Likelihoods for distributions - summary

- **Bayesian inference unchanged**

  → simply insert L of distribution to calculate P(H|data)

$$P(H_{s+b} \mid \vec{N}) = \frac{L(\vec{N} \mid H_{s+b}) P(H_{s+b})}{L(\vec{N} \mid H_{s+b}) P(H_{s+b}) + L(\vec{N} \mid H_b) P(H_b)}$$

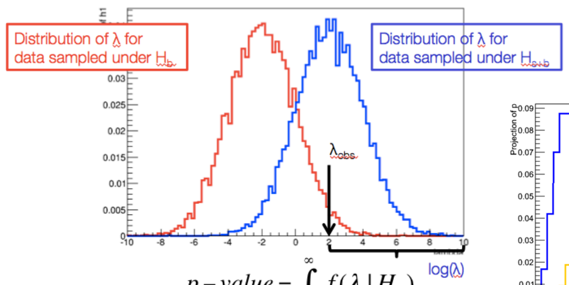
- **Frequentist inference procedure *modified***

  → Pure P(data|hypo) not useful for non-counting data
  → Order all possible data with a (LR) test statistic in 'extremity'
  → Quote p(data|hypo) as 'p-value' for hypothesis
  Probability to obtain observed data, *or more extreme,* is X%

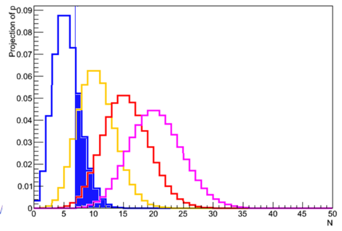
'Probability to obtain 13 or more 4-lepton events under the no-Higgs hypothesis is 10^{-7}'

'Probability to obtain 13 or more 4-lepton events under the SM Higgs hypothesis is 50%'

- **Definition: p-value**



$$p-value = \int_{\lambda_{obs}}^{\infty} f(\lambda \mid H_b)$$



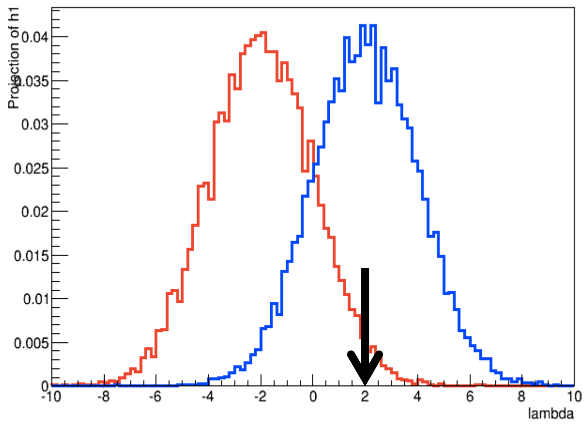
The likelihood principle

- Note that ‘ordering procedure’ introduced by test statistic also has a profound implication on interpretation
- Bayesian inference only uses the Likelihood of the observed data

$$P(H_{s+b} \mid \vec{N}) = \frac{L(\vec{N} \mid H_{s+b})P(H_{s+b})}{L(\vec{N} \mid H_{s+b})P(H_{s+b}) + L(\vec{N} \mid H_b)P(H_b)}$$

- While the observed Likelihood Ratio also only uses likelihood of observed data.

$$\lambda(\vec{N}) = \frac{L(\vec{N} \mid H_{s+b})}{L(\vec{N} \mid H_b)}$$



- **Distribution f(λ|N), and thus p-value, also uses likelihood of non-observed outcomes** (in fact Likelihood of every possible outcome is used)

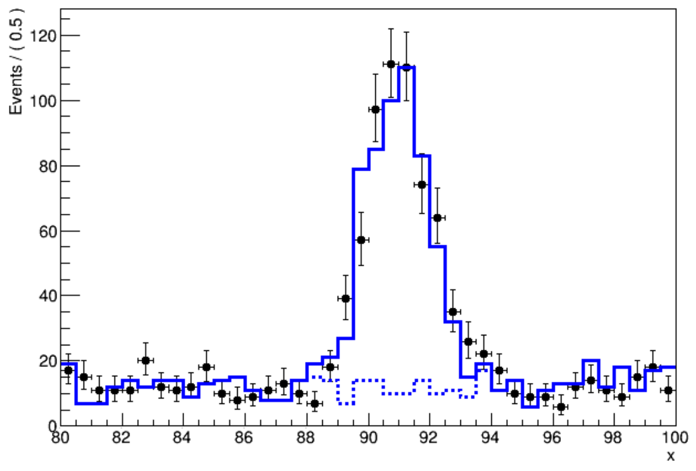
Likelihood Principle

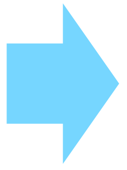
- In **Bayesian** methods and **likelihood-ratio** based methods, the probability (density) for obtaining the *data at hand is used (via the likelihood function), but probabilities for obtaining other data are not used!*
- In contrast, in typical **frequentist** calculations (e.g., a p-value which is the probability of obtaining a value as extreme or *more extreme than that observed), one uses probabilities of data not seen.*
- This difference is captured by the *Likelihood Principle**:

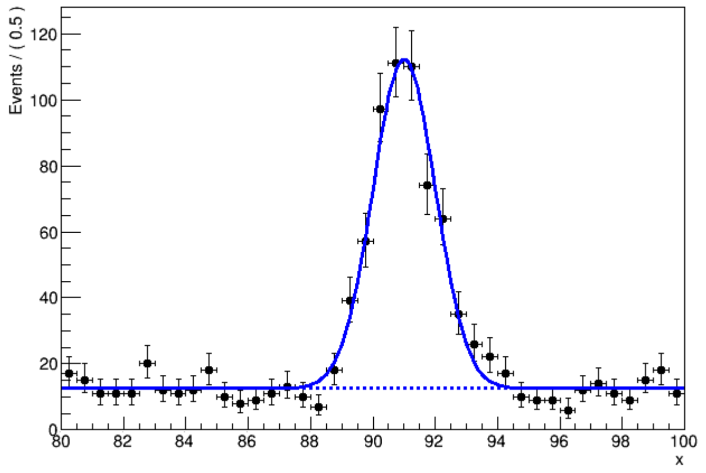
  If two experiments yield likelihood functions which are proportional, then Your inferences from the two experiments should be identical.

Generalizing to continuous distributions

- Can generalize likelihood to described continuous distributions







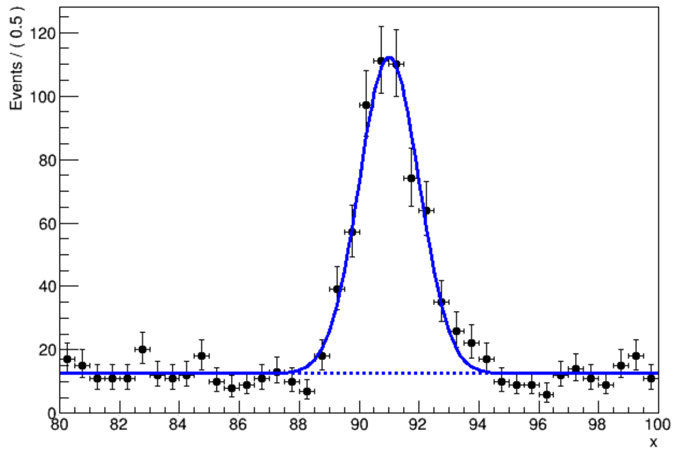
$$L(\vec{N}) = \prod_i Poisson(N_i \mid \tilde{s}_i + \tilde{b}_i)$$

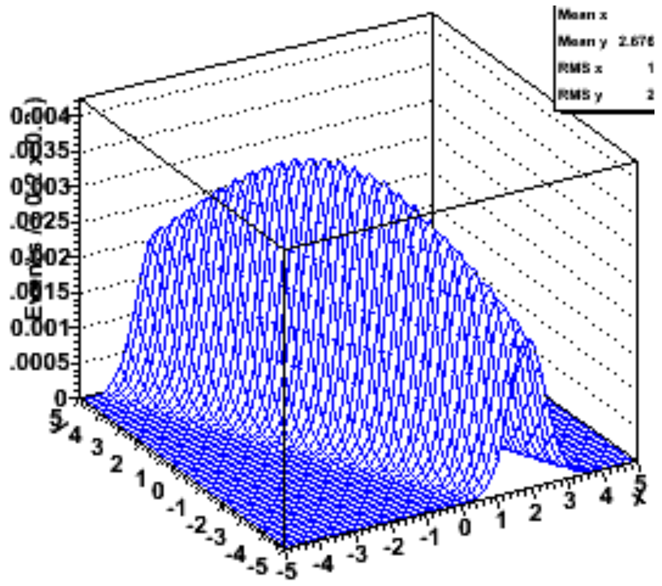
$$L(\vec{m}_{ll}) = \prod_i \left[\tilde{f}_{sig} \mathrm{Gauss}(m_{ll}^{(i)}, 91, 1) + (1 - \tilde{f}_{sig}) \cdot \mathrm{Uniform}(m_{ll}^{(i)}) \right]$$

- Probability model becomes a probability *density* model
  - Integral of probability density model over full space of observable is always 1 (just like sum of bins of a probability model is always 1)
  - Integral of p.d.f. over a range of observable results in a probability
- Probability density models have (in principle) more analyzing power
  - But relies on your ability to formulate an analytical model (e.g. hard at LHC)

Generalizing to multiple dimensions

- Can also generalize likelihood models to distributions in *multiple* observables





$$L(\vec{x}) = \prod_i f(x_i)$$

$$L(\vec{x},\vec{y}) = \prod_i f(x_i, y_i)$$

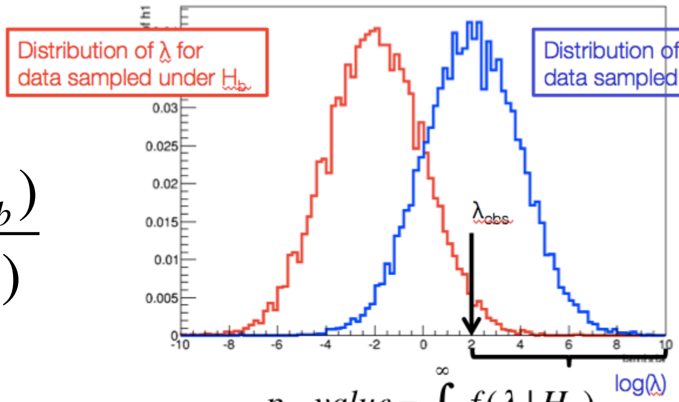
- Neither generalization (binned→continuous, one→multiple observables) has any further consequences for Bayesian or Frequentist inference procedures

