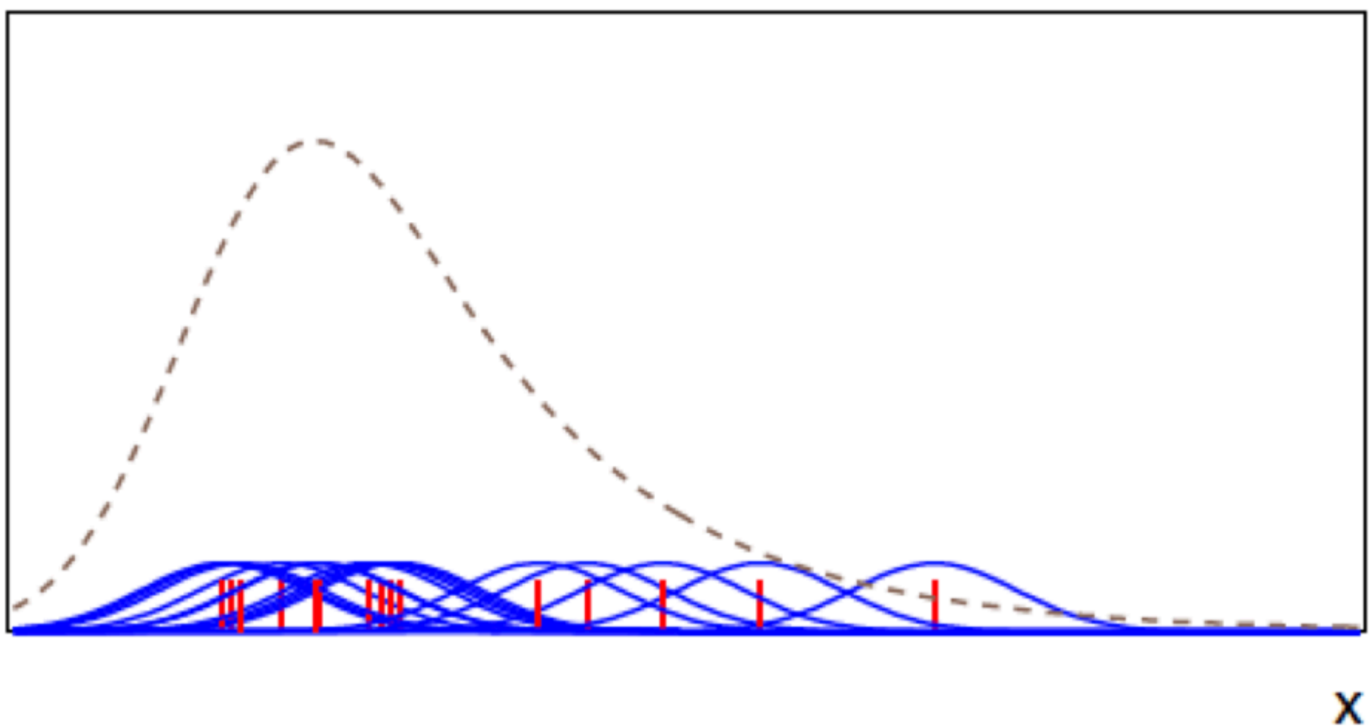


Goal is to find an approximation to the pdf using the generated date values.

G Cowan

Density Estimation

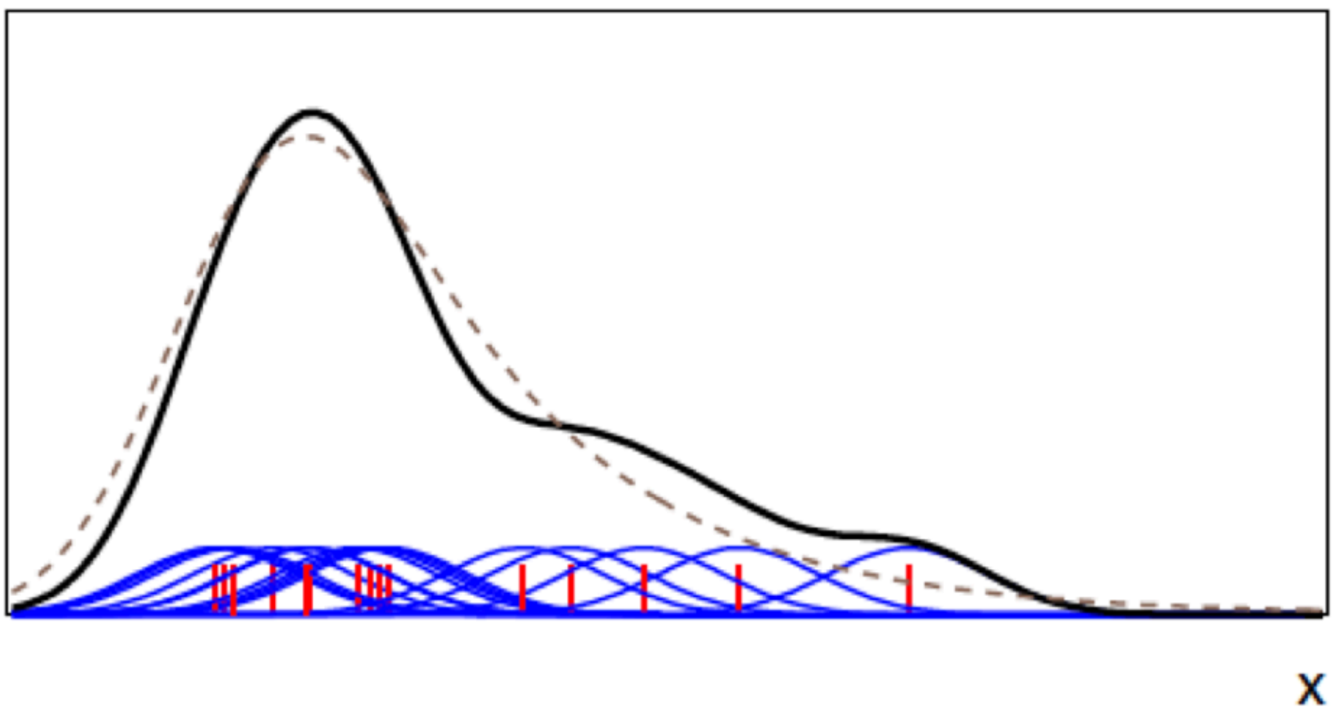
Place a kernel pdf (here a Gaussian) centred around each generated event weighted by $1/N_{\text{event}}$:



G Cowan

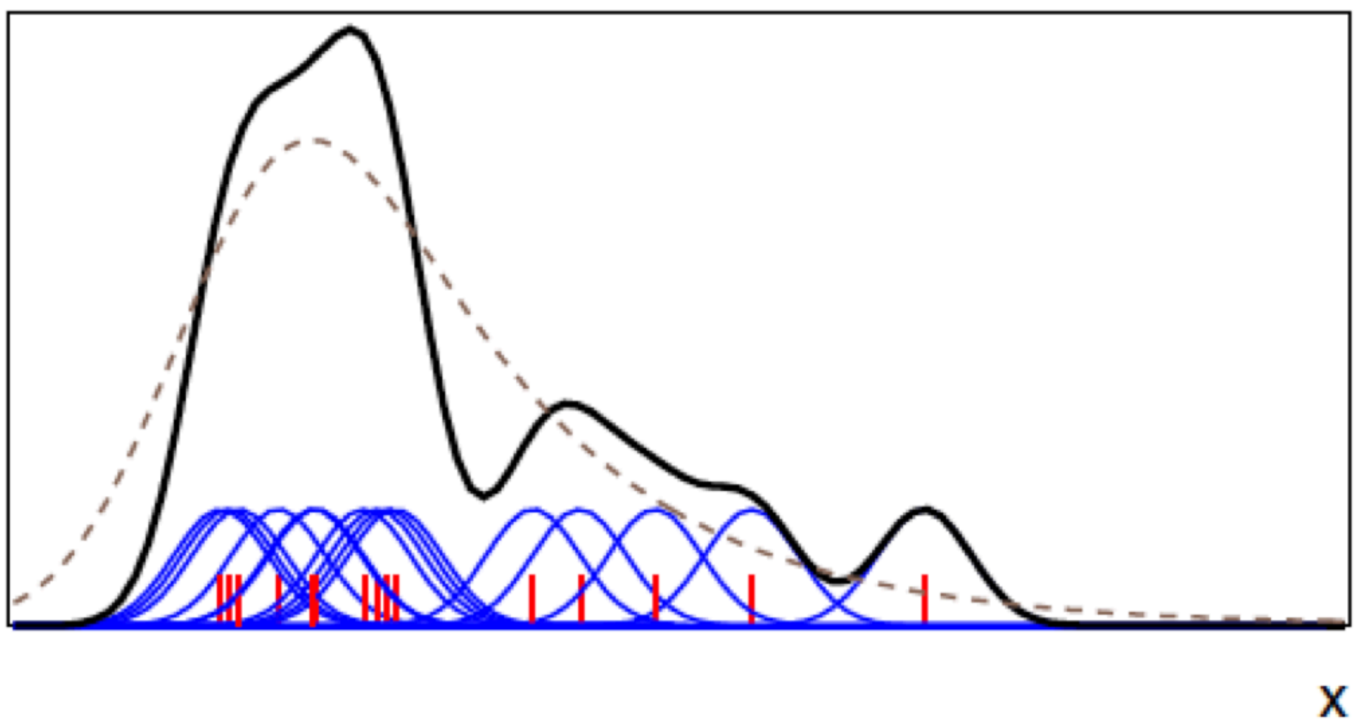
Density Estimation

The KDE estimate the pdf is given by the sum of all of the Gaussians:



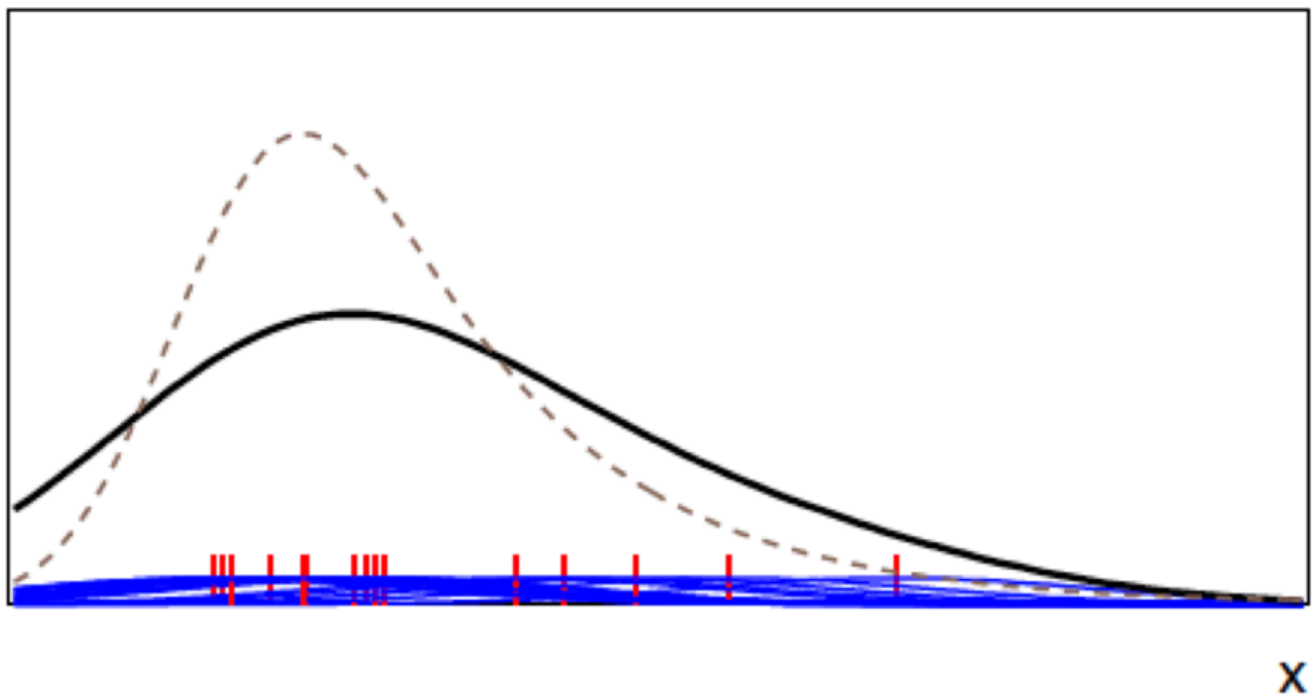
Density Estimation

The width h of the Gaussians is analogous to the bin width of a histogram. If it is too small, the estimator has noise:



Density Estimation

If width of Gaussian kernels too large, structure is washed out:



KDE

Discussion

KDE evaluation can be very slow

loop over all examples for every eval.

KDE training is trivial

zero time, simple construction on data

Adaptive strategies

Make wider kernels were low prob.

Still suffers from problems in very high dimensional applications.

Neural Networks

Neural networks

Strategy:

$$f(\bar{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^1$$

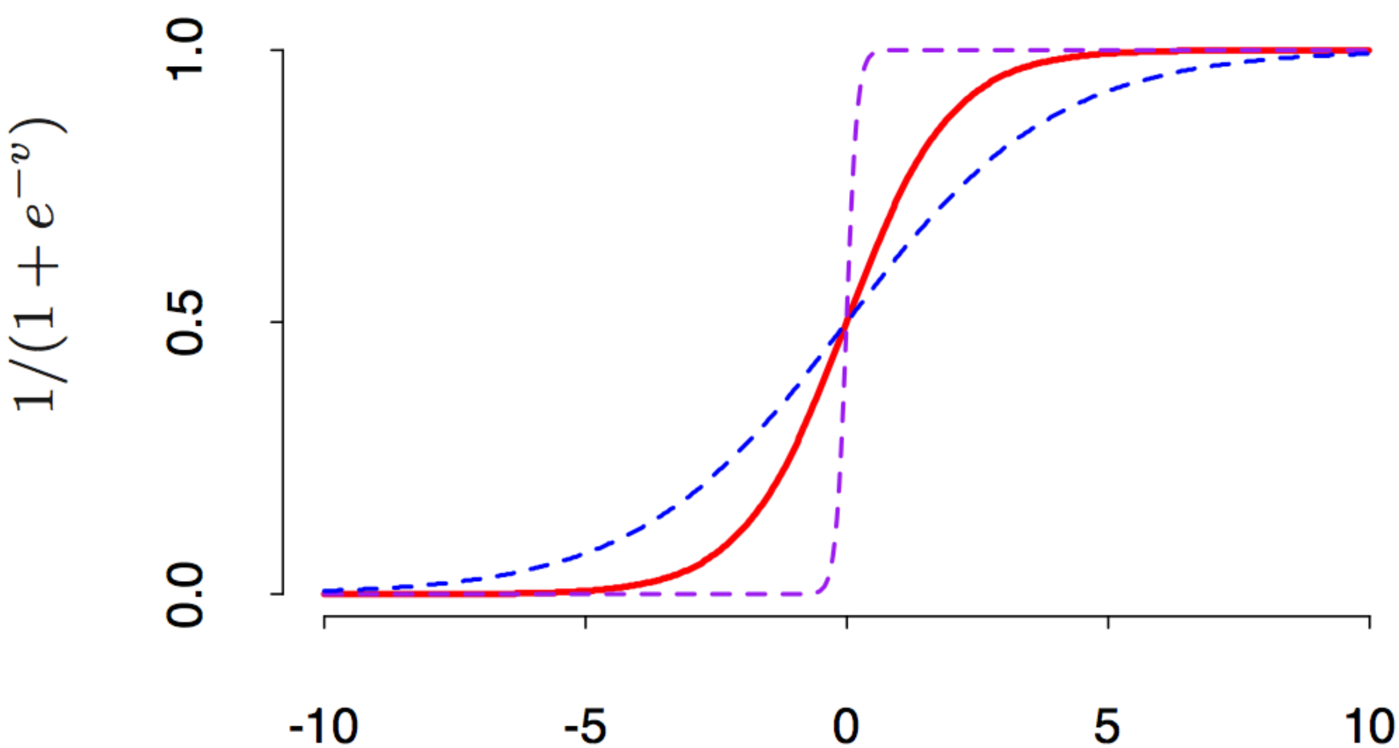
Build f(x)=y(x) out of a pile of convoluted mini-functions

$$y(\vec{x}) = h\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

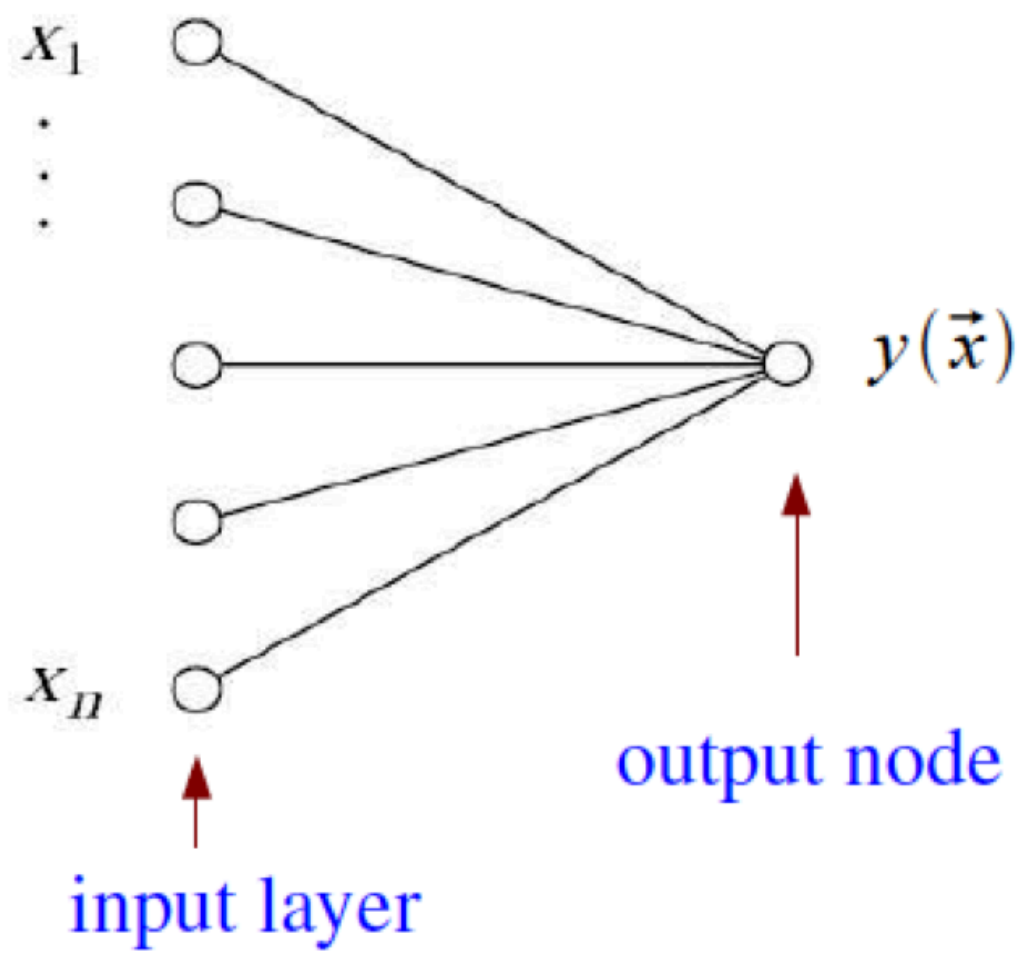
here h() is a non-linear *activation* function and the w factors are unknown parameters

Neuron

Example activation function



Simple visualization



What weights?

Every set of weight values defines a different function y(x). Which to use?

Define a good function as one which minimizes the error:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{a=1}^{N} |y(\vec{x}_a, \boldsymbol{w}) - t_a|^2 = \sum_{a=1}^{N} E_a(\boldsymbol{w})$$

Contribution to error from each event

Finding good weights

We have

a weight space $\qquad y(\vec{x}) = h\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$

a quality metric $\qquad E(\boldsymbol{w})$

We need

to find the max quality (or min error)

Search the space!

Searching for weights

Consider gradient descent method: from an initial guess in weight space $\boldsymbol{w}^{(1)}$ take a small step in the direction of maximum decrease. I.e. for the step τ to $\tau+1$,

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E(\boldsymbol{w}^{(\tau)})$$

learning rate ($\eta>0$)

If we do this with the full error function $E(\boldsymbol{w})$, gradient descent does surprisingly poorly;

But gradient descent turns out to be useful with an online (sequential) method, i.e., where we update $\boldsymbol{w}$ for each training event a, (cycle through all training events):

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_a(\boldsymbol{w}^{(\tau)})$$

Back prop

Error backpropagation ("backprop") is an algorithm for finding the derivatives required for gradient descent minimization.

The network output can be written $y(\boldsymbol{x}) = h(u(\boldsymbol{x}))$ where

$$u(\vec{x}) = \sum_{j=0} w_{1j}^{(2)} \varphi_j(\vec{x}), \qquad \varphi_j(\vec{x}) = h\left(\sum_{k=0} w_{jk}^{(1)} x_k\right)$$

where we defined $\phi_0 = x_0 = 1$ and wrote the sums over the nodes in the preceding layers starting from 0 to include the offsets.

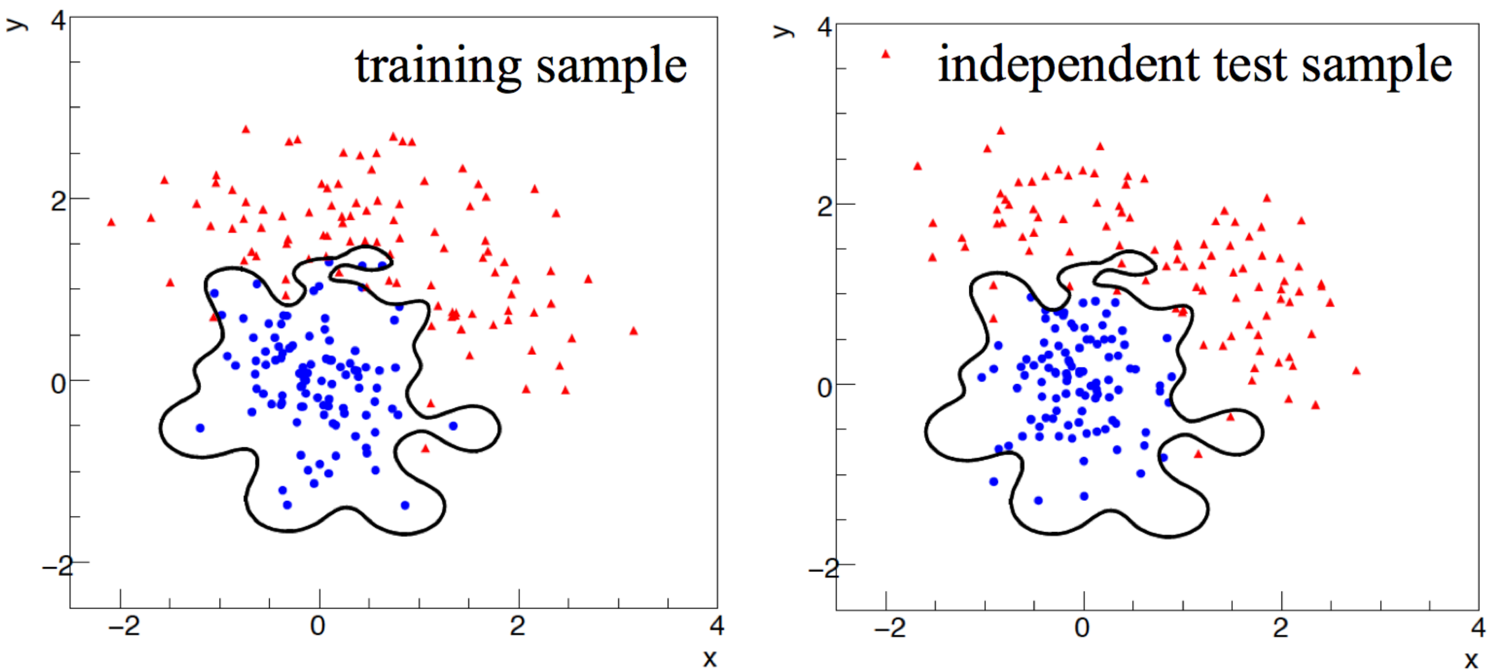
So e.g. for event a we have $$\frac{\partial E_a}{\partial w_{1j}^{(2)}} = (y_a - t_a)\, h'(u(\vec{x}))\, \varphi_j(\vec{x})$$

derivative of activation function

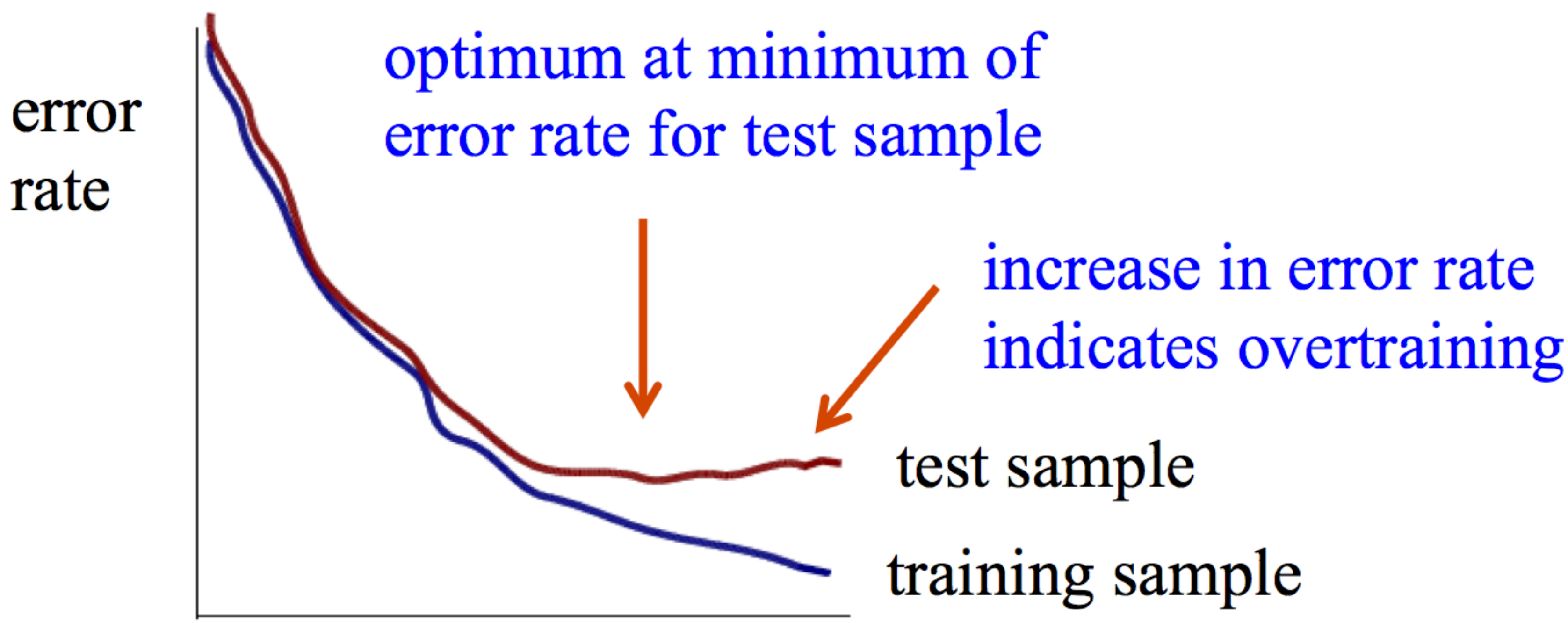
Chain rule gives all the needed derivatives.

How much to train?

A complex network, heavily trained will learn the statistical fluctuations of the training examples.



Avoiding overtraining

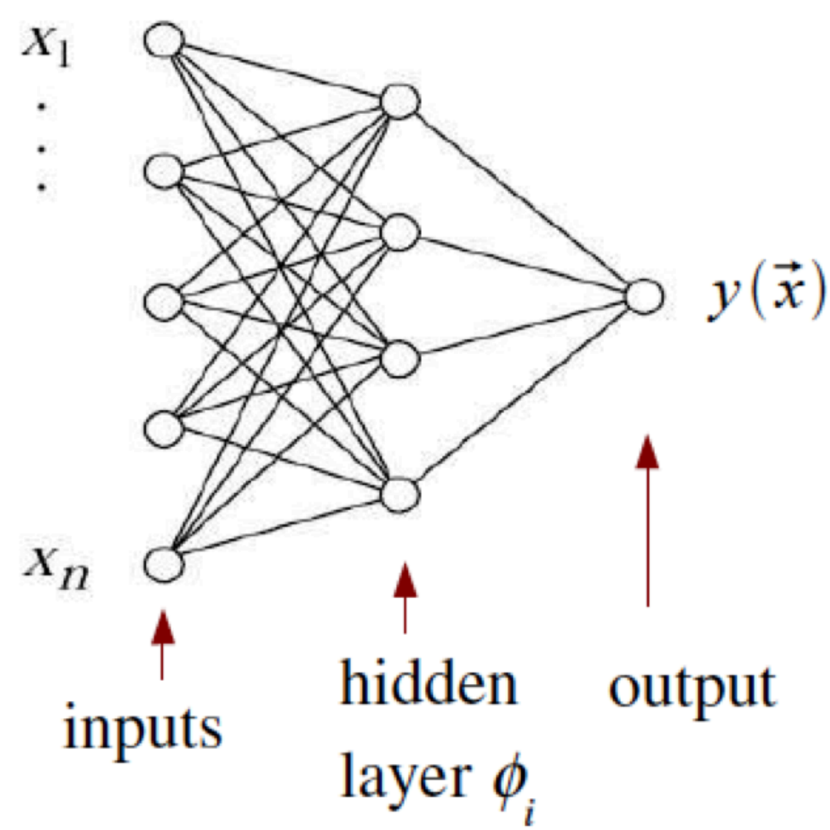


More complex networks

Superscript for weights indicates layer number

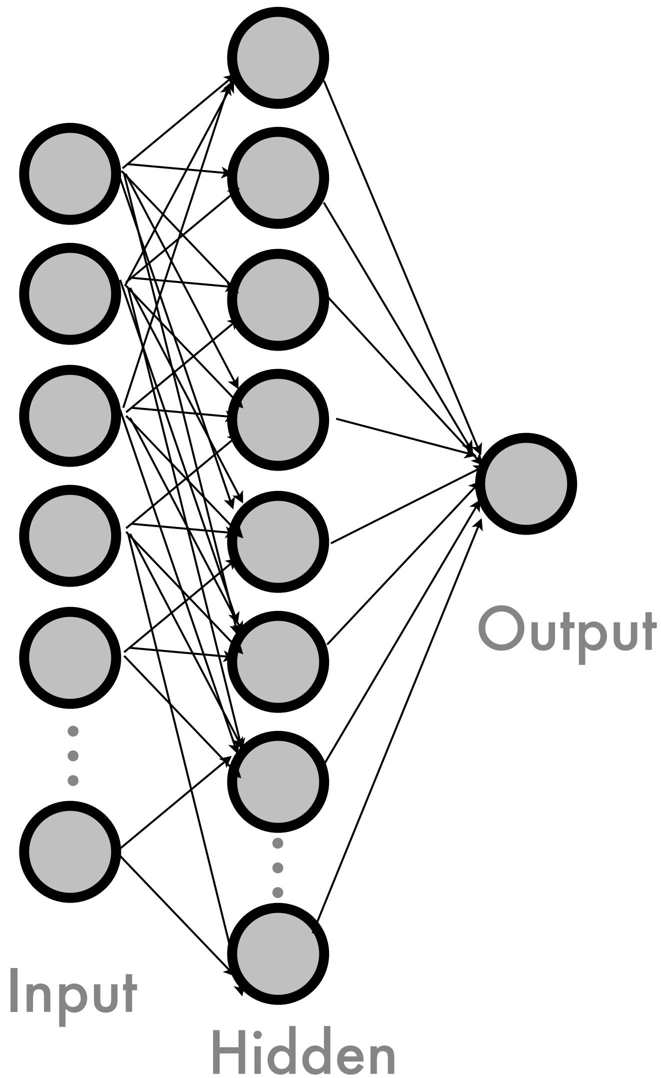
$$\varphi_i(\vec{x}) = h\left(w_{i0}^{(1)} + \sum_{j=1}^{n} w_{ij}^{(1)} x_j \right)$$

$$y(\vec{x}) = h\left(w_{10}^{(2)} + \sum_{j=1}^{n} w_{1j}^{(2)} \varphi_j(\vec{x}) \right)$$



How complex?

Essentially a functional fit with many parameters



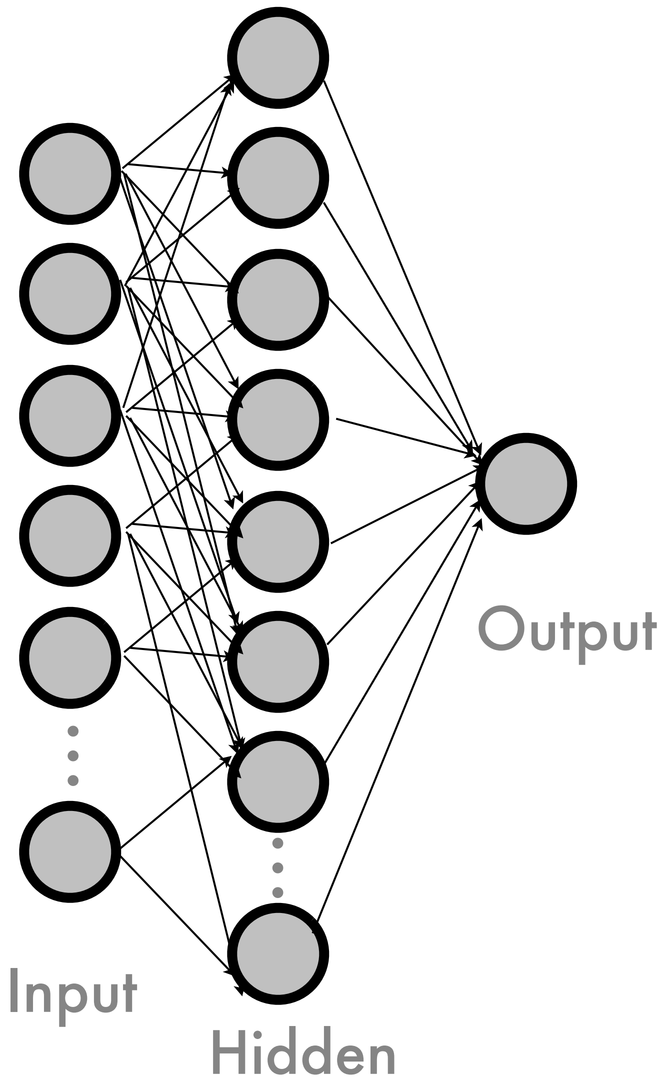
<u>Single layer</u>

In theory any function can be learned with a single hidden layer.