Wouter Verkerke, NIKHEF

The Likelihood Ratio test statistic as tool for event selection

- Note that hypothesis testing with two simple hypotheses for observable distributions, exactly describes ‘event selection’ problem
- In fact we have already ‘solved’ the optimal event selection problem! Given two hypothesis H_{s+b} and H_b that predict an complex multivariate distribution of observables, **you can always classify all events in terms of ‘signal-likeness’ (a.k.a ‘extremity’) with a likelihood ratio**

$$\lambda(\vec{x},\vec{y},\vec{z},\ldots) = \frac{L(\vec{x},\vec{y},\vec{z},\ldots \mid H_{s+b})}{L(\vec{x},\vec{y},\vec{z},\ldots \mid H_b)}$$

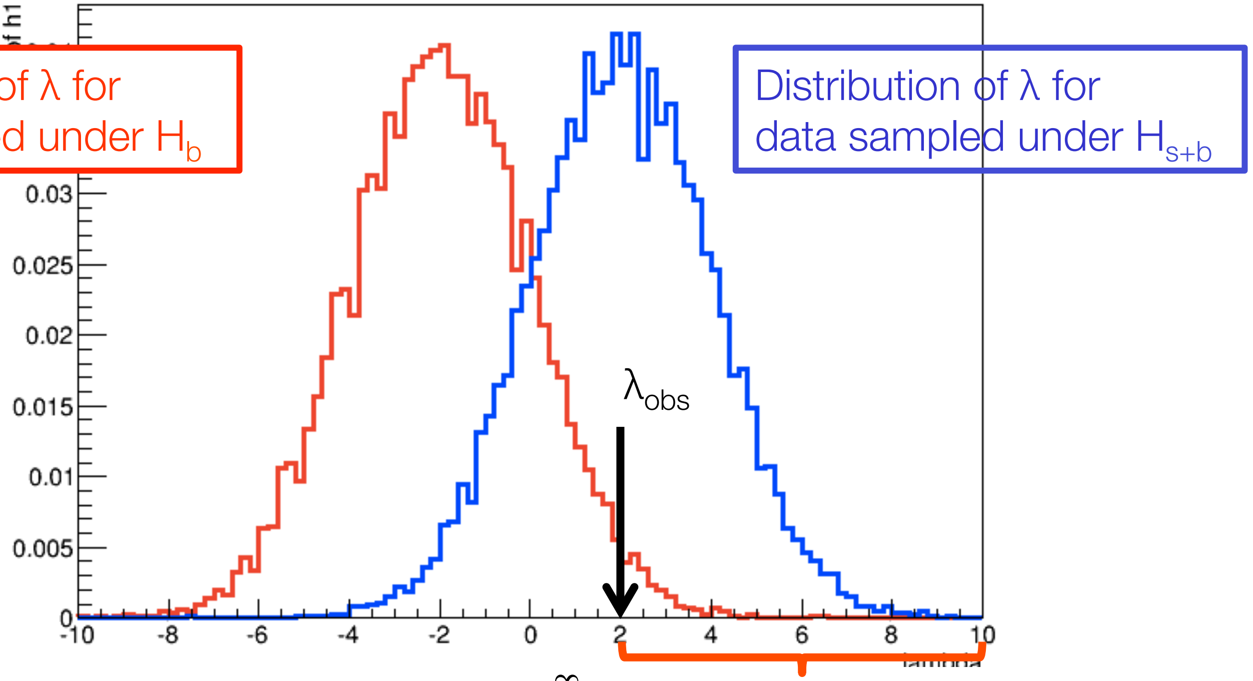


$$p-value = \int_{\lambda_{obs}}^{\infty} f(\lambda \mid H_b)$$

- So far we have exploited λ to calculate a frequentist p-value **tomorrow now explore properties ‘cut on λ’ as basis of (optimal) event selection**

The distribution of the test statistic

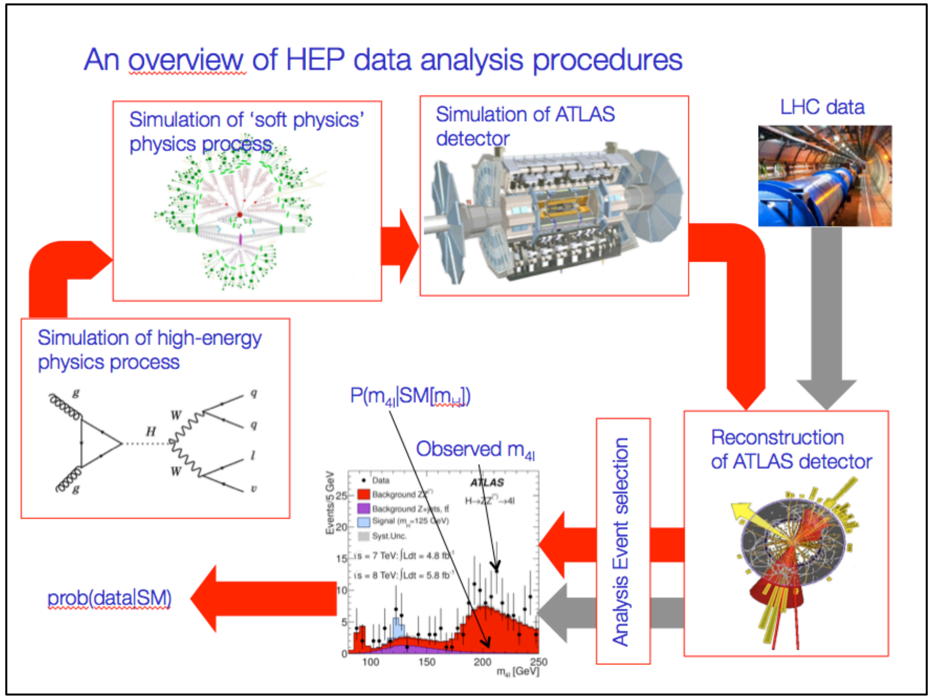
- Distribution of a test statistic is *generally not known*
- Use toy MC approach to approximate distribution
  - Generate many toy datasets N under H_b and H_{s+b} and evaluate $\lambda(N)$ for each dataset



$$p-value = \int_{\lambda_{obs}}^{\infty} f(\lambda \mid H_b)$$

Intermezzo – Generating toy data

- Two approaches to obtaining simulated data
- First approach is 'Physics Monte Carlo Chain', described earlier
  - Time consuming, but injects detailed knowledge about physics, detector, output is full collision information, and relation to underlying theory details
- Alternative approach is sample sampling the probability model 'toy MC'
  - Fast (generally), only requires access to probability model
  - Can only produce datasets with observables that are described by the probability model → Sufficient to study distribution of test statistics


An overview of HEP data analysis procedures
Simulation of 'soft physics' physics process
Simulation of ATLAS detector
LHC data
Simulation of high-energy physics process
P(m4l|SM[mH])
Observed m4l
Analysis Event selection
Reconstruction of ATLAS detector
prob(data|SM)
ATLAS
H→ZZ(*)→4l
Data
Background ZZ(*)
Background Z+jets, tt̄
Signal (mH=125 GeV)
Syst.Unc.
√s = 7 TeV: ∫Ldt = 4.8 fb⁻¹
√s = 8 TeV: ∫Ldt = 5.8 fb⁻¹
Events/5 GeV
m4l [GeV]

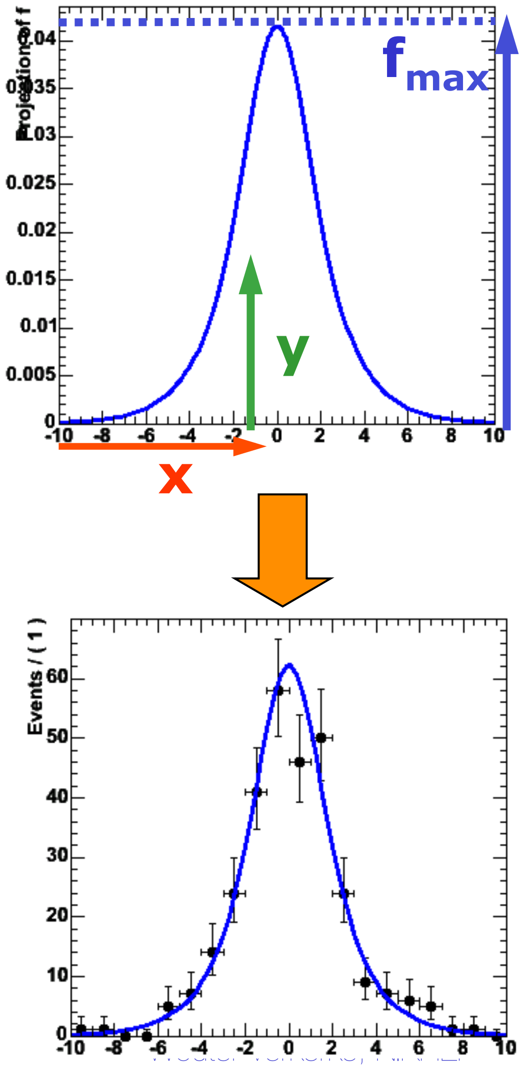
How do you efficiently generate a toy dataset from a probability model?