But might require very large hidden layer

Neural Networks

Essentially a functional fit with many parameters



Problem:
Networks with > 1 layer are
very difficult to train.

Consequence:
Networks are not good
at learning non-linear functions.
(like invariant masses!)

In short:
Can't just throw 4-vectors at NN.

Search for Input

ATLAS-CONF-2013-108

Can't just use 4v

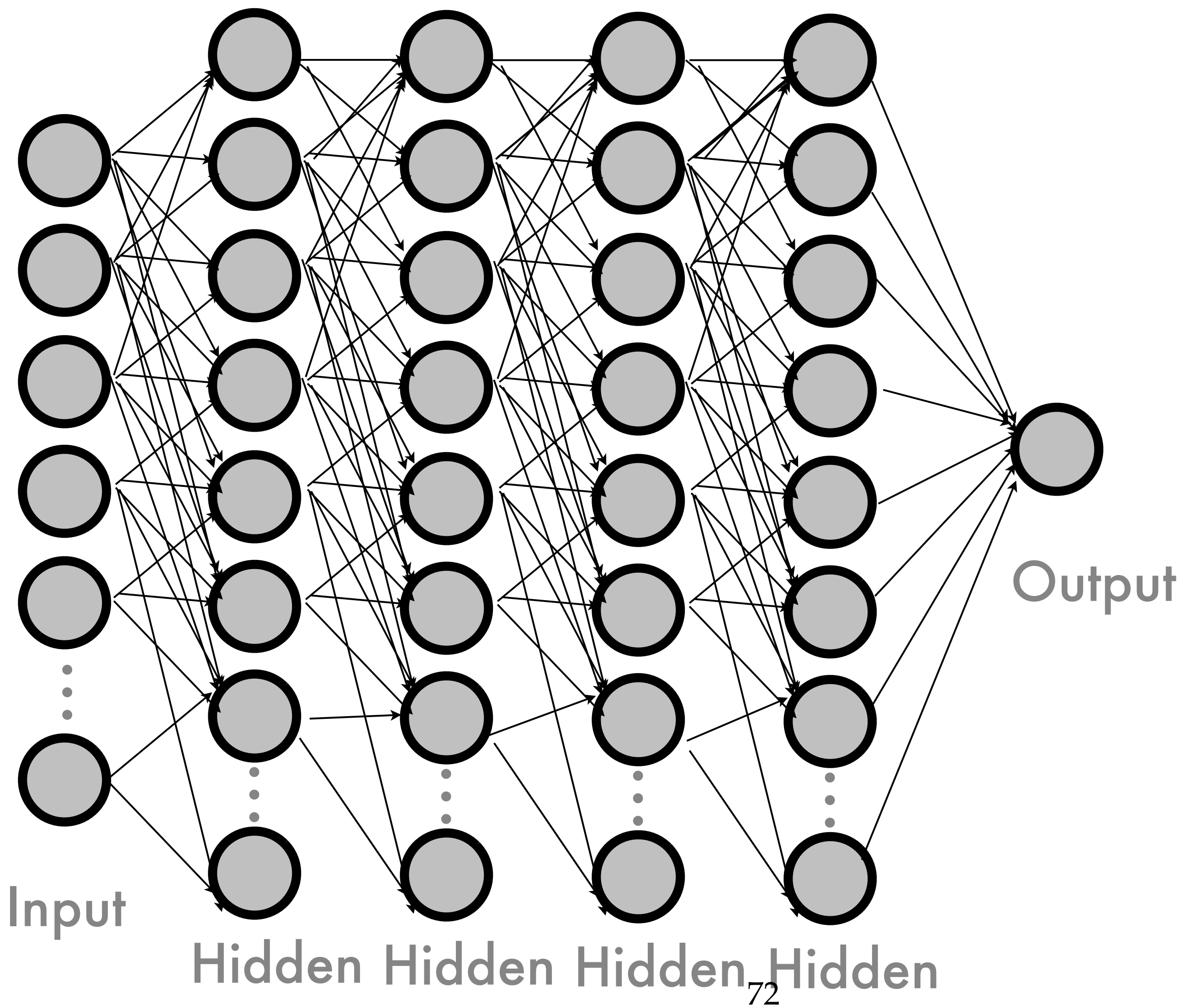
Can't give it too many inputs

Painstaking search through input feature space.

Variable	VBF			Boosted		
	$\tau_{lep}\tau_{lep}$	$\tau_{lep}\tau_{had}$	$\tau_{had}\tau_{had}$	$\tau_{lep}\tau_{lep}$	$\tau_{lep}\tau_{had}$	$\tau_{had}\tau_{had}$
$m_{\tau\tau}^{\mathrm{MMC}}$	•	•	•	•	•	•
$\Delta R(\tau, \tau)$	•	•	•		•	•
$\Delta\eta(j_1, j_2)$	•	•	•			
m_{j_1,j_2}	•	•	•			
$\eta_{j_1} \times \eta_{j_2}$		•	•			
$p_{\mathrm{T}}^{\mathrm{Total}}$		•	•			
sum p_{T}					•	•
$p_{\mathrm{T}}(\tau_1)/p_{\mathrm{T}}(\tau_2)$					•	•
$E_{\mathrm{T}}^{\mathrm{miss}}\,\phi$ centrality		•	•	•	•	•
$x_{\tau 1}$ and $x_{\tau 2}$						•
$m_{\tau\tau,j_1}$				•		
m_{ℓ_1,ℓ_2}				•		
$\Delta\phi_{\ell_1,\ell_2}$				•		
sphericity				•		
$p_{\mathrm{T}}^{\ell_1}$				•		
$p_{\mathrm{T}}^{j_1}$				•		
$E_{\mathrm{T}}^{\mathrm{miss}}/p_{\mathrm{T}}^{\ell_2}$				•		
m_{T}		•			•	
$\min(\Delta\eta_{\ell_1\ell_2,\mathrm{jets}})$	•					
$j_3\ \eta$ centrality	•					
$\ell_1 \times \ell_2\ \eta$ centrality	•					
$\ell\ \eta$ centrality		•				
$\tau_{1,2}\ \eta$ centrality			•			

Table 3: Discriminating variables used for each channel and category. The filled circles identify which variables are used in each decay mode. Note that variables such as $\Delta R(\tau, \tau)$ are defined either between the two leptons, between the lepton and τ_{had}, or between the two τ_{had} candidates, depending on the decay mode.

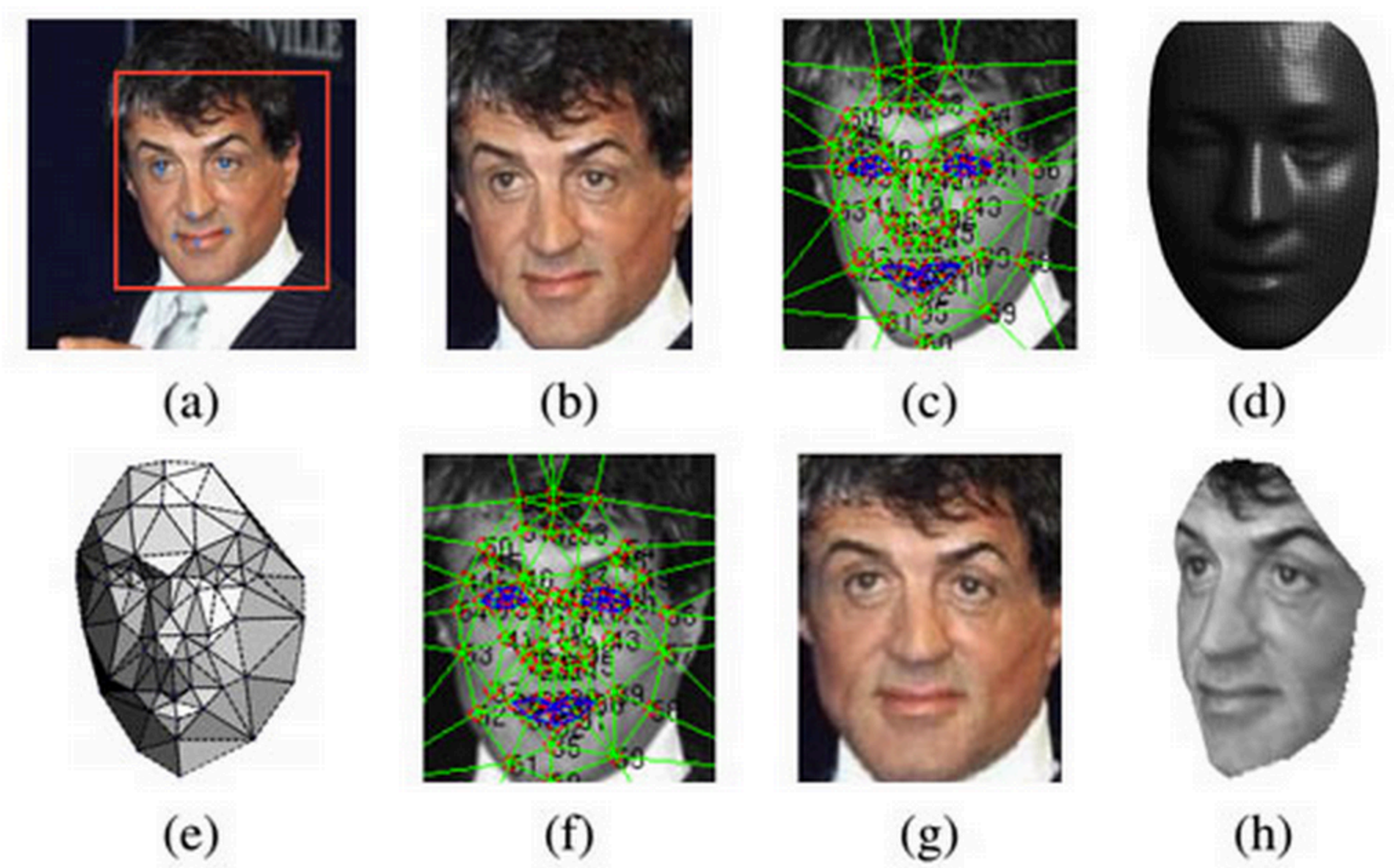
Deep networks



New tools
let us
train
deep
networks.

How well
do they work?

Real world applications



Head turn: DeepFace uses a 3-D model to rotate faces, virtually, so that they face the camera. Image (a) shows the original image, and (g) shows the final, corrected version.

Paper

nature COMMUNICATIONS

ARTICLE

Received 19 Feb 2014 | Accepted 4 Jun 2014 | Published 2 Jul 2014

DOI: 10.1038/ncomms5308

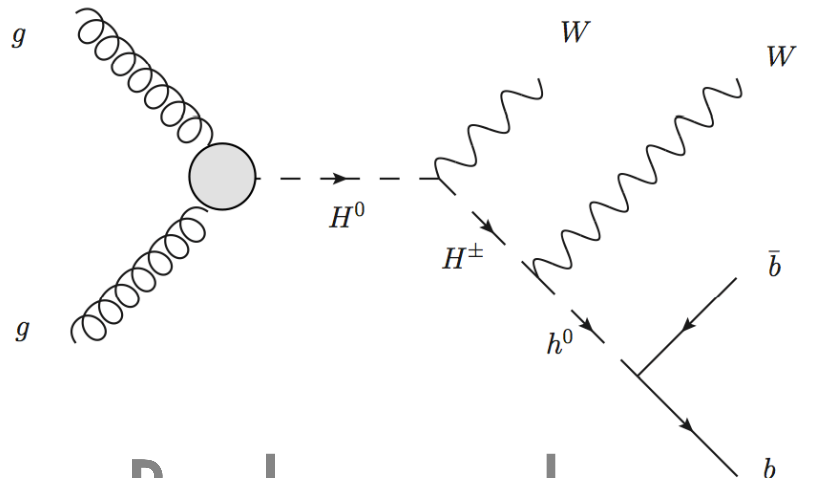
Searching for exotic particles in high-energy physics with deep learning

P. Baldi[1], P. Sadowski[1] & D. Whiteson[2]

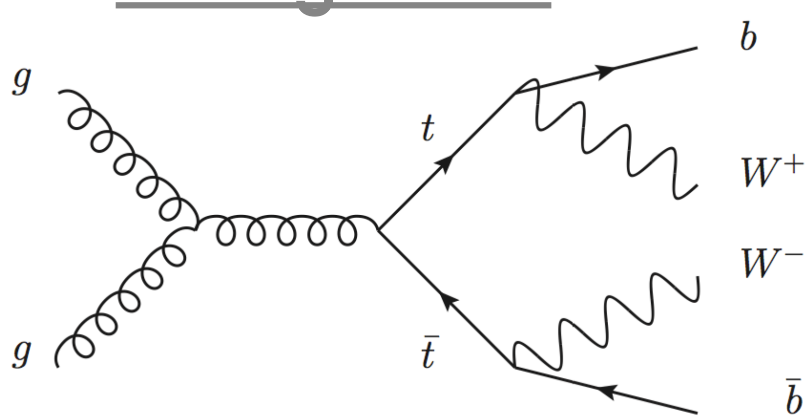
arXiv: 1402.4735

Benchmark problem

Signal



Background



Can deep networks automatically discover useful variables?

4-vector inputs

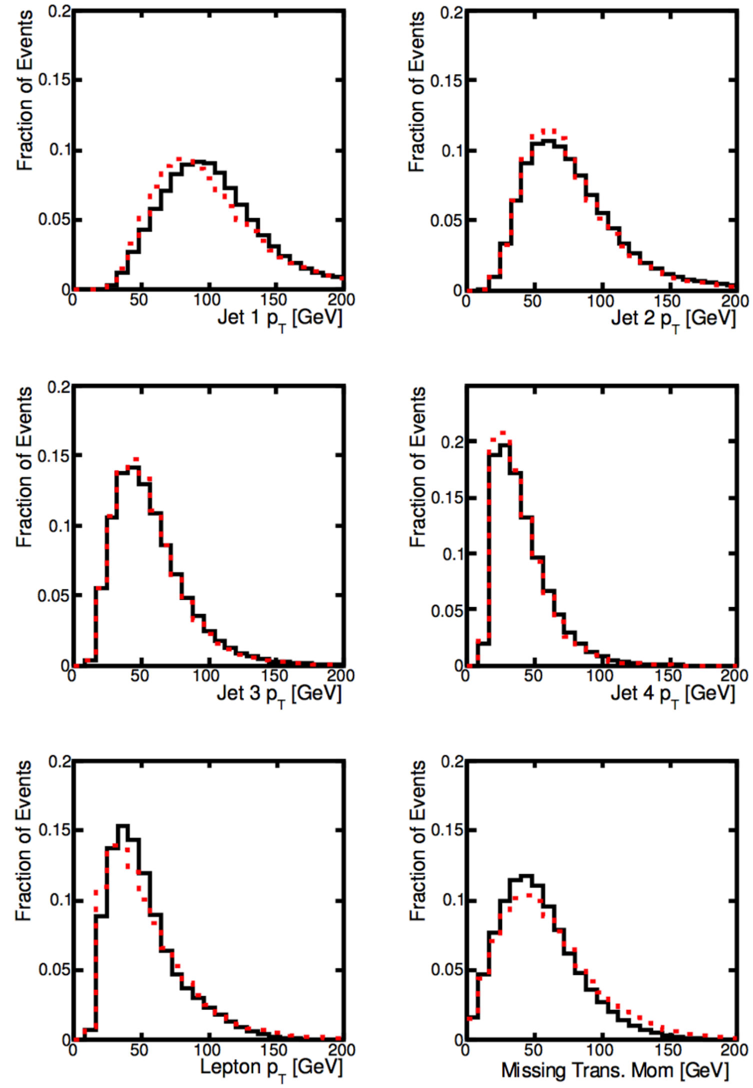
<u>21 Low-level vars</u>

jet+lepton mom. (3x5)

missing ET (2)

jet btags (4)

Not much separation visible in 1D projections



4-vector inputs

<u>7 High-level vars</u>

m(WWbb)

m(Wbb)

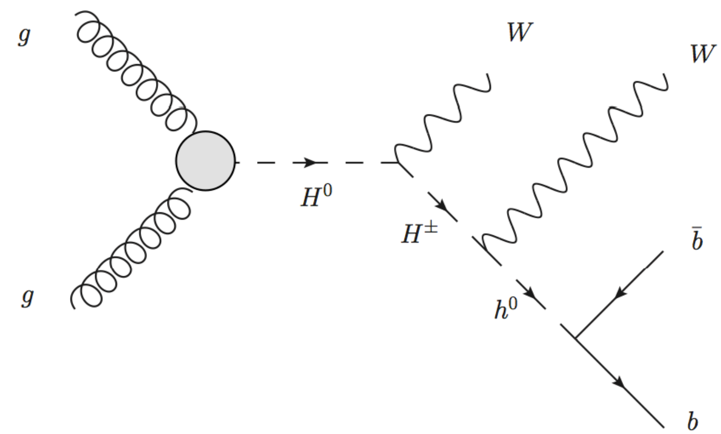
m(bb)

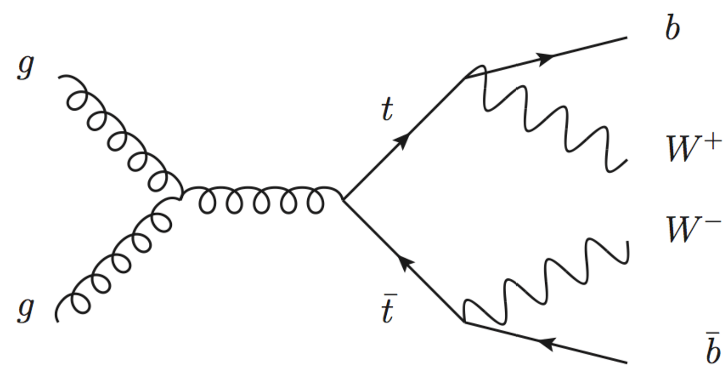
m(bjj)

m(jj)

m(lv)

m(blv)





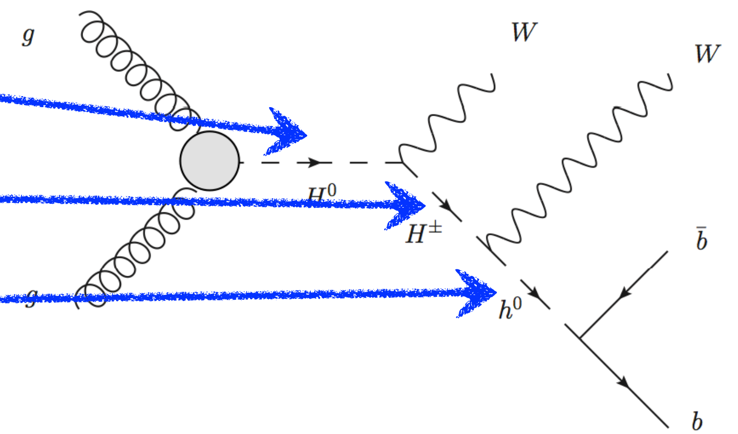
4-vector inputs

<u>7 High-level vars</u>

m(WWbb)

m(Wbb)

m(bb)

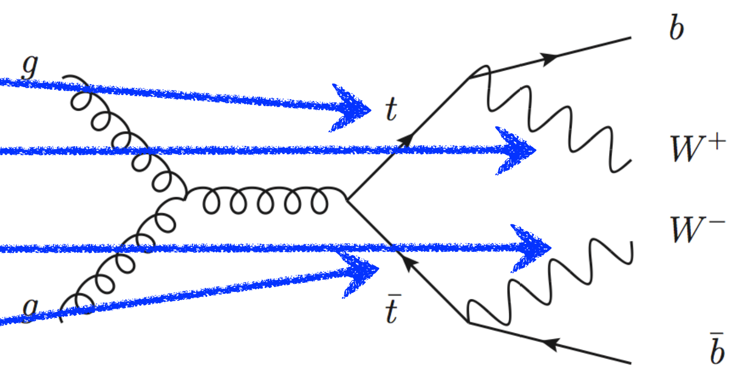


m(bjj)

m(jj)

m(lv)

m(blv)



4-vector inputs

<u>**7 High-level vars**</u>

m(WWbb)

m(Wbb)

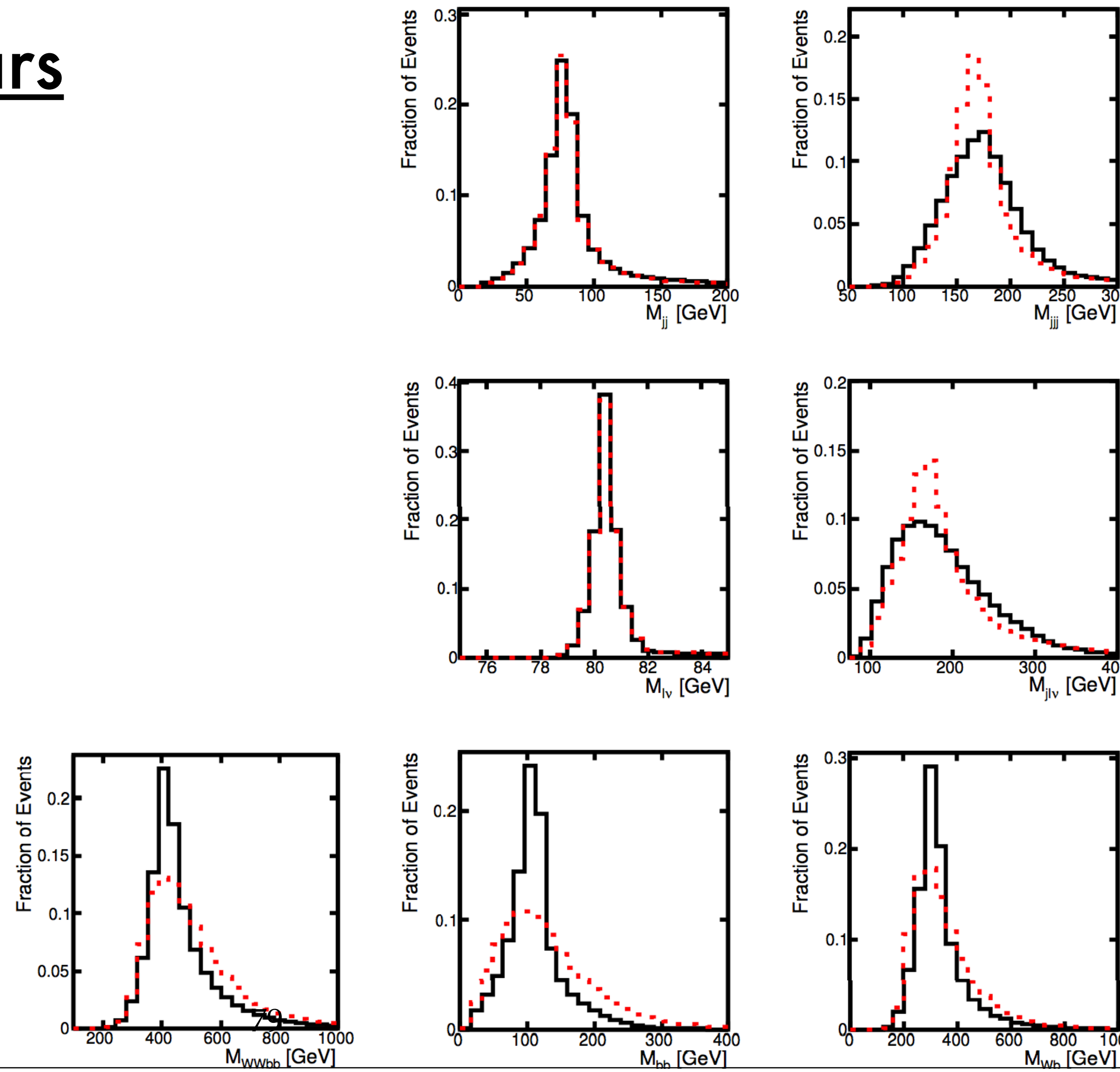
m(bb)

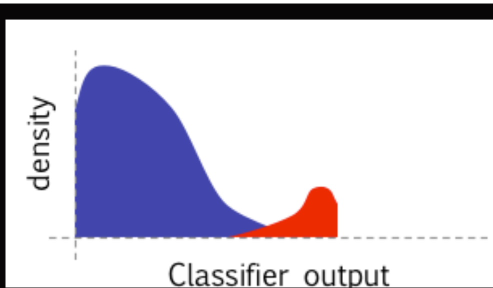
m(bjj)

m(jj)

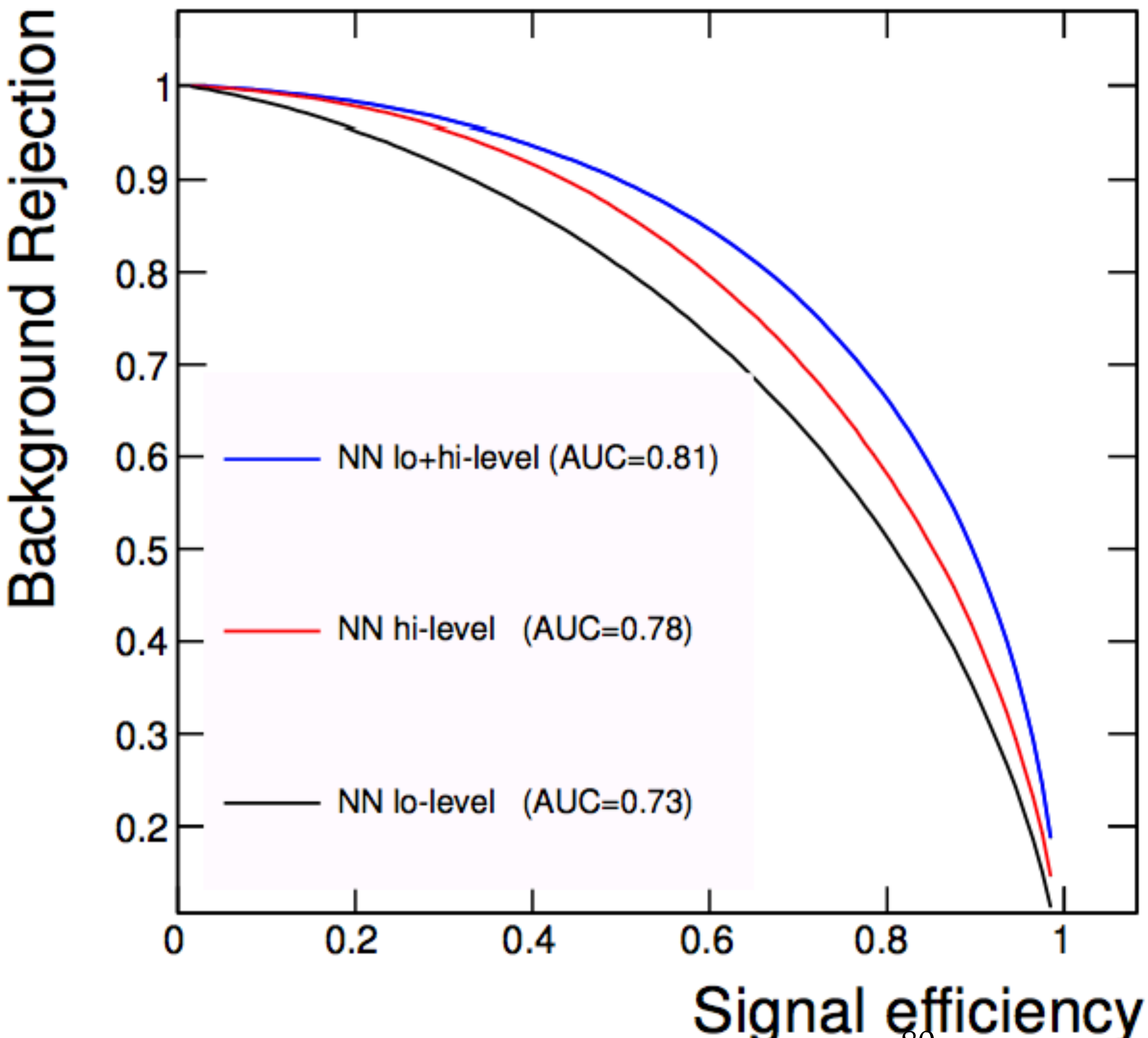
m(lv)

m(blv)





Standard NNs



Results

Adding hi-level boosts performance

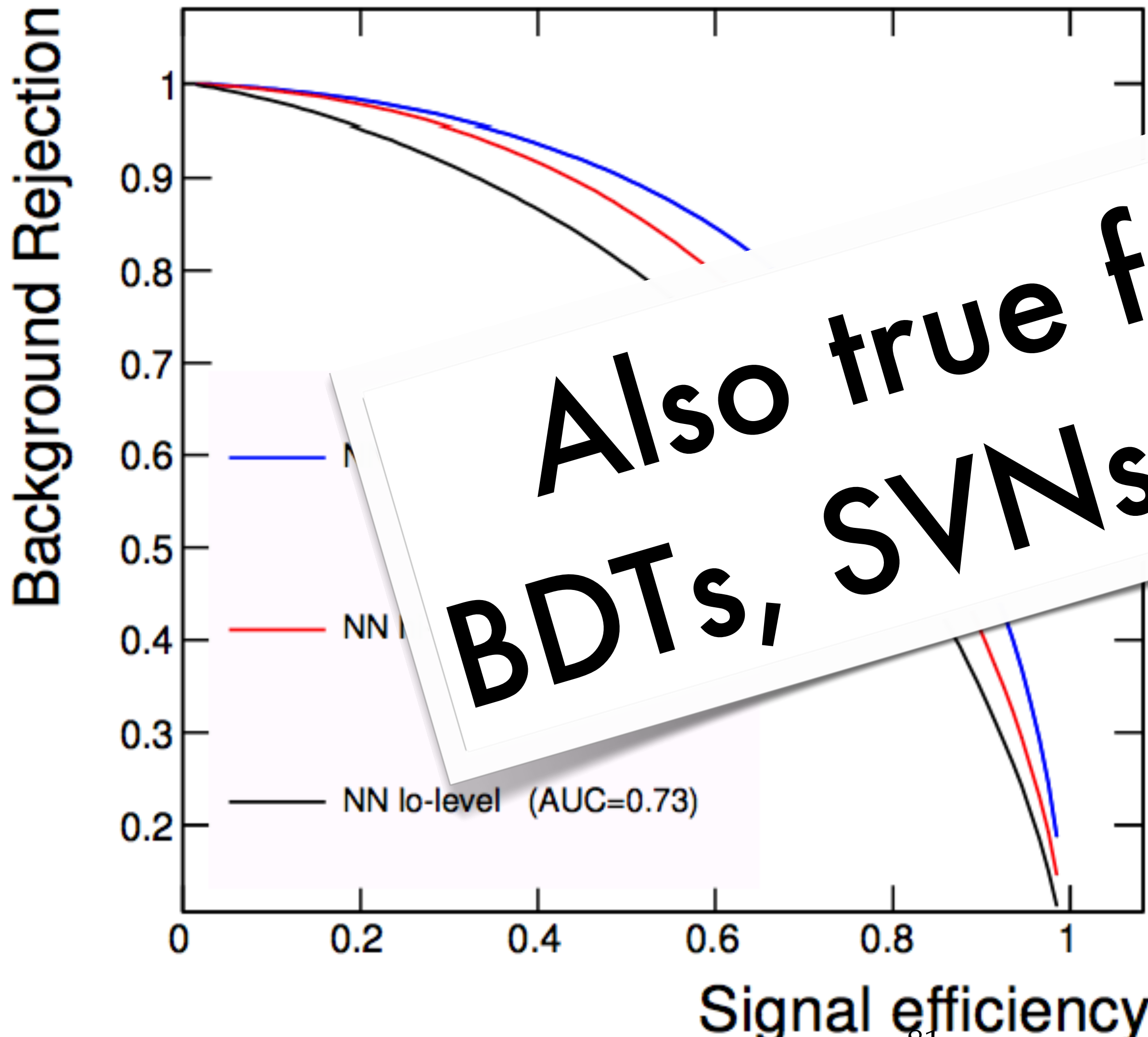
Better: lo+hi-level.

Conclude:

NN can't find hi-level vars.