- Simplest method is accept/reject sampling

1) Determine maximum of function f_{max}
2) Throw random number x
3) Throw another random number y
4) If $y<f(x)/f_{max}$ keep x, otherwise return to step 2)

– PRO: Easy, always works

– CON: It can be inefficient if function is strongly peaked. Finding maximum empirically through random sampling can be lengthy in >2 dimensions

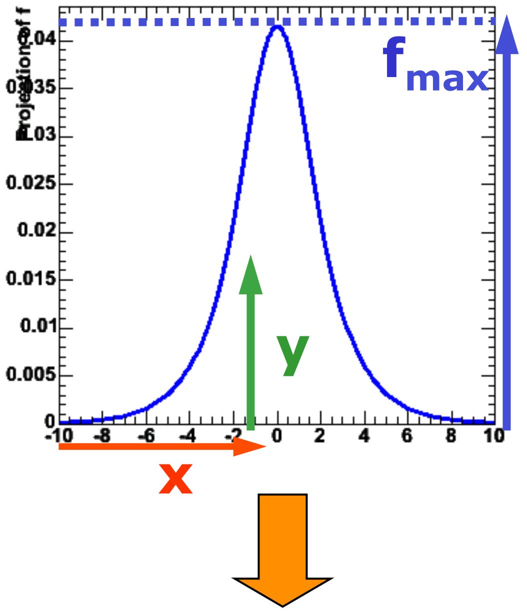


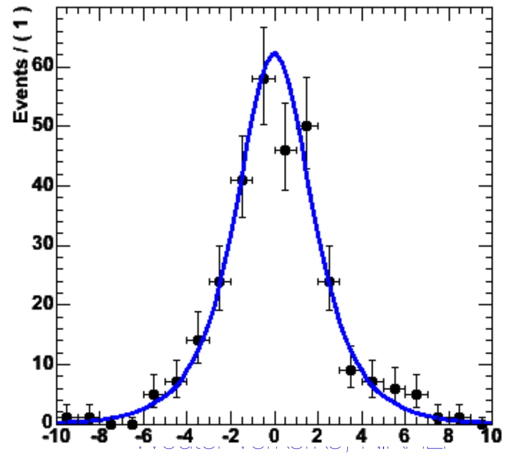
How do you efficiently generate a toy dataset from a probability model?

- Simplest method is accept/reject sampling

    1) Determine maximum of function f_{max}
    2) Throw random number x
    3) Throw another random number y
    4) If $y<f(x)/f_{max}$ keep x,
       otherwise return to step 2)

    – PRO: Easy, always works
    – CON: It can be inefficient if function is strongly peaked.
      Finding maximum empirically through random sampling can be lengthy in >2 dimensions





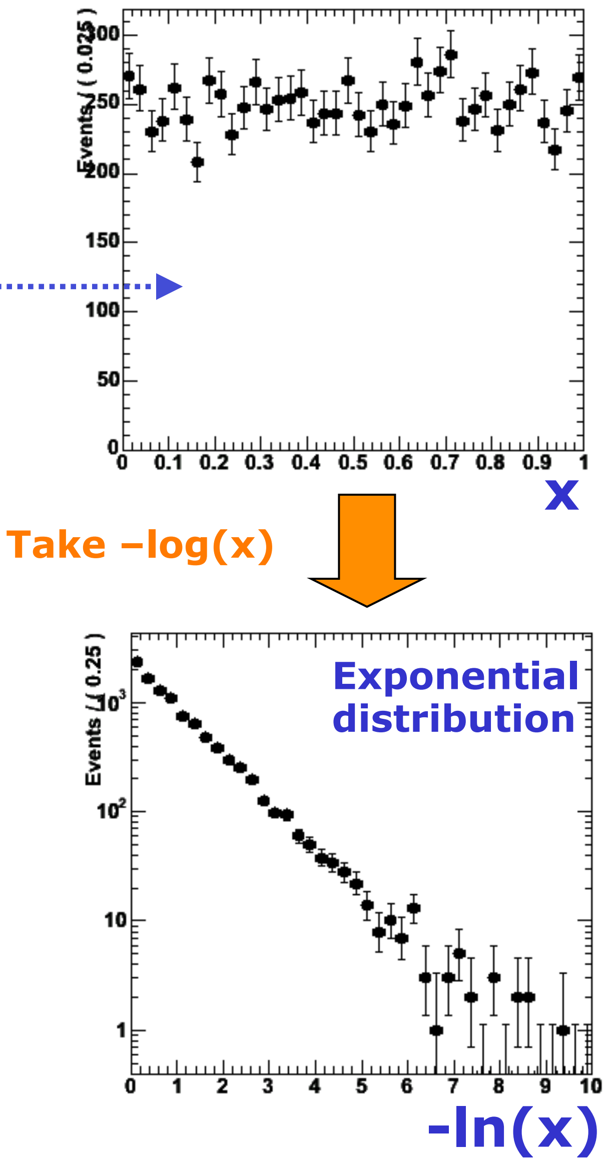
Toy MC generation – Inversion method

- Fastest: function inversion

1) Given f(x) find inverted function F(x) so that f(F(x)) = x
2) Throw uniform random number x
3) Return F(x)

– PRO: Maximally efficient

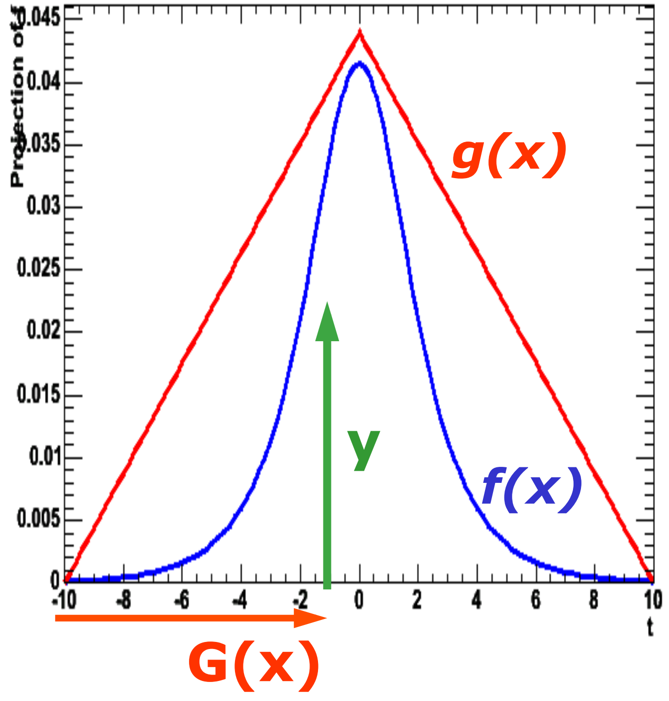
– CON: Only works for invertible functions



Toy MC Generation – importance sampling

- Hybrid: Importance sampling

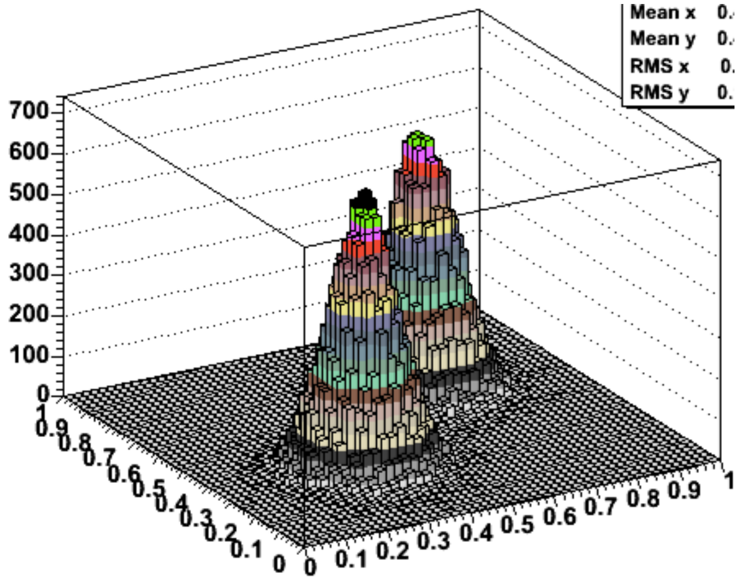
1) Find 'envelope function' g(x) that is invertible into G(x) and that fulfills g(x)>=f(x) for all x
2) Generate random number x from G using inversion method
3) Throw random number 'y'
4) If y<f(x)/g(x) keep x, otherwise return to step 2

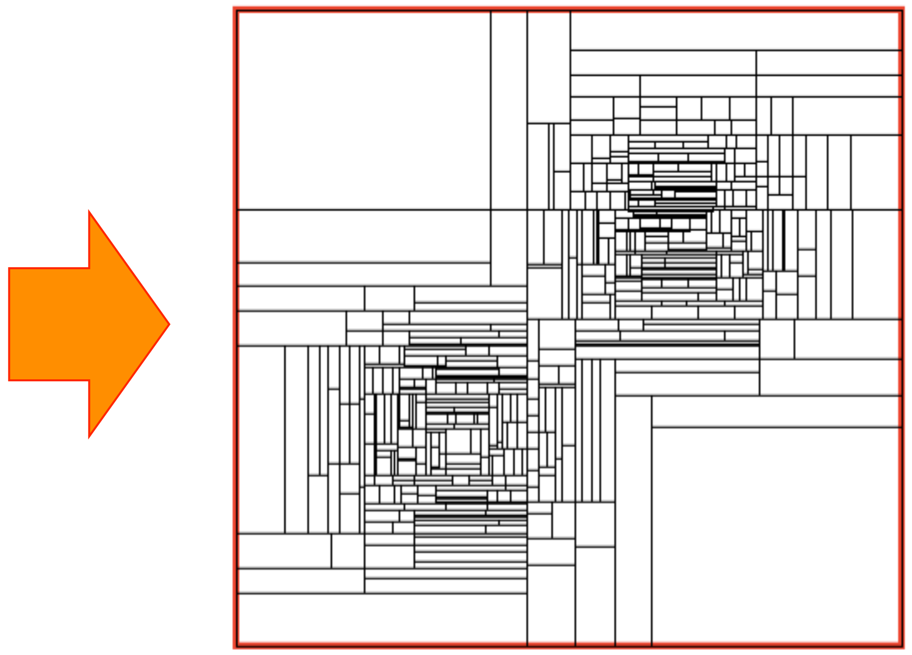


  - PRO: Faster than plain accept/reject sampling
    Function does not need to be invertible
  - CON: Must be able to find invertible envelope function