Hi-level vars do not have all info

Standard NNs



Also true for BDTs, SVNs, etc

<u>Results</u>

g hi-level

s performance

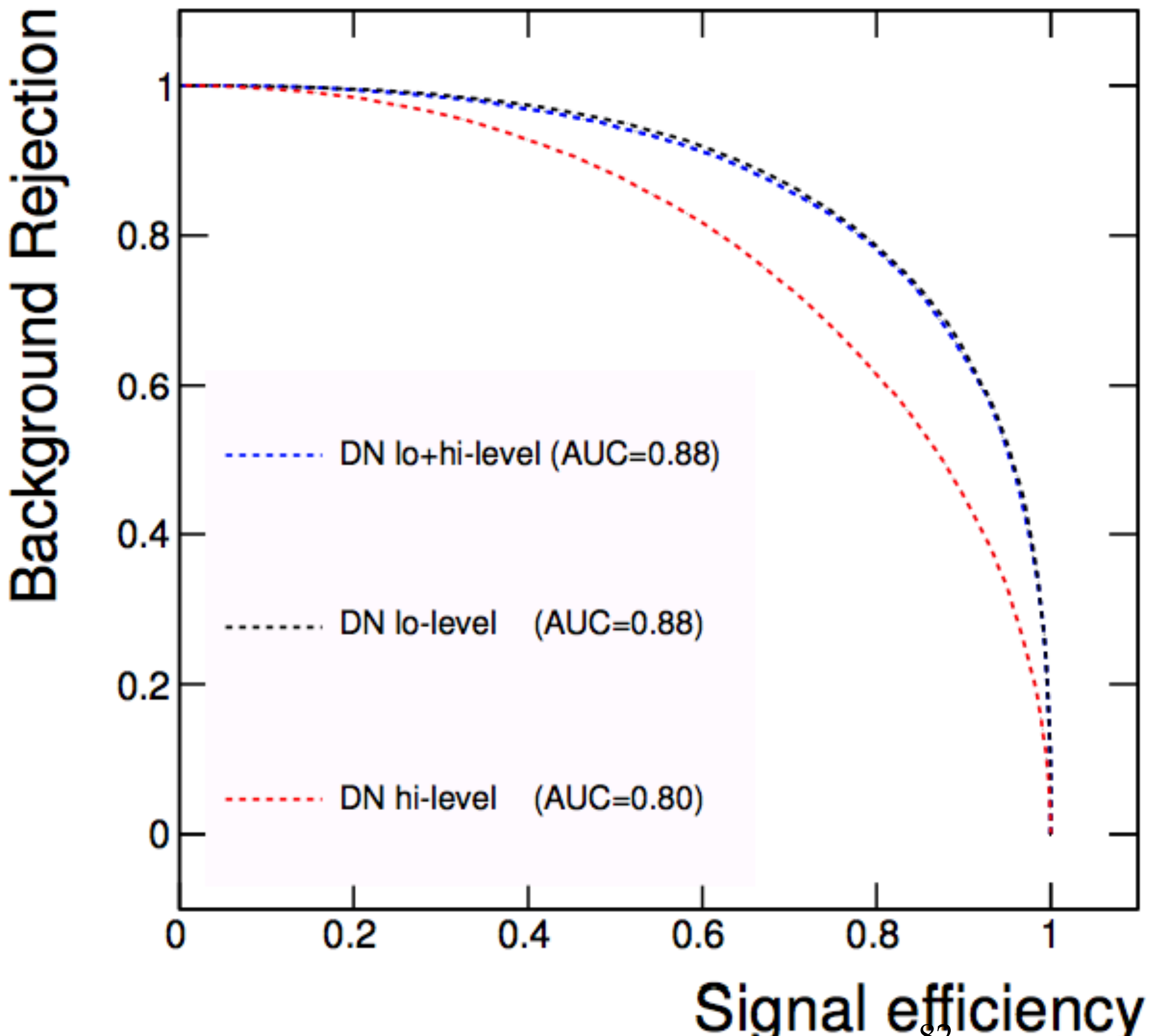
lo+hi-level.

<u>nclude:</u>

NN can't find
hi-level vars.

Hi-level vars
do not have all info

Deep Networks



Results

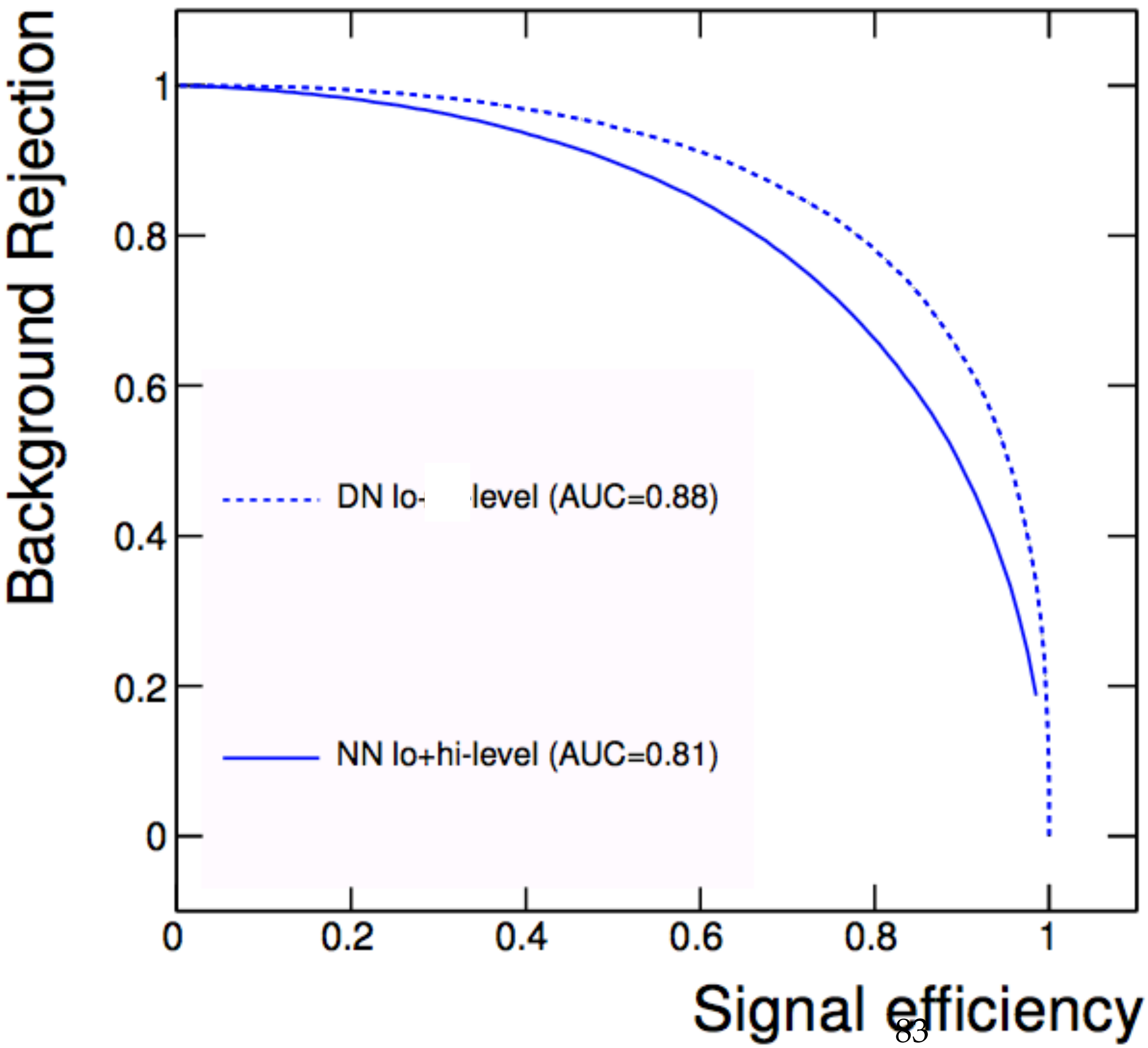
Lo+hi = lo.

Conclude:

DN can find hi-level vars.

Hi-level vars do not have all info are unnecessary

Deep Networks



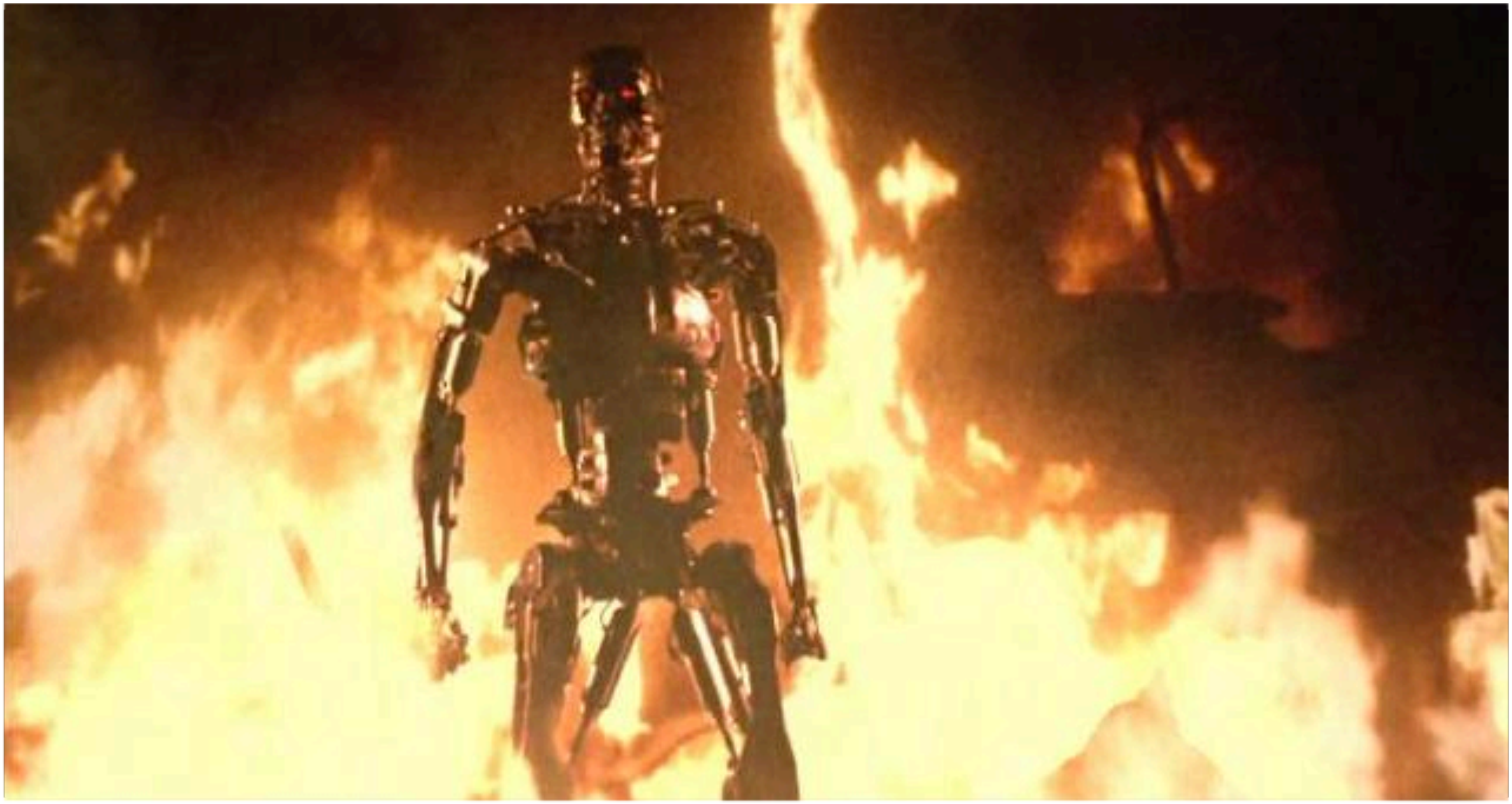
Results

DN > NN

Conclude:

DN does better than human assisted NN

The AIs win



Results

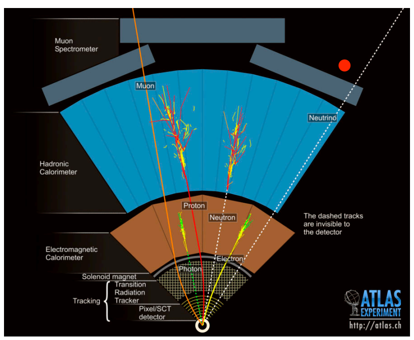
Identified example benchmark where traditional NNs fail to discover all discrimination power.

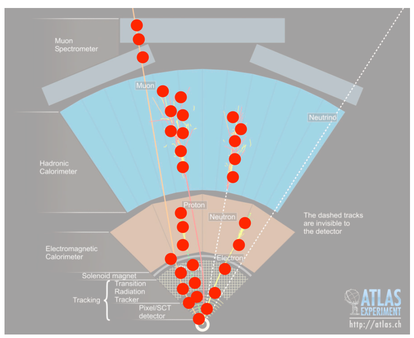
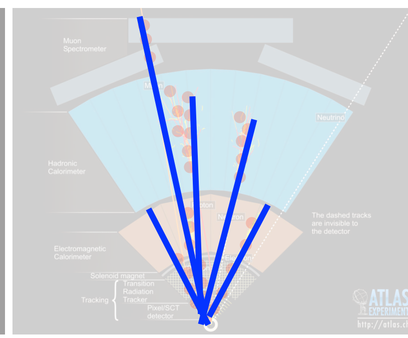
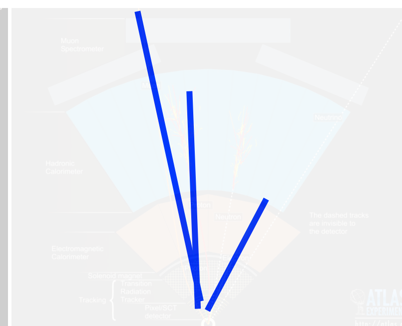
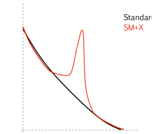
Adding human insight helps traditional NNs.

Deep networks succeed **without human insight.** **Outperform** human-boosted traditional NNs.

What is possible?

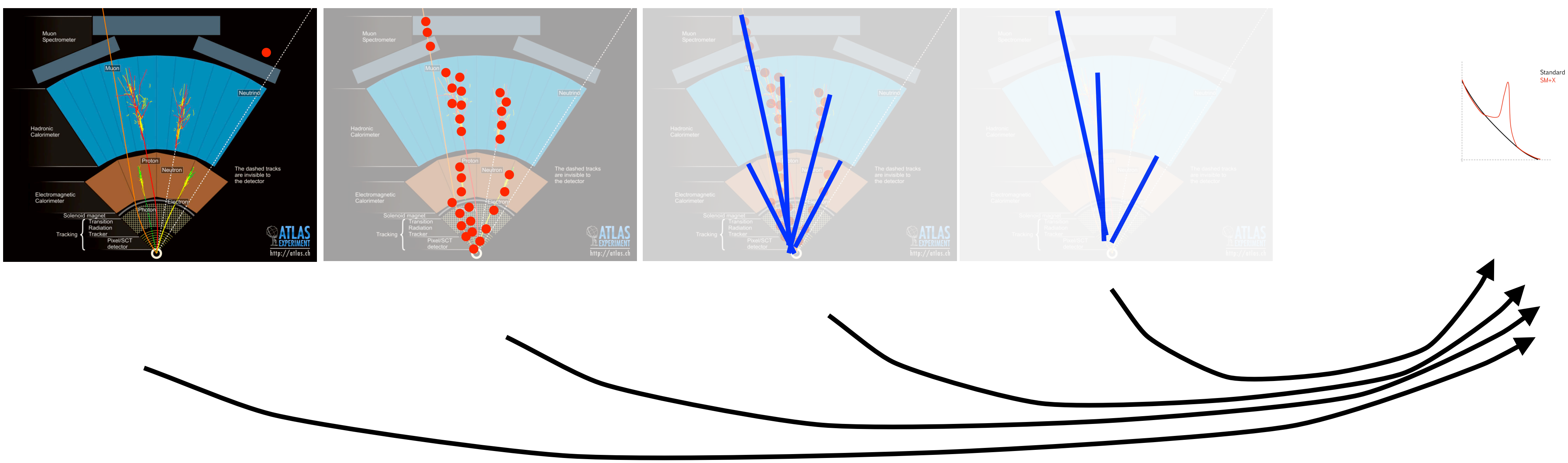
Raw	Sparsified	Reco	Select	Ana
1e7	1e3	100	50	1



What is possible?

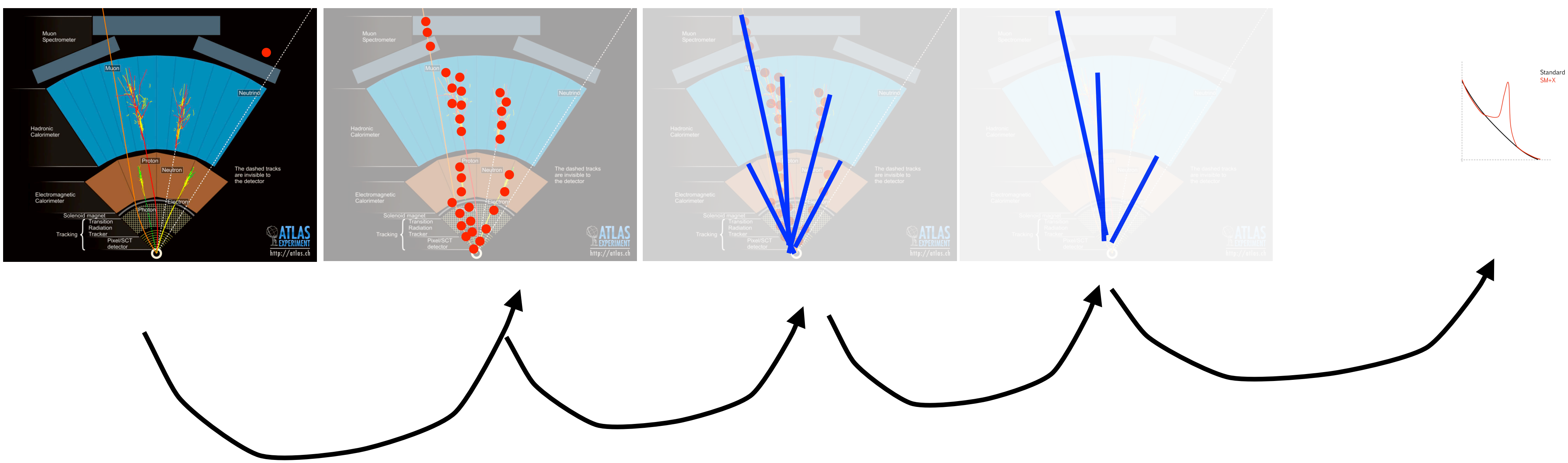
Raw	Sparsified	Reco	Select	Ana
1e7	1e3	100	50	1



Skip more steps with ML?

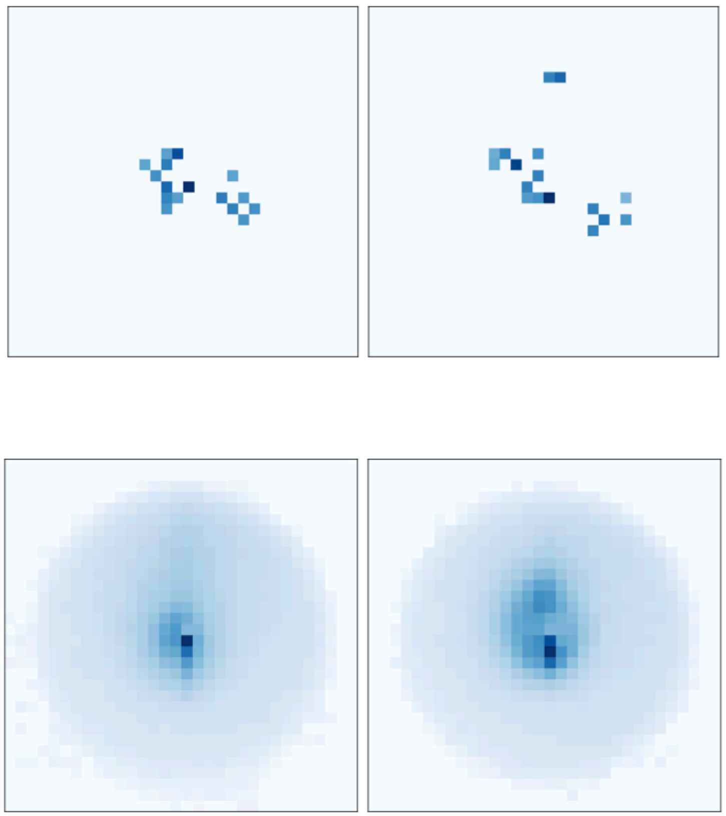
Or this?

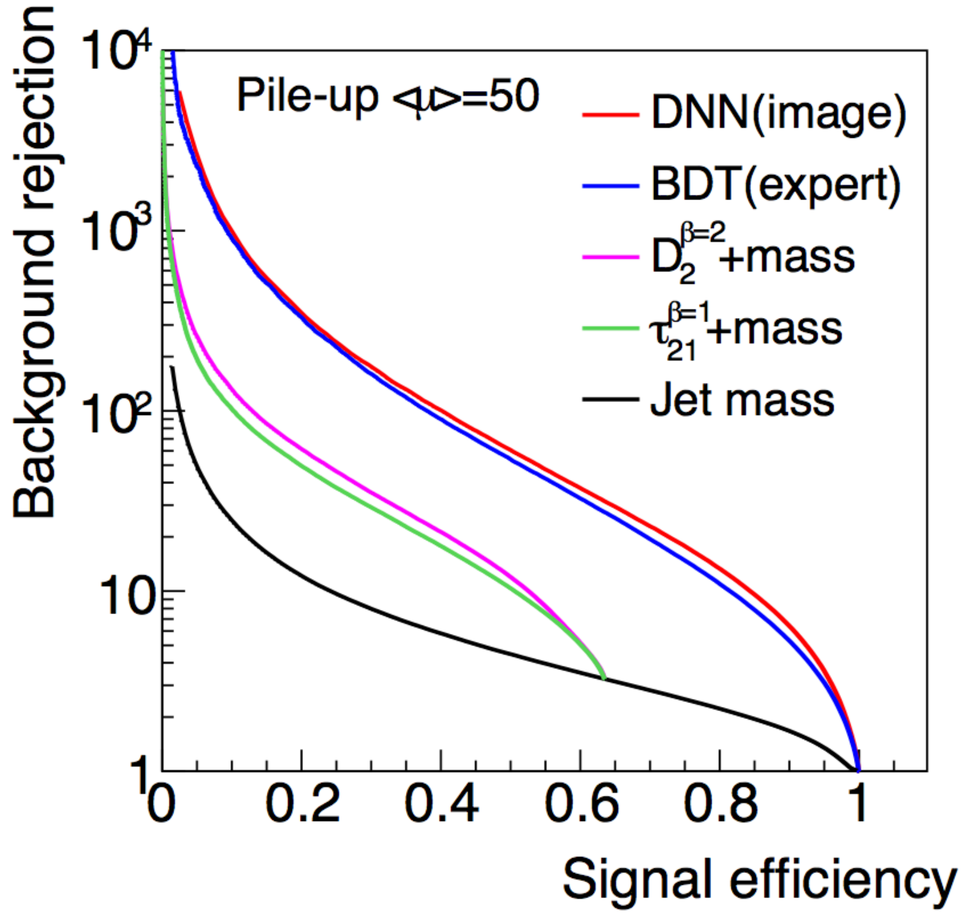
Raw	Sparsified	Reco	Select	Ana
1e7	1e3	100	50	1



Improve each step with ML?

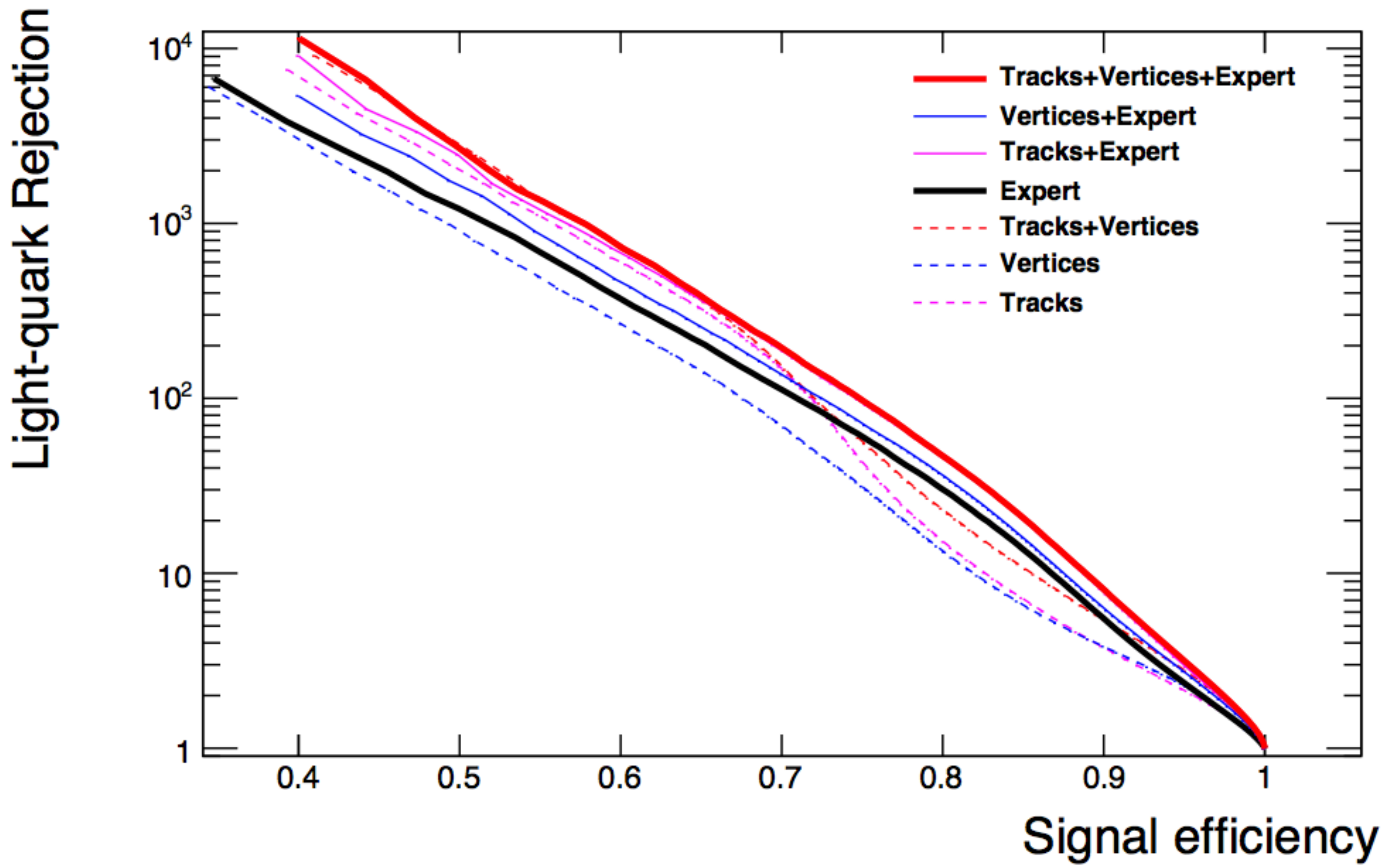
Jet tagging





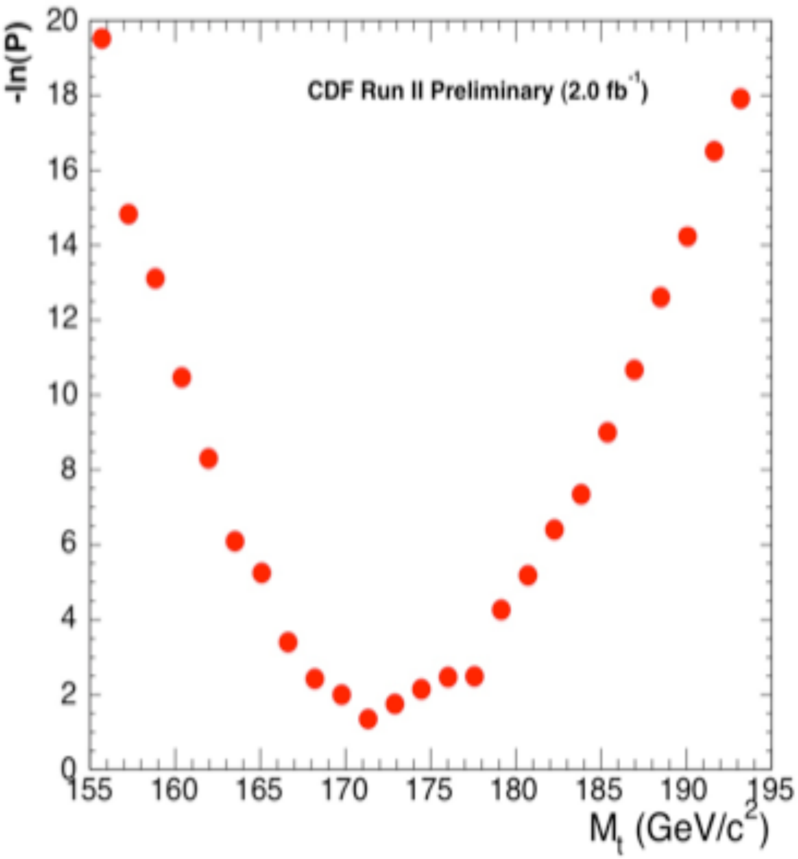
Pile-up <μ>=50
DNN(image)
BDT(expert)
$D_2^{\beta=2}$+mass
$\tau_{21}^{\beta=1}$+mass
Jet mass
Background rejection
Signal efficiency
10^4
10^3
10^2
10
1
0
0.2
0.4
0.6
0.8
1

b-tagging



Light-quark Rejection
10^4
10^3
10^2
10
1
0.4
0.5
0.6
0.7
0.8
0.9
1
Signal efficiency
Tracks+Vertices+Expert
Vertices+Expert
Tracks+Expert
Expert
Tracks+Vertices
Vertices
Tracks

Optimization



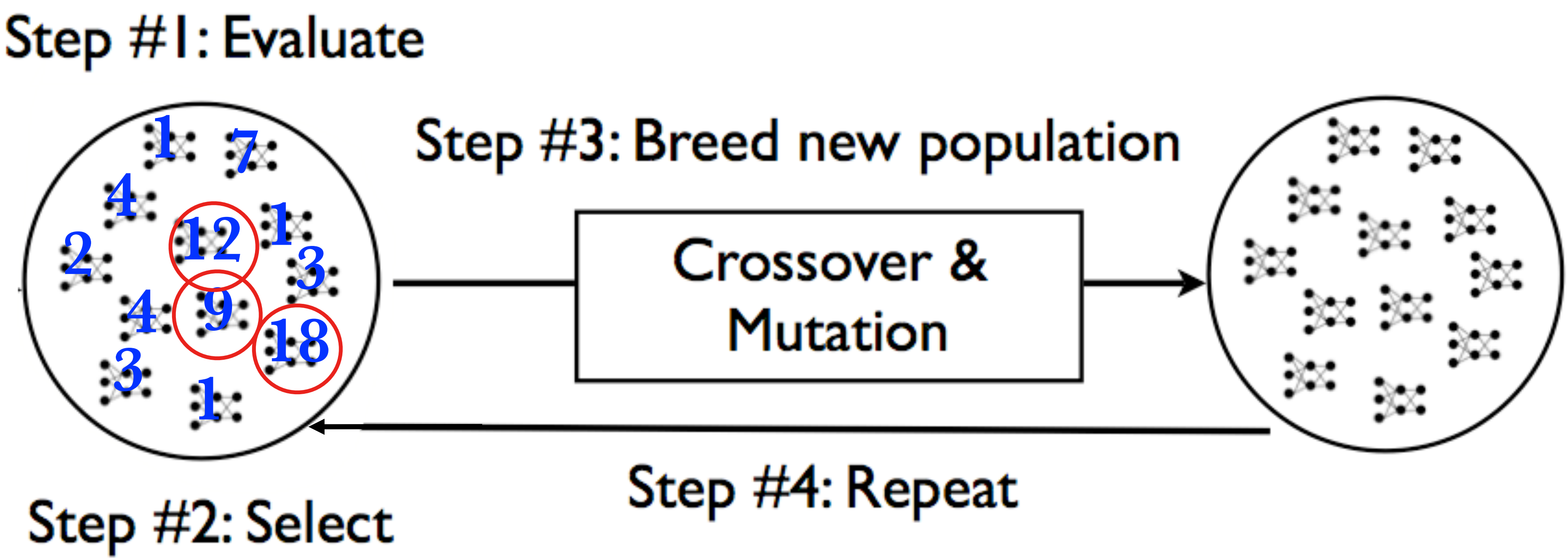
How to select events which give a top mass measurement with the smallest uncertainty?

- Uncertainty is a property of the set of events, not an individual event. No truth labels for each event.

- Various background affect measurement differently.

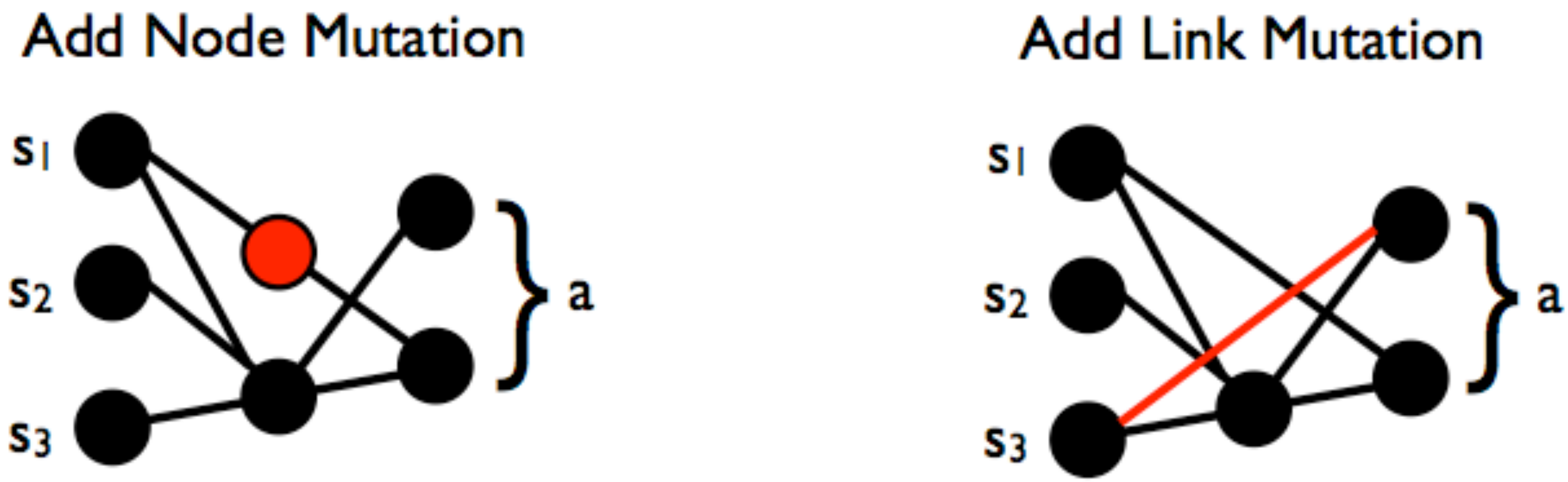
- Classifiers are not well suited. Optimize directly!