Toy MC Generation – importance sampling in >1D

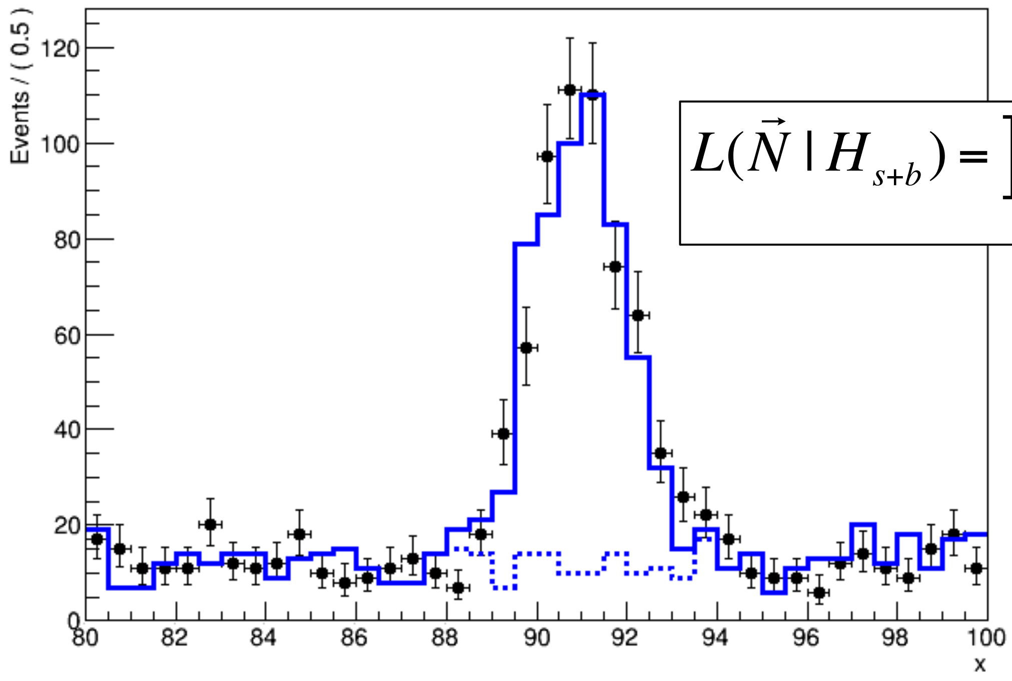
- General algorithms exists that can construct empirical envelope function
  - Divide observable space recursively into smaller boxes and take uniform distribution in each box
  - Example shown below from FOAM algorithm





Toy MC Generation – importance sampling in >1D

- For *binned distributions*, can generate content of each bin on toy dataset independently, using a Poisson process



$$L(\vec{N} \mid H_{s+b}) = \prod_i Poisson(N_i \mid \tilde{s}_i + \tilde{b}_i)$$

- Note that efficient generation of Poisson random number relies on combination of importance sampling (for small μ, using exponential envelope, for large μ using Cauchy distribution)

Roadmap for this course

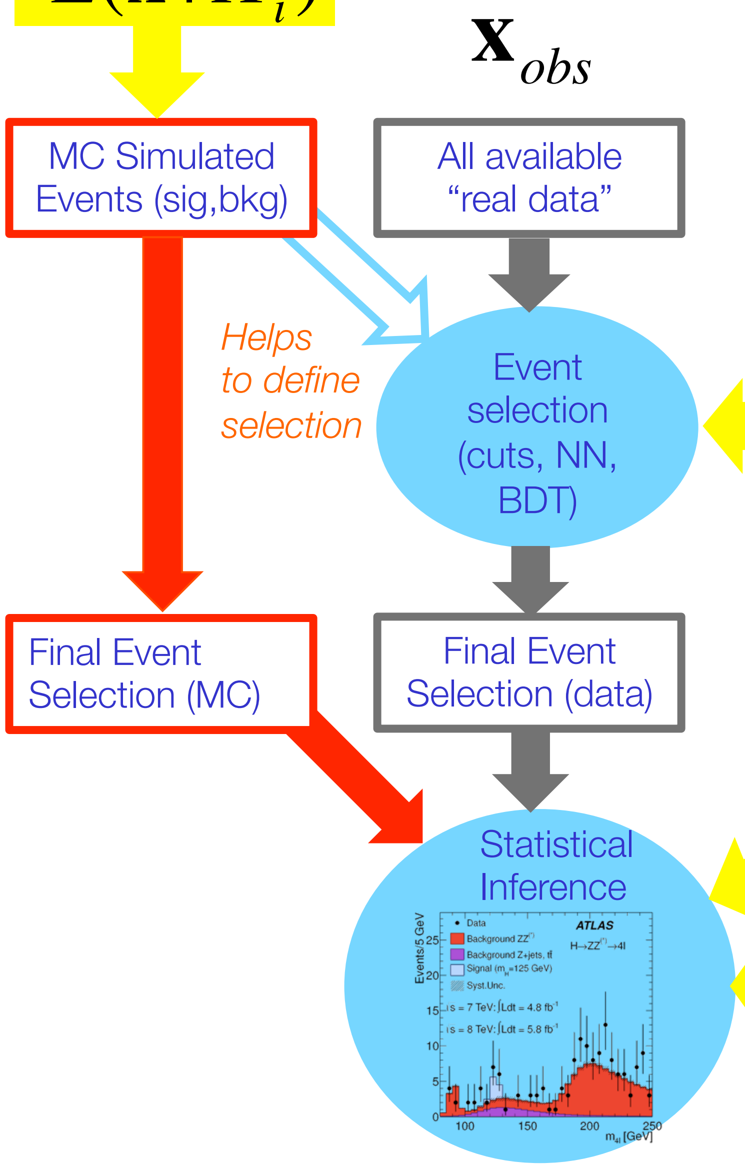
- Start with basics, gradually build up to complexity of

Statistical tests with simple hypotheses for counting data	*"What do we mean with probabilities"*
Statistical tests with simple hypotheses for distributions	*"p-values"*
Hypothesis testing as basis for event selection	*"Optimal event selection & machine learning"*
Composite hypotheses (with parameters) for distributions	*"Confidence intervals, Maximum Likelihood"*
Statistical inference with nuisance parameters	*"Fitting the background"*
Response functions and subsidiary measurements	*"Sideband fits and systematic uncertainties"*

HEP workflow versus statistical concepts

“Likelihood”

$$L(\mathbf{x} \mid H_i)$$



$\mathbf{x}_{obs}$
MC Simulated Events (sig,bkg)
All available “real data”
Helps to define selection
Event selection (cuts, NN, BDT)
Final Event Selection (MC)
Final Event Selection (data)
Statistical Inference
Events/5 GeV
Data
Background ZZ(*)
Background Z+jets, tt̄
Signal (mH=125 GeV)
Syst.Unc.
ATLAS
H→ZZ(*)→4l
√s = 7 TeV: ∫Ldt = 4.8 fb-1
√s = 8 TeV: ∫Ldt = 5.8 fb-1
m4l [GeV]

“Likelihood Ratio”

$$\lambda(\mathbf{x}) \equiv \frac{L(\mathbf{x} \mid H_{s+b})}{L(\mathbf{x} \mid H_b)} > \alpha$$

“p-value from Likelihood Ratio test statistic”

$$p_0(\mathbf{x} \mid H_i) = \int_{\lambda_{obs}}^{\infty} f(\lambda \mid H_i)$$

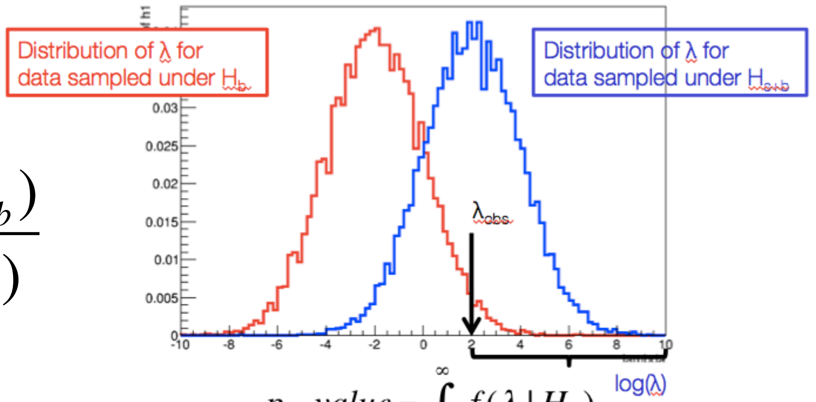
$$P(H_{s+b} \mid \mathbf{x}) = \frac{L(\mathbf{x} \mid H_{s+b}) P(H_{s+b})}{L(\mathbf{x} \mid H_{s+b}) P(H_{s+b}) + L(\mathbf{x} \mid H_b) P(H_b)}$$

“Bayesian posterior probability”

The Likelihood Ratio test statistic as tool for event selection

- Note that hypothesis testing with two simple hypotheses for observable distributions, exactly describes ‘event selection’ problem
- In fact we have already ‘solved’ the optimal event selection problem! Given two hypothesis H_{s+b} and H_b that predict an complex multivariate distribution of observables, **you can always classify all events in terms of ‘signal-likeness’ (a.k.a ‘extremity’) with a likelihood ratio**

$$\lambda(\vec{x},\vec{y},\vec{z},\ldots) = \frac{L(\vec{x},\vec{y},\vec{z},\ldots \mid H_{s+b})}{L(\vec{x},\vec{y},\vec{z},\ldots \mid H_b)}$$



$$p-value = \int_{\lambda_{obs}}^{\infty} f(\lambda \mid H_b)$$

Wouter Verkerke

- So far we have exploited λ to calculate a frequentist p-value **tomorrow now explore properties ‘cut on λ’ as basis of (optimal) event selection**

Event selection

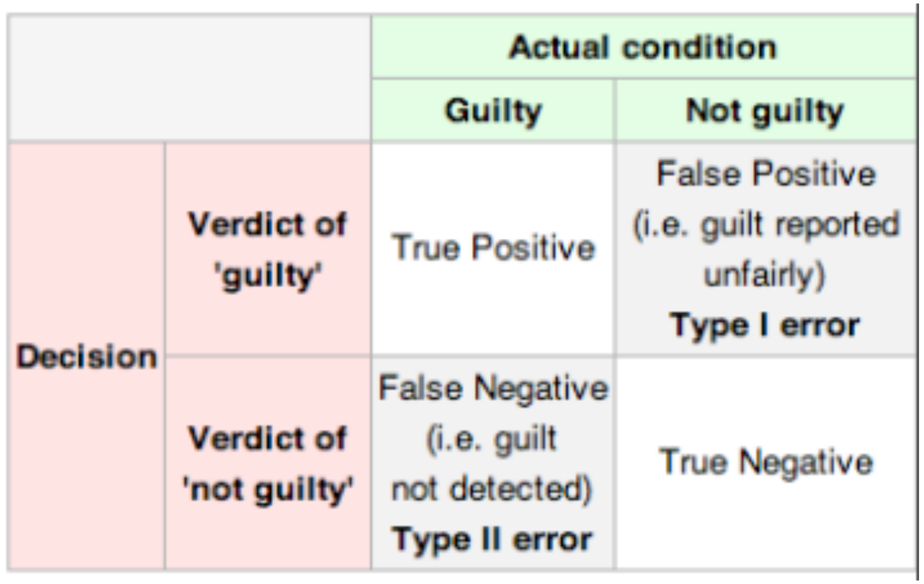
- The event selection problem:
  - Input: Two classes of events “signal” and “background”
  - Output: Two categories of events “selected” and “rejected”
- Goal: select as many signal events as possible,
  reject as many background events as possible

- Note that optimization goal as stated is ambiguous.
  - But can choose a well-defined by optimization goal by e.g. fixing desired background acceptance rate, and then choose procedure that has highest signal acceptance.