Optimization



NEAT

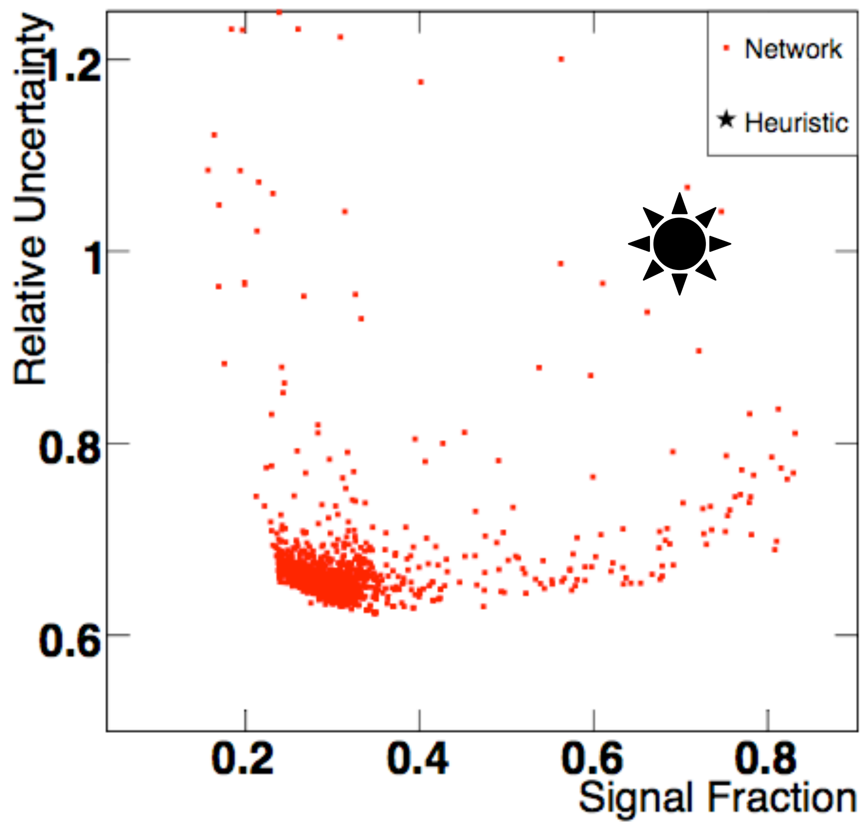
Using NEAT, we can search the space of topologies at the same time!



NEAT algorithm

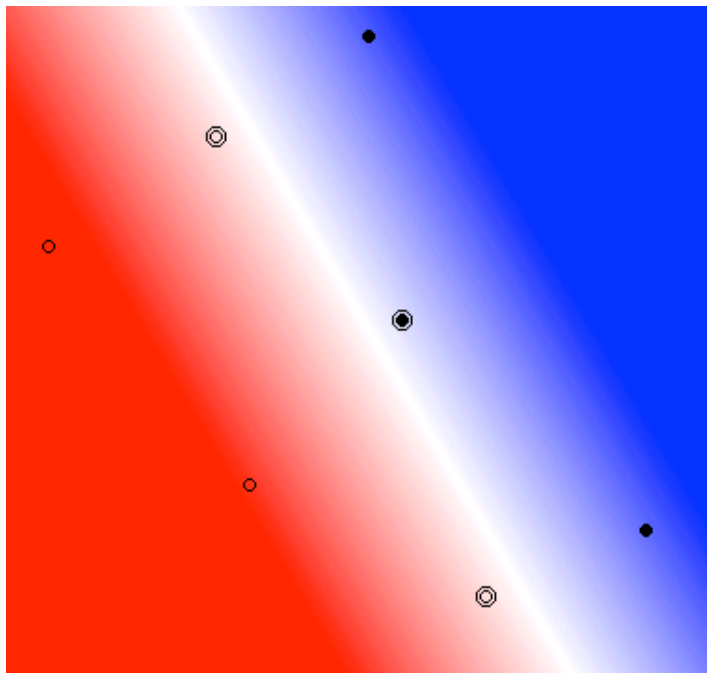
[Stanley & Miikkulainen 2002]

Performance vs Purity



Support Vector Machines

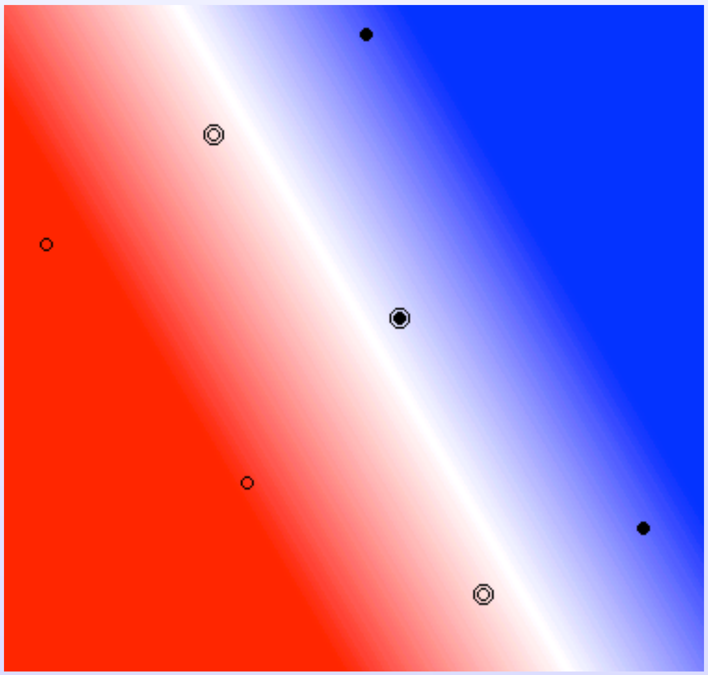
Linear problem



Consider a simple, linear separation problem

Support Vector Machines

• To find the hyperplane that gives the highest separation (lowest "energy"), we maximize the Lagrangian w.r.t α_i:



$$L = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$(\boldsymbol{x}_i, y_i)$ are training data

α_i are positive Lagrange multipliers

The solution is:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

Where $\alpha_i = 0$ for non support vectors

(images from applet at *http://svm.research.bell-labs.com/*)

Support Vector Machines

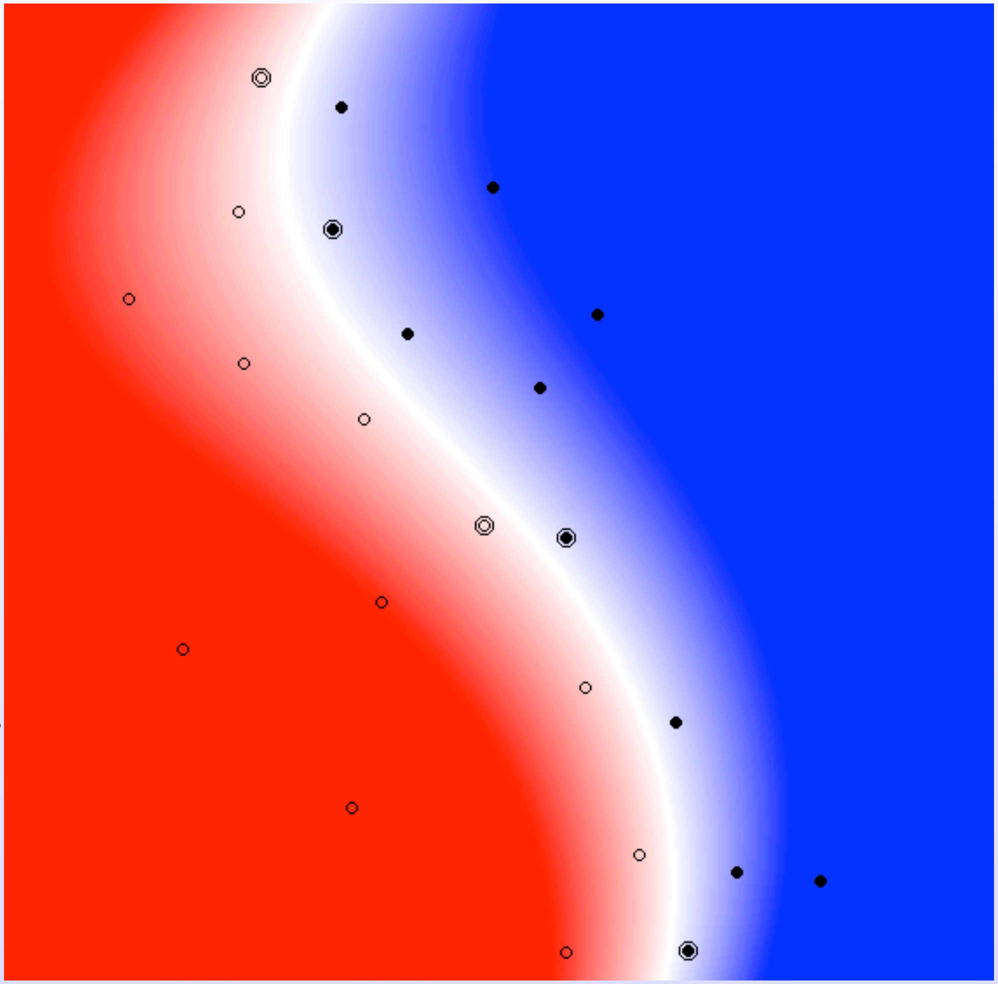
But not many problems of interest are linear.

Map data to higher dimensional space where separation can be made by hyperplanes

$$\Phi : R^d \mapsto \mathsf{H}$$

We want to work in our original space. Replace dot product with kernel function:

$$K(x,x) = \mathbf{x}_i \bullet \mathbf{x}_j$$

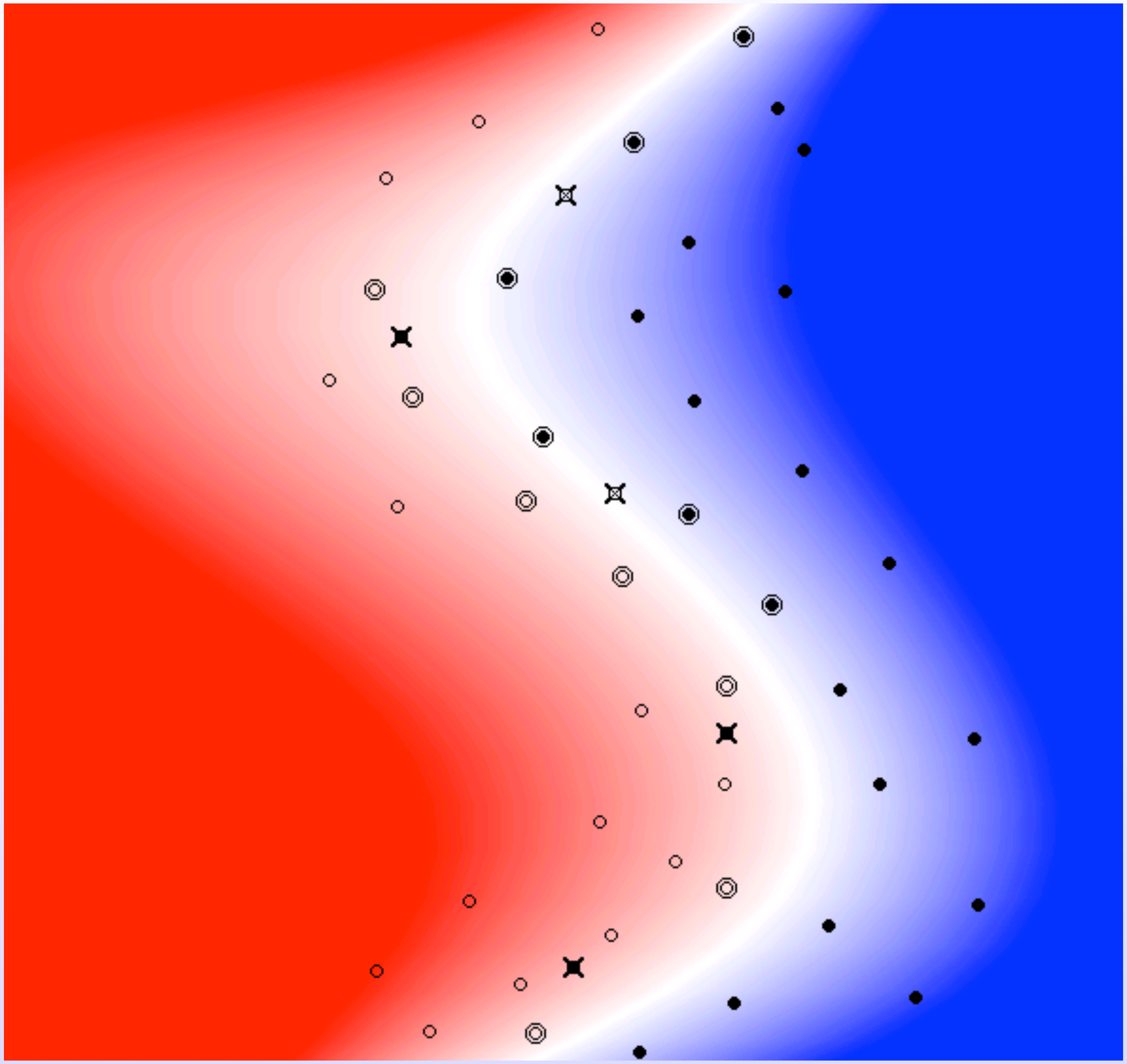


Support Vector Machines

Neither are entirely separable problems very difficult.

- Allow an imperfect decision boundary, but add a penalty.

- Training errors, points on the wrong side of the boundary, are indicated by crosses.



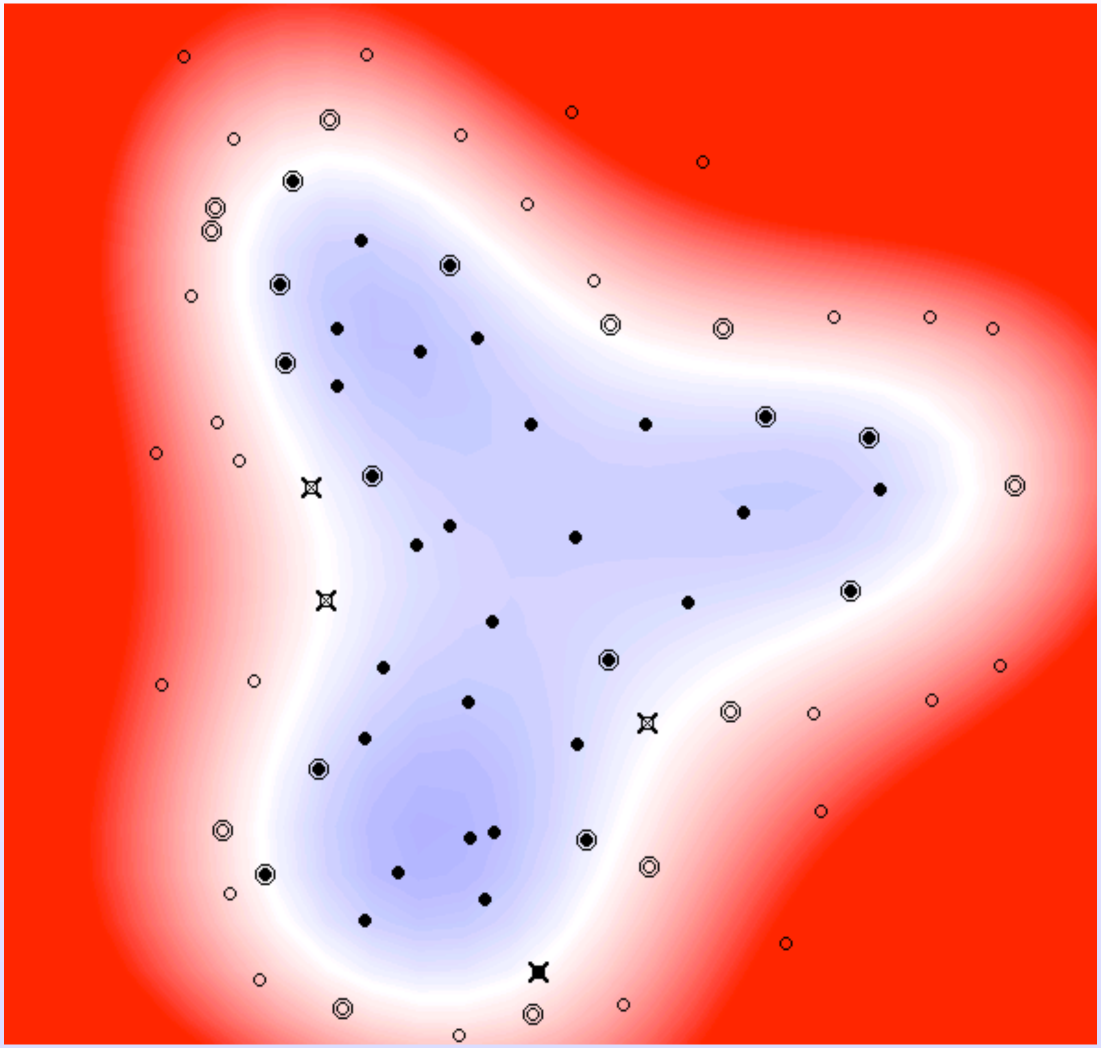
Support Vector Machines

We are not limited to linear or polynomial kernels.

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

Gives a highly flexible SVM

- Gaussian kernel SVMs outperformed PDEs in recognizing handwritten numbers from the USPS database.