- Relates to “classical hypothesis testing”
  - Two competing hypothesis (traditionally named ‘null’ and ‘alternate’)
  - Here null = background, alternate = signal

Terminology of classical hypothesis testing

- Definition of terms
  - Rate of type-I error = α
  - Rate of type-II error = β
  - Power of test is $1-\beta$



		Actual condition	
		Guilty	**Not guilty**
Decision	**Verdict of 'guilty'**	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

- Treat hypotheses asymmetrically
  - Null hypo is usually special → Fix rate of type-I error
  - Criminal convictions: Fix rate of unjust convictions
  - Higgs discovery: Fix rate of false discovery
  - Event selection: Fix rate of background that is accepted
- Now can define a well stated goal for optimal testing
  - Maximize the power of test (minimized rate of type-II error) for given α
  - Event selection: Maximize fraction of signal accepted

The Neyman-Pearson lemma

- In 1932-1938 Neyman and Pearson developed a theory in which one must consider competing hypotheses
  - Null hypothesis (H_0) = Background only
  - Alternate hypotheses (H_1) = e.g. Signal + Background

  and proved that

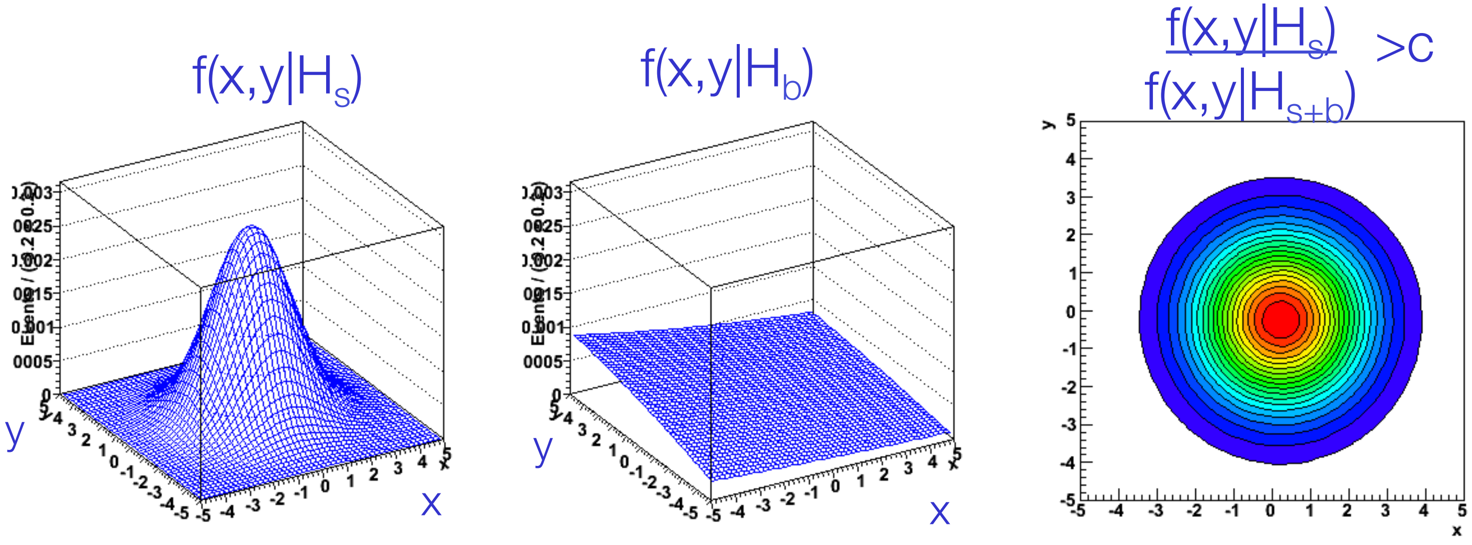
- The region W that minimizes the rate of the type-II error (not reporting true discovery) is a contour of the Likelihood Ratio

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

- Any other region of the same size will have less power

The Neyman-Pearson lemma

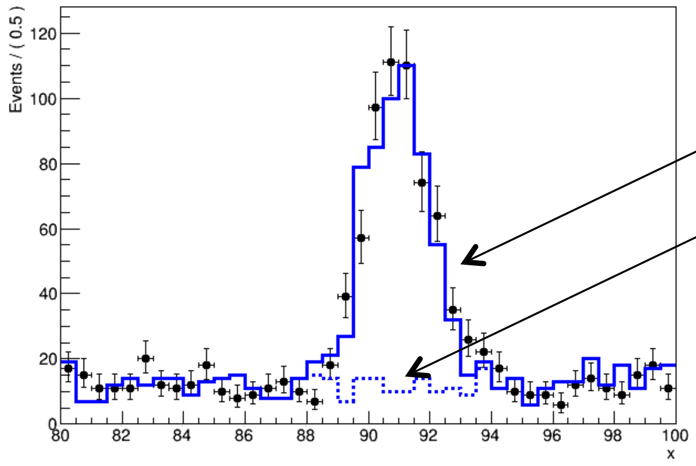
- Example of application of NP-lemma with two observables



- Cut-off value c controls type-I error rate ('size' = bkg rate)
  Neyman-Pearson: LR cut gives best possible 'power' = signal eff.
- So why don't we *always* do this? (instead of training neural networks, boosted decision trees etc)

Why Neyman-Pearson doesn't always help

- The problem is that we usually don't have explicit formulae for the pdfs $f(\vec{x}|\mathrm{s}),\ f(\vec{x}|\mathrm{b})$.
- Instead we may have Monte Carlo samples for signal and background processes
  - Difficult to reconstruct analytical distributions of pdfs from MC samples, especially if number of dimensions is large
- If physics problem has only few observables can still estimate estimate pdfs with histograms or kernel estimation,
  - But in such cases one can also forego event selection and go straight to hypothesis testing / paramater estimation with all events



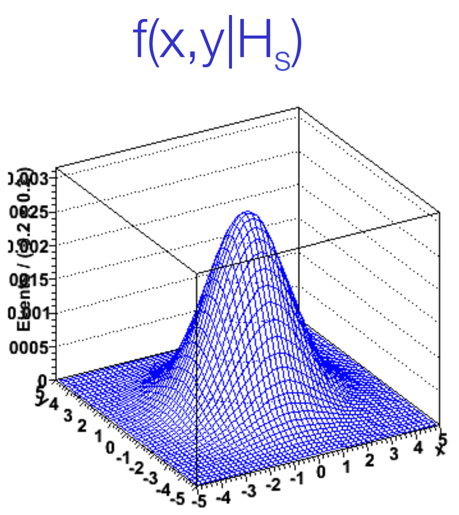
Approximation of true f(x|s)

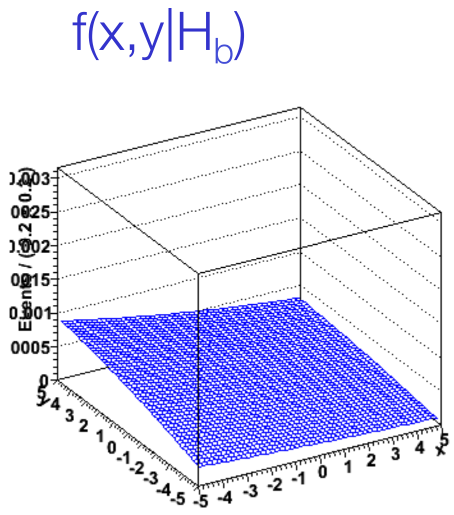
Approximation of true f(x|b)

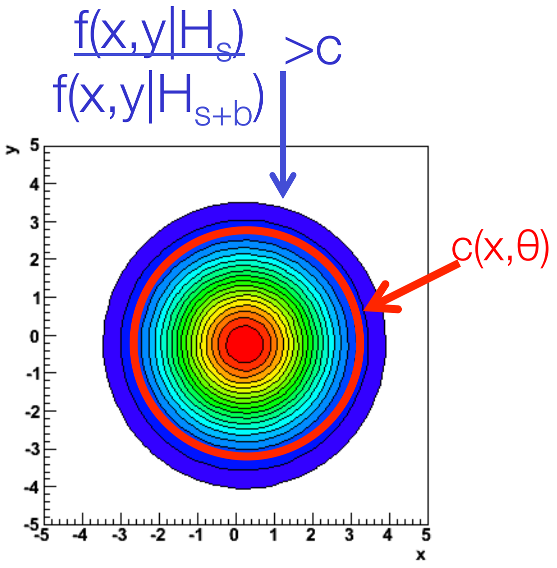
Hypothesis testing with a large number of observables

- When number of observables is large follow different strategy
- Instead of aiming at approximating p.d.f.s f(x|s) and f(x|b) aim to approximate decision boundary with an empirical parametric form

$$A_\alpha(\vec{x}) = \left[\frac{f(\vec{x}\,|\,s)}{f(\vec{x}\,|\,s+b)} > \alpha \right] \quad \Rightarrow \quad A_\alpha(\vec{x}) = c(\vec{x},\vec{\theta})$$

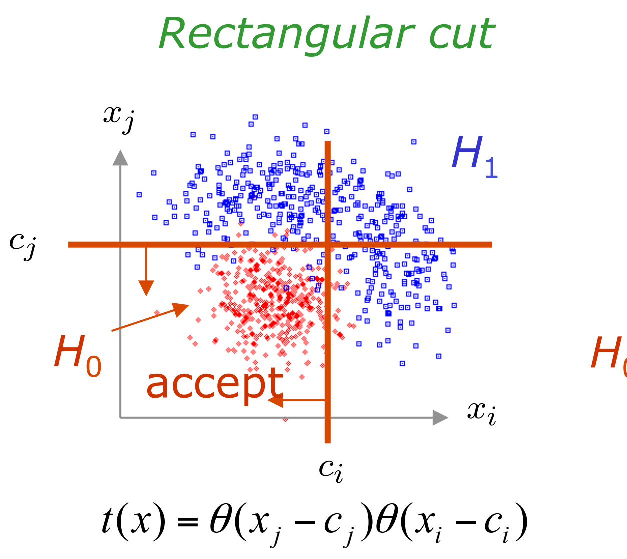




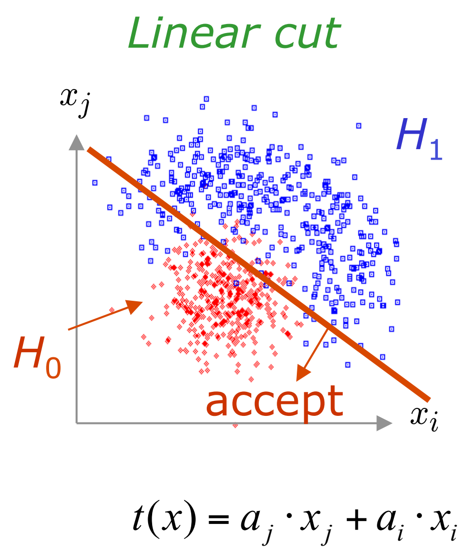


Empirical parametric forms of decision boundaries

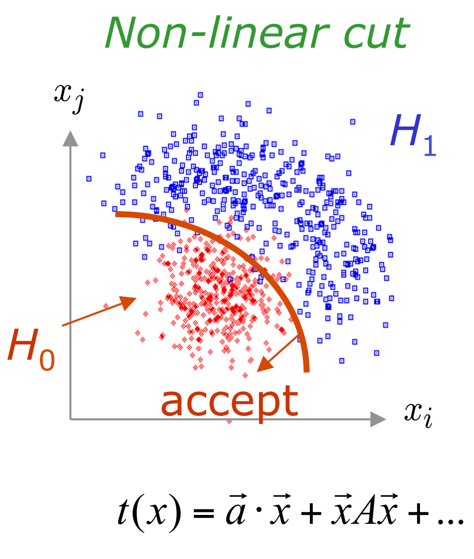
- Can in principle choose any type of Ansatz parametric shape



$$t(x) = \theta(x_j - c_j)\theta(x_i - c_i)$$



$$t(x) = a_j \cdot x_j + a_i \cdot x_i$$



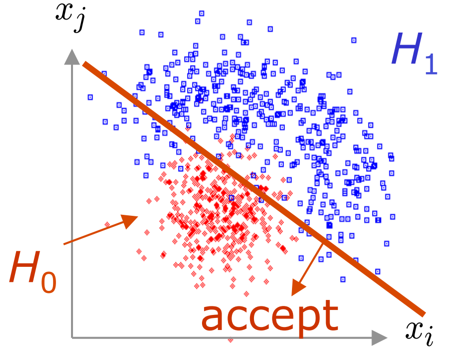
$$t(x) = \vec{a} \cdot \vec{x} + \vec{x} A \vec{x} + \ldots$$

- Goal of Ansatz form is estimate of a ‘signal probability’ for every event in the observable space x (just like the LR)
- Choice of desired type-I error rate (selected background rate), can be set later by choosing appropriate cut on Ansatz test statistic.

The simplest Ansatz – A linear disciminant

- A linear discriminant constructs t(x) from a linear combination of the variables x_i

$$t(\vec{x}) = \sum_{i=1}^{N} a_i x_i = \vec{a} \cdot \vec{x}$$



  - A cut on t(x) results in a linear decision plane in x-space

- What is optimal choice of direction vector a?
- Solution provided by the Fisher – The Fisher discriminant

$$F(\vec{x}) = \overbrace{(\vec{\mu}_S - \vec{\mu}_B)^T V^{-1}}^{\vec{a}} \vec{x}$$

Mean values in x_i for sig,bkg

Inverse of variance matrix of signal/background (assumed to be the same)

R.A. Fisher
Ann. Eugen. 7(1936) 179.

The simplest Ansatz – A linear disciminant

- Operation advantage of Fisher discrimant is that test statistic parameters can be *calculated* (no iterative estimation is required)

$$F(\vec{x}) = \overbrace{\left(\vec{\mu}_S - \vec{\mu}_B\right)^T V^{-1}}^{\vec{a}}\vec{x}$$

Mean values in x_i for sig,bkg

Inverse of variance matrix of signal/background (assumed to be the same)

R.A. Fisher
Ann. Eugen. 7(1936) 179.

- Fisher discriminant is optimal test statistic (i.e. maps to Neyman Pearson Likelihood Ratio) for case where both hypotheses are multivariate Gaussian distributions with the same variance, but diffferent means

$$\left.\begin{aligned} f(x\,|\,s) &= Gauss(\vec{x}-\vec{\mu}_s, V) \\ f(x\,|\,b) &= Gauss(\vec{x}-\vec{\mu}_b, V) \end{aligned}\right\}$$

Multivariate Gaussian distributions with **different means** but **same width** for signal and background

The simplest Ansatz – A linear disciminant

- How the Fisher discriminant follows from the LR test statistic

$$-\log\left(\frac{f(x|s)}{f(x|b)}\right) = 0.5\left(\frac{x-\mu_s}{\sigma^2}\right)^2 - 0.5\left(\frac{x-\mu_b}{\sigma^2}\right)^2 + C$$

$$= 0.5\,\frac{x^2 - 2x\mu_s + \mu_s^2 - x^2 + 2x\mu_b - \mu_b^2}{\sigma^2} + C$$

$$= \frac{x(\mu_s-\mu_b)}{\sigma^2} + C'$$

- Generalization for multidimensional Gaussian distributions

$$\log\lambda(x) = \frac{x(\mu_s-\mu_b)}{\sigma^2} + C' \xrightarrow{\sigma^2\to V} \lambda(x) = \vec{x}(\vec{\mu}_s - \vec{\mu}_b)V^{-1} + C'$$

- Note that since we took -log of λ, F(x) is not signal probability, but we can trivially recover this

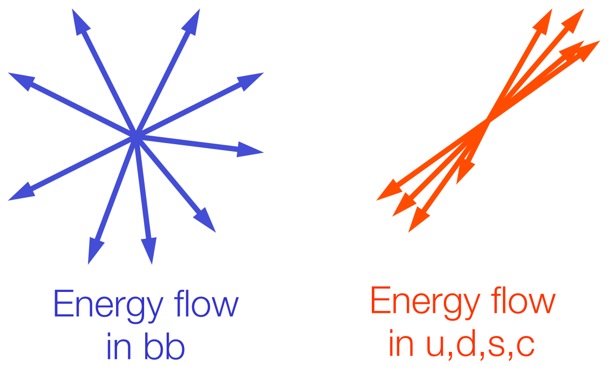
$$P_s(F) = \frac{1}{1+e^{-F}}$$

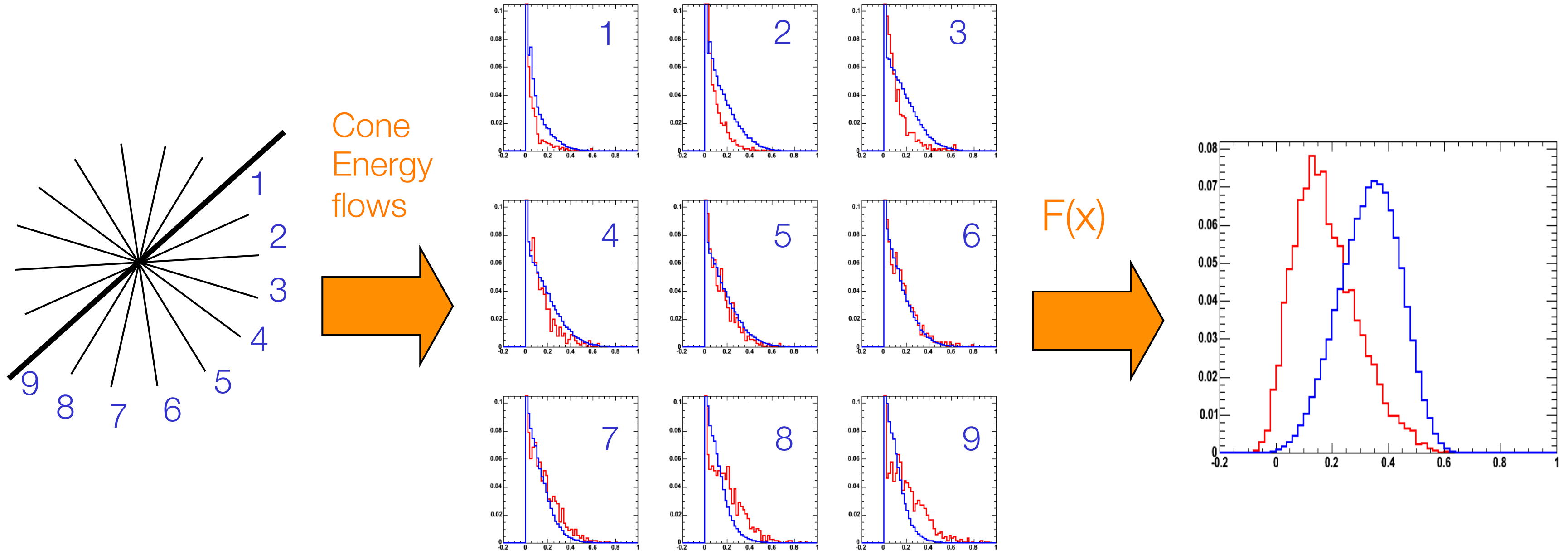
If λ=1, x is equally likely under s,b
Then F = -log(λ)=0 → P = 50%

“Logistic sigmoid function”