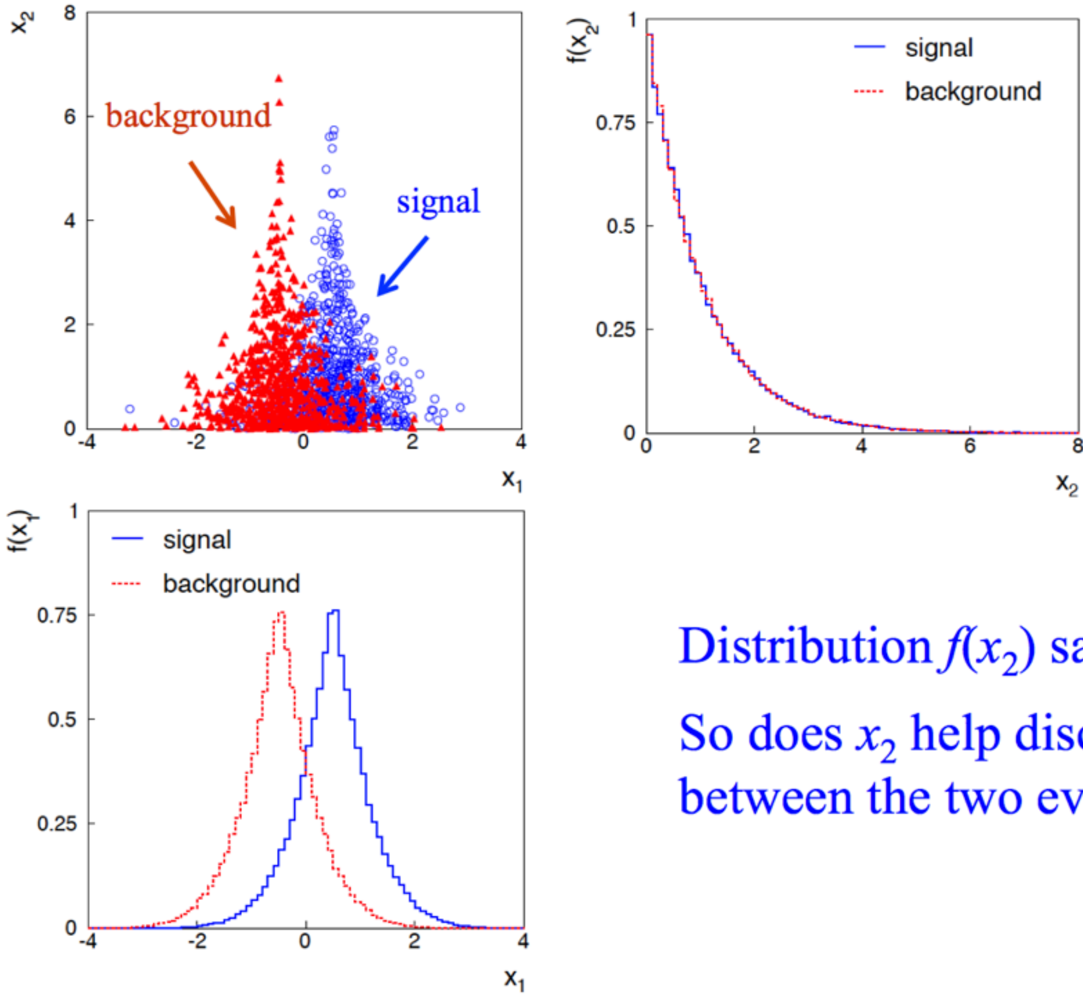
Algorithm Comparisons

Algorithm	**Advantages**	**Disadvantages**
Neural Nets	• Very fast evaluation	• Build structure by hand • Black box • Local optimization
PDE	• Transparent operation	• Slow evaluation • Requires high statistics
SVM	• Fast evaluation • Kernel positions chosen automatically • Global optimization	• Complex • Training can be time intensive • Kernel selection by hand

Example

2D example

Joint and marginal distributions of x_1, x_2



Distribution $f(x_2)$ same for s, b.

So does x_2 help discriminate between the two event types?

2D Example

Consider two variables, x_1 and x_2, and suppose we have formulas for the joint pdfs for both signal (s) and background (b) events (in real problems the formulas are usually not available).

$f(x_1|x_2) \sim$ Gaussian, different means for s/b,
Gaussians have same σ, which depends on x_2,
$f(x_2) \sim$ exponential, same for both s and b,
$f(x_1, x_2) = f(x_1|x_2)\, f(x_2)$:

$$f(x_1, x_2|\mathrm{s}) = \frac{1}{\sqrt{2\pi}\sigma(x_2)} e^{-(x_1-\mu_\mathrm{s})^2/2\sigma^2(x_2)}\, \frac{1}{\lambda} e^{-x_2/\lambda}$$

$$f(x_1, x_2|\mathrm{b}) = \frac{1}{\sqrt{2\pi}\sigma(x_2)} e^{-(x_1-\mu_\mathrm{b})^2/2\sigma^2(x_2)}\, \frac{1}{\lambda} e^{-x_2/\lambda}$$

$$\sigma(x_2) = \sigma_0 e^{-x_2/\xi}$$

G Cowan

Likelihood Ratio

Neyman-Pearson lemma says best critical region is determined by the likelihood ratio:

$$t(x_1, x_2) = \frac{f(x_1, x_2|\mathrm{s})}{f(x_1, x_2|\mathrm{b})}$$

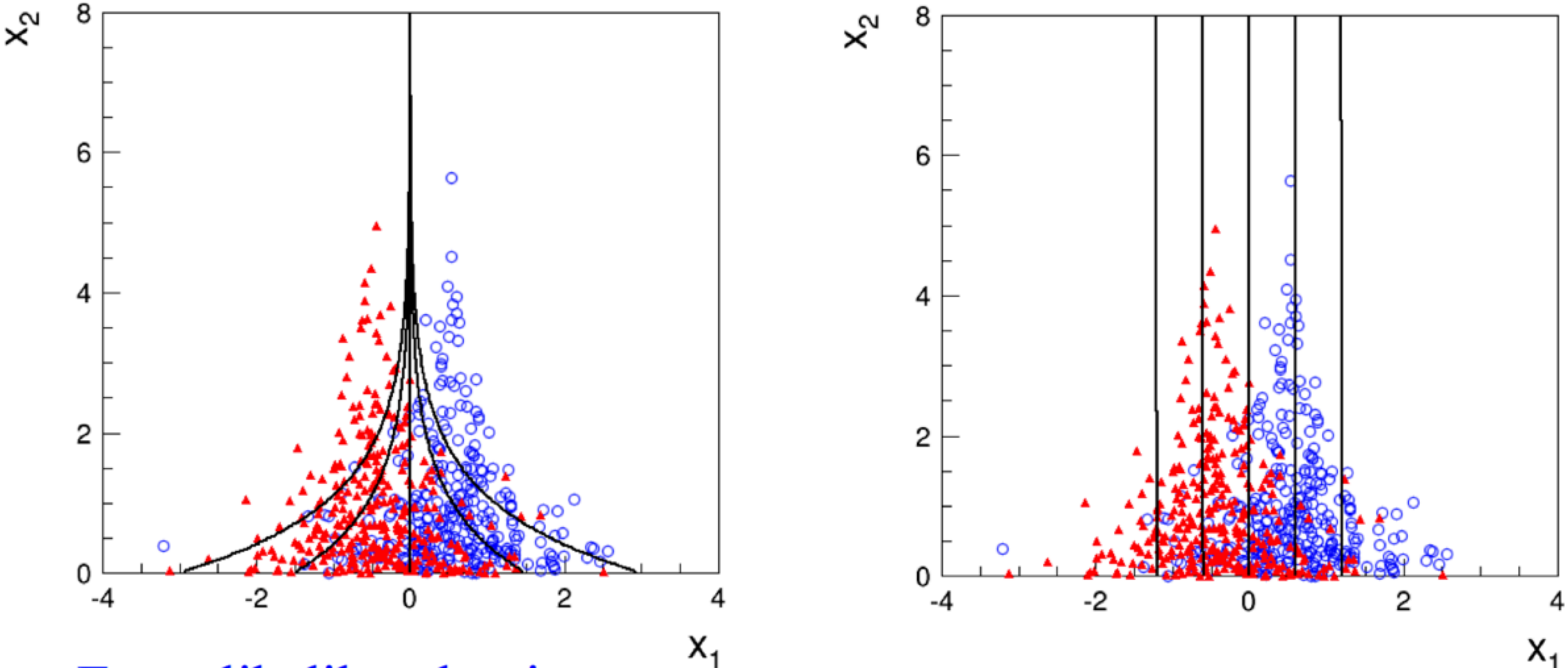
Equivalently we can use any monotonic function of this as a test statistic, e.g.,

$$\ln t = \frac{\frac{1}{2}(\mu_\mathrm{b}^2 - \mu_\mathrm{s}^2) + (\mu_\mathrm{s} - \mu_\mathrm{b})x_1}{\sigma_0^2 e^{-2x_2/\xi}}$$

Boundary of optimal critical region will be curve of constant ln t, and this depends on x_2!

G Cowan

Contours of constant MVA output

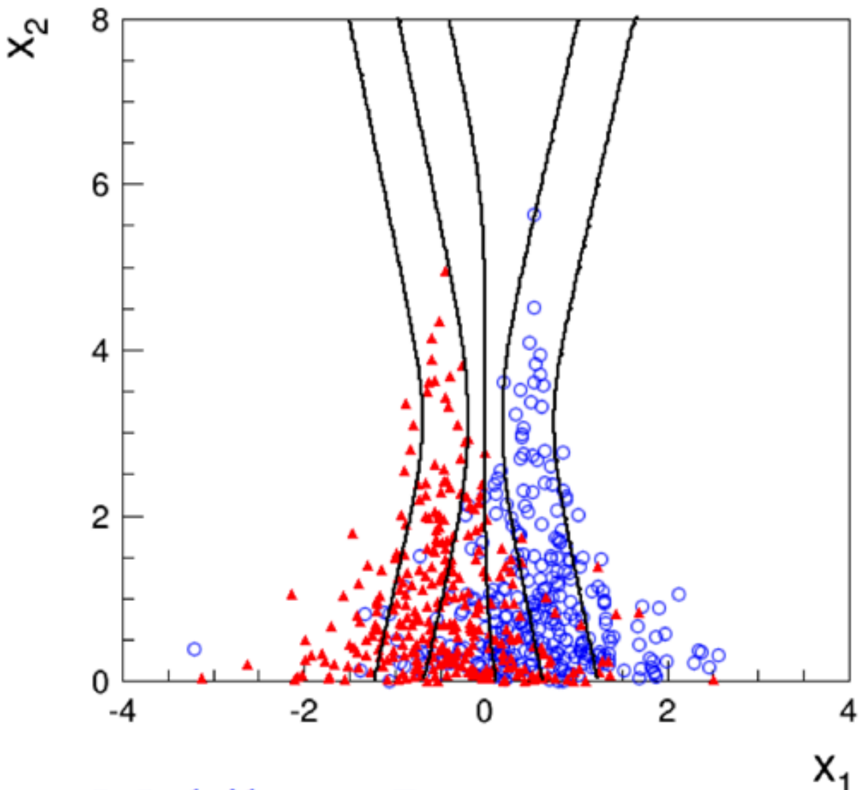


Exact likelihood ratio

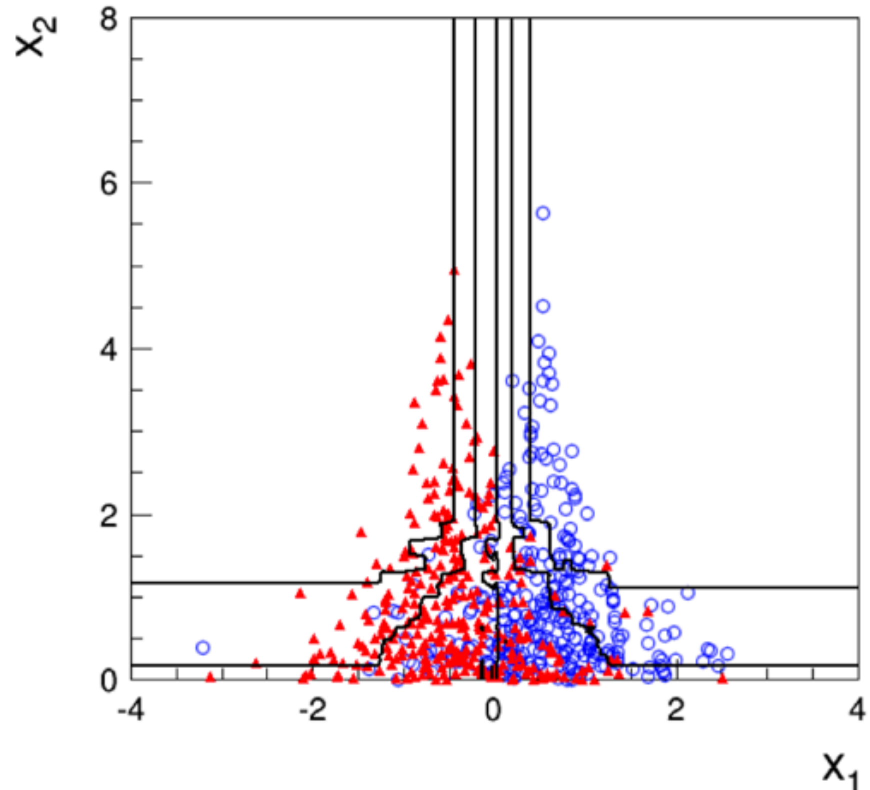
Fisher discriminant

G Cowan

Contours of constant MVA output



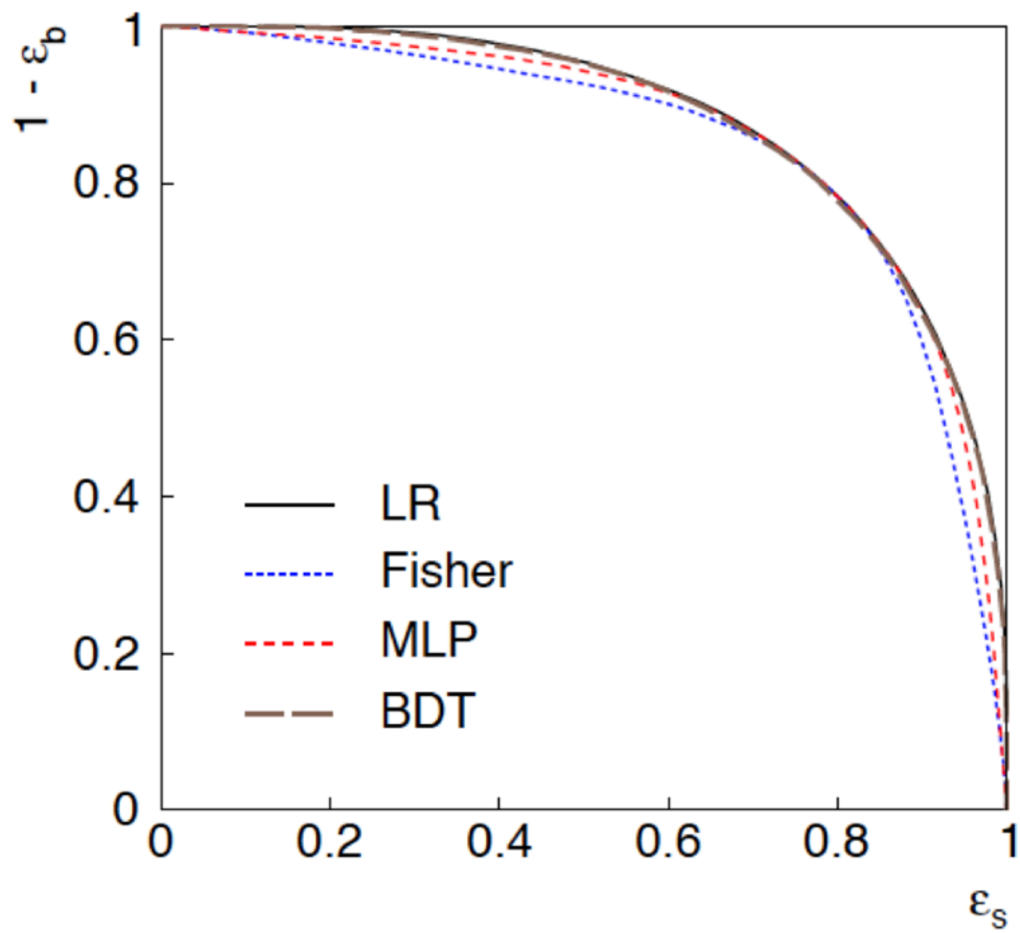
Multilayer Perceptron
1 hidden layer with 2 nodes



Boosted Decision Tree
200 iterations (AdaBoost)

Training samples: 10^5 signal and 10^5 background events

ROC



ROC = "receiver operating characteristic" (term from signal processing).

Shows (usually) background rejection ($1-\varepsilon_b$) versus signal efficiency ε_s.

Higher curve is better; usually analysis focused on a small part of the curve.

Summary

<u>Machine Learning is powerful</u>
main purpose is dimensional reduction

several tools: NN, SVM, KDE
others not discussed: BDT, PCA

Each have strengths and weaknesses