Example of Fisher discriminant use in HEP

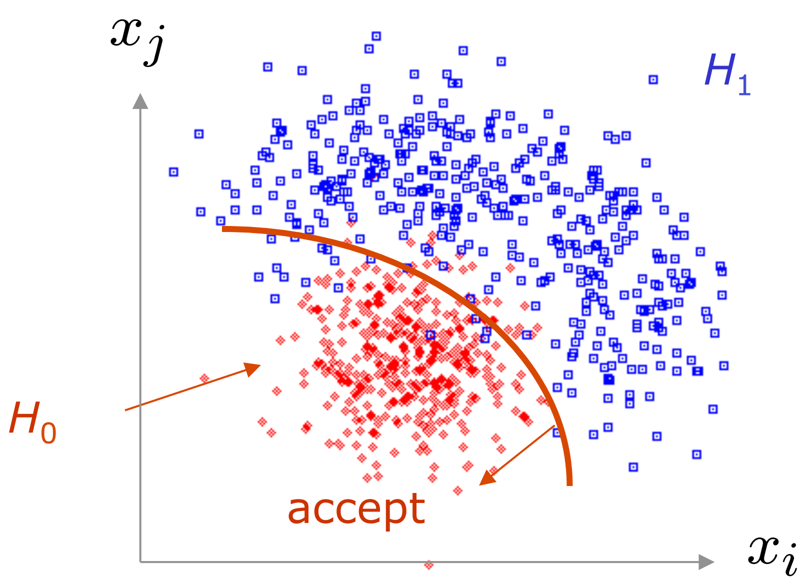
- The “CLEO” Fisher discriminant
  - Goal: distinguish between $e+e- \rightarrow$ Y4s $\rightarrow \bar{b}b$ and $\overline{uu}, \overline{dd}, \overline{ss}, \overline{cc}$
  - Method: Measure energy flow in 9 concentric cones around direction of B candidate





Non-linear test statistics

- In most real-life HEP applications signal and background are not multi-variate Gaussian distributions with different means
- Will need more complex Ansatz shapes than Fisher discriminant
- Loose ability analytically calculate parameters of Ansatz model from Likelihood Ratio test statistic (as was done for Fisher)



- Choose an Ansatz shapes with tunable parameters
  - Artificial Neural Networks
  - Decision Trees
  - Support Vector Machines
  - Rule Ensembles
- Need numeric procedure to estimate Ansatz parameters → Machine learning or Bayesian Learning

Machine Learning – General Principles

- Given a Ansatz parametric test statistic T(x|θ), quantify ‘risk’ due ‘loss of performance’ due to misclassifications by T as follows

Loss function (~ log of Gaussian Likelihood)

$$\underset{\text{Risk function}}{R(\theta)} = \int \overbrace{\left(T(\vec{x}\,|\,\theta) - \underset{\text{Target value of T for background classification}}{0}\right)^2}^{} f(\vec{x}\,|\,b)\,d\vec{x} \quad + \quad \int \overbrace{\left(T(\vec{x}\,|\,\theta) - \underset{\text{Target value of T for signal classification}}{1}\right)^2}^{} f(\vec{x}\,|\,s)\,d\vec{x}$$

- Practical issue: *since f(x|s,b) not analytically available, cannot evaluate risk function*. Solution → Substitute risk with ‘empirical risk’ which substitutes integral with Monte Carlo approximation

$$\underset{\text{Empirical Risk function}}{E(\theta)} = \frac{1}{N_b} \sum_{D(x|b)} \left(T(\vec{x}_i\,|\,\theta) - 0\right)^2 \quad + \quad \frac{1}{N_s} \sum_{D(x|s)} \left(T(\vec{x}_i\,|\,\theta) - 1\right)^2$$

x_i is a set of points sampled from f(x|b)

x_i is a set of points sampled from f(x|s)

Machine Learning – General Principles

- Minimization of empirical risk E(θ) can be performed with numerical methods (many tools are available, e.g. TMVA)
- But approximation of empirical risk w.r.t analytical risk introduces possibility for ‘overtraining’:

  If MC samples for signal and background are small, and number of parameters θ, one can always reduce empirical risk to zero (‘perfect selection’)

  (Conceptually similar to χ^2 fit : if you fit a 10th order polynomial to 10 points – you will always perfectly describe the data. You will however not perfectly describe an independent dataset sampled from the same parent distribution)
- Even if empirical risk is not reduced to zero by training, it may still be smaller than true risk → Control effect by evaluating empirical risk also on independent validation sample during minimization. If ER on samples start to diverge, stop minimization

Bayesian Learning – General principles

- Can also applied Bayesian methodology to learning process of decision boundaries
- Given a dataset D(x,y) and a Ansatz model with parameters w, aim is to estimate parameters w

P(w) = posterior density on parameters of discriminant

Likelihood of the data under hypothesis w

$$P(w|\vec{x},y) = \frac{L(\vec{x},y|w)P(w)}{P(\vec{x},y)}$$

$$= \frac{\boxed{L(y|w,\vec{x})L(x|w)}P(w)}{\int L(y|w,\vec{x})dwL(\vec{x})}$$

L(a,b)=L(a|b)L(b)

$$= \frac{\boxed{L(y|w,\vec{x})}P(w)}{\int L(y|w,\vec{x})dwL(\vec{x})}$$

L(x|w)=1 since input observables independent of model

Training data
x: inputs
y: class label
(S/B) typically

Bayesian Learning – General principles

- Inserting a binomial likelihood function to model classification the classification problem

$$L(y \mid x,w) = \prod_i T(x_i,w)^y \left[1 - T(x_i,w)\right]^{1-y}$$

$$\Downarrow$$

- The parameters w are thus estimated from the Bayesian posteriors densities

$$\boxed{P(w \mid \vec{x}, y) = \frac{L(y \mid w,\vec{x})P(w)}{\int L(y \mid w,\vec{x})\,dwL(\vec{x})}}$$

  - No iterative minimization, but Note that integrals over ‘w-space’ can usually only be performed numerically and if w contains many parameters, this is computationally challenging

- If class of function T(x,w) is large enough it will contain a function T(x,w*) that represents the true minimum in E(w)
  - I.e. T(x,w*) is the Bayesian equivalent of of Frequentist TS that is NP L ratio
  - In that case the test statistic is

$$L(y \mid x,w) = \prod_i T(x_i,w)^y \left[1 - T(x_i,w)\right]^{1-y}$$

$$\Downarrow$$

With y=0,1 only

$$T(x,w^*) = \int yL(y \mid x)\,dy$$

$$= L(y=1 \mid x) = \frac{L(x \mid y=1)P(y=1)}{L(x \mid y=0)P(y=0) + L(x \mid y=1)P(y=1)}$$

Machine/Bayesian learning – Non-linear Ansatz functions

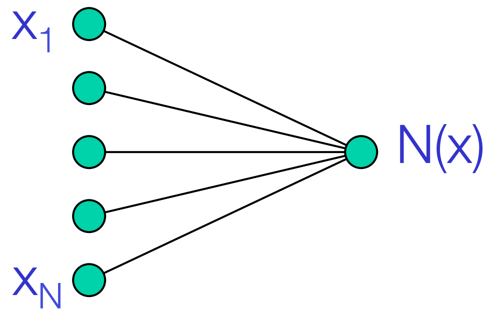
- Artificial Neural Network is one of the most popular non-linear ansatz forms. In it simplest incarnation the classifier function is

s(t) is the activation function, usually a logistic sigmoid

$$N(\vec{x}) = s\left(a_0 + \sum_i a_i x_i\right)$$

$$s(t) = \frac{1}{1+e^{-t}}$$

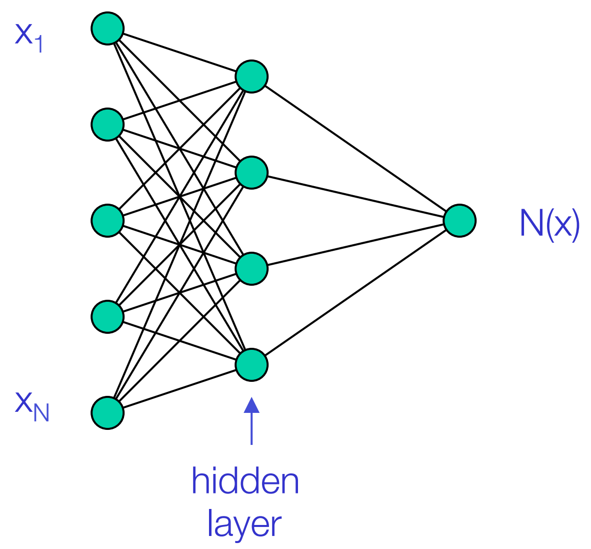
- This formula corresponds to the 'single layer perceptron'
  - Visualization of single layer network topology



Since the activation function s(t) is monotonic, a single layer N(x) is equivalent to the Fisher discriminant F(x)

Neural networks – general structure

- The single layer model and easily be generalized to a *multilayer* perceptron



$$N(\vec{x}) = s(a_0 + \sum_{i=1,}^{m} a_i h_i(\vec{x}))$$

with $$h_i(\vec{x}) = s(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j)$$

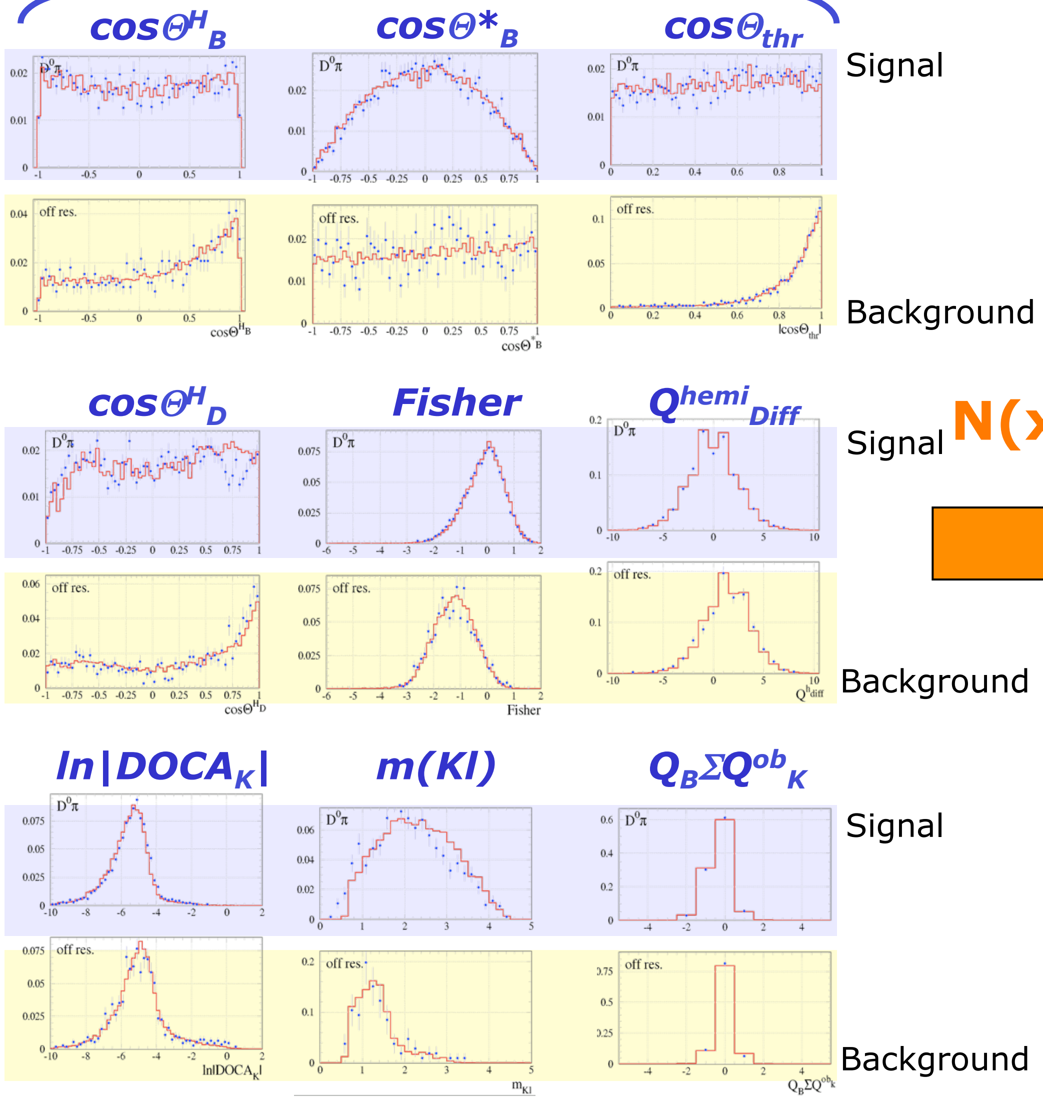
with a_i and w_{ij} weights (connection strengths)

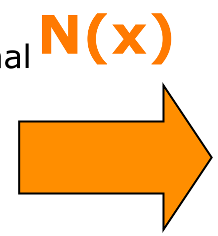
- Easy to generalize to arbitrary number of layers
- Feed-forward net: values of a node depend only on earlier layers (usually only on preceding layer) ‘the network architecture’
- More nodes bring N(x) allow it to be closer to optimal (Neyman Pearson / Bayesian posterior) but with much more parameters to be determined

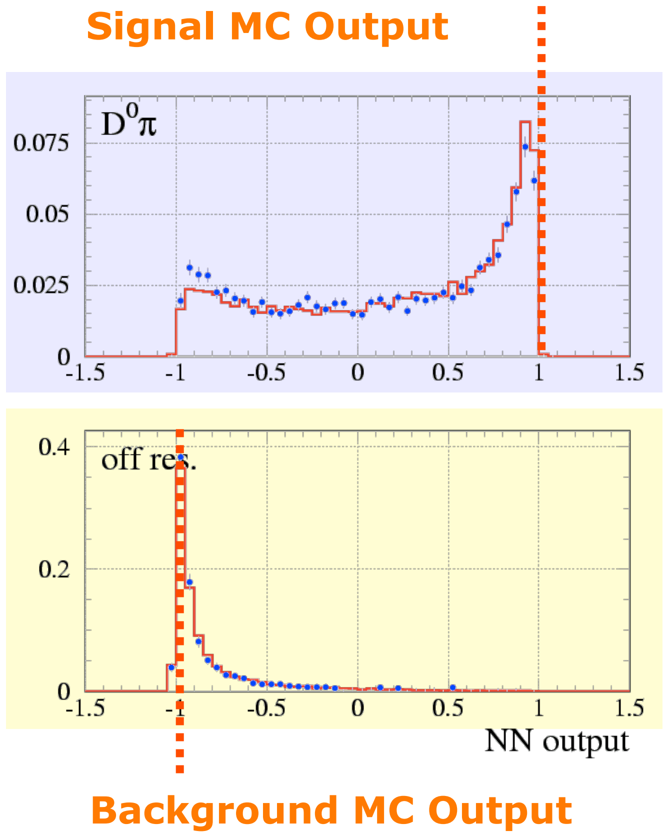
Neural networks – training example

Input Variables (9)

Output Variables (1)







Wouter Verkerke, UCSB

Practical aspects of machine learning

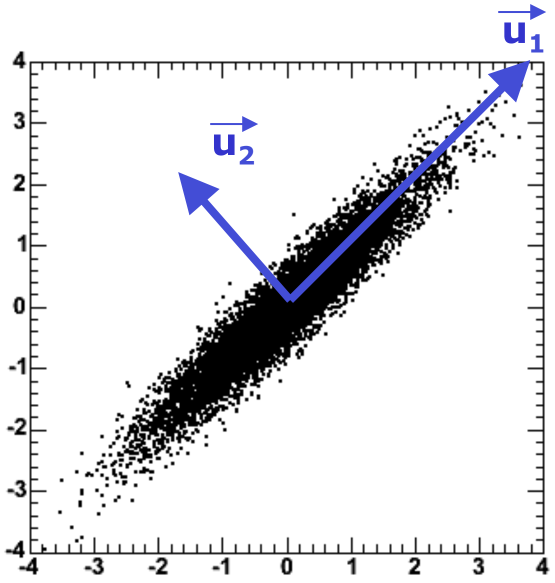
- Choose input variables sensibly
  - Don't include badly understood observables (such as #tracks/evt), variables that are not expected carry useful information
  - Generally: "Garbage in = Garbage out"
- Traditional Machine learning provides no guidance of useful complexity of test statistic (e.g. NN topology, layers)
  - Usually better to start simple and gradually increase complexity and see how that pays off
- Bayesian learning can (in principle) provide guidance on model complexity through Bayesian model selection
  - Bayes factors automatically includes a penalty for including too much model structure.

$$K = \frac{P(D|H_1)}{P(D|H_2)} = \frac{\int L(D|\theta_1,H_1)P(\theta_2|H_1)d\theta_2}{\int L(D|\theta_2,H_2)P(\theta_2|H_2)d\theta_2}$$

  - But availability of Bayesian model selection depends in practice on the software that you use.

Practical aspects of machine learning

- Don't make the learning problem unnecessarily difficult for the machine
- E.g. remove strong correlation with **explicit decorrelation** before learning step
  - Can use Principle Component Analysis
  - Or Cholesky decomposition (rotate with square-root of covariance matrix)



- Also: remember that for 2-class problem (sig/bkg) that each have multivariate Gaussian distributions with different means, the optimal discriminant is known analytically
  - Fisher discriminant is analytical solution. NN solution reduces to single-layer perceptron
- Thus, you can help your machine by transforming your inputs in a form **as close as possible to the Gaussian form** by transforming your input observables

Gaussianization of input observables

- You can transform *any* distribution in a Gaussian distribution in two steps
- 1 – Probability integral transform

"…seems likely to be one of the most fruitful conceptions introduced into statistical theory in the last few years" −Egon Pearson (1938)

$$y(x) = \int_{-\infty}^{x} f(x'|H)dx'$$

turns any distribution f(x) into a flat distribution in y(x)

- 2 – Inverse error function

$$x^{\text{Gauss}} = \sqrt{2} \cdot \text{erf}^{-1}\left(2x^{\text{flat}} - 1\right)$$

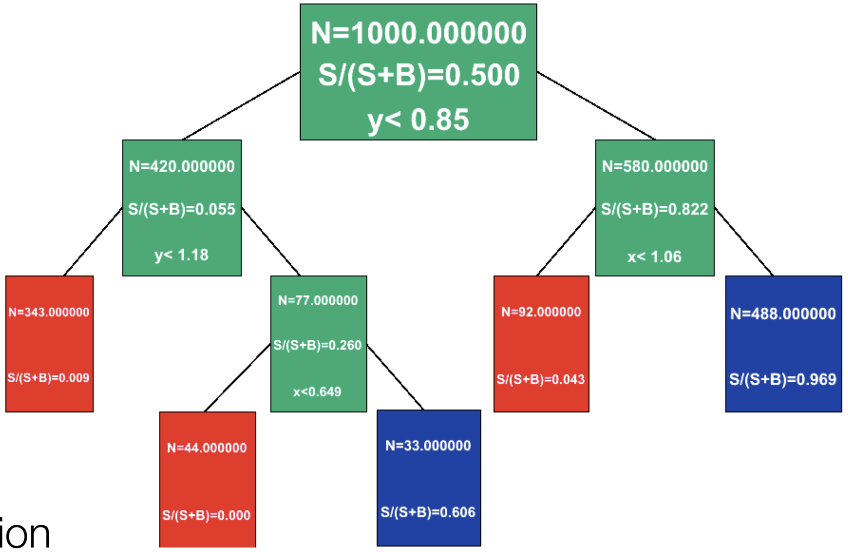
$$\text{erf}\left(x\right) = \frac{2}{\sqrt{\pi}} \int_0^{x} e^{-t^2}\,dt$$

turns flat distribution into a Gaussian distribution

- Note that you can make either signal or background Gaussian, but usually not *both*

A very different type of Ansatz - Decision Trees

- A Decision Tree encodes sequential rectangular cuts
  - But with a lot of underlying theory on training and optimization
  - Machine-learning technique, widely used in social sciences
  - L. Breiman et al., “Classification and Regression Trees” (1984)



- Basic principle
  - Extend cut-based selection
  - Try not to rule out events failing a particular criterion
  - Keep events rejected by one criterion and see whether other criteria could help classify them properly