Lecture 2

Extended likelihood function

- Given a sample of N measurements of the variables $(x_1, \ldots, x_n)$, the likelihood function expresses the probability density of the sample, as a function of the unknown parameters:

$$L = \prod_{i=1}^{N} f(x_1^i, \cdots, x_n^i; \theta_1, \cdots, \theta_m)$$

- If the size N of the sample is also a random variable, the extended likelihood function is usually also used:

$$L = P(N; \theta_1, \cdots, \theta_m) \prod_{i=1}^{N} f(x_1^i, \cdots, x_n^i; \theta_1, \cdots, \theta_m)$$

- Where $P(N; \theta_1, \ldots, \theta_m)$ is in practice always a Poisson distribution whose expected rate is a function of the unknown parameters

- In many cases it is convenient to use $-\ln L$ or $-2\ln L$: $\prod_i \to \sum_i$

Extended likelihood function

- For Poissonian signal and background processes:

$$L(x_i; s, b, \theta) = \frac{(s+b)^n e^{-(s+b)}}{n!} \prod_{i=1}^{n} (f_s P_s(x_i;\theta) + f_b P_b(x_i;\theta))$$

$$f_s = \frac{s}{s+b}, \quad f_b = \frac{b}{s+b} \quad \Rightarrow \quad = \frac{e^{-(s+b)}}{n!} \prod_{i=1}^{n} (s P_s(x_i;\theta) + b P_b(x_i;\theta))$$

- We can fit simultaneously s, b and θ minimizing: constant!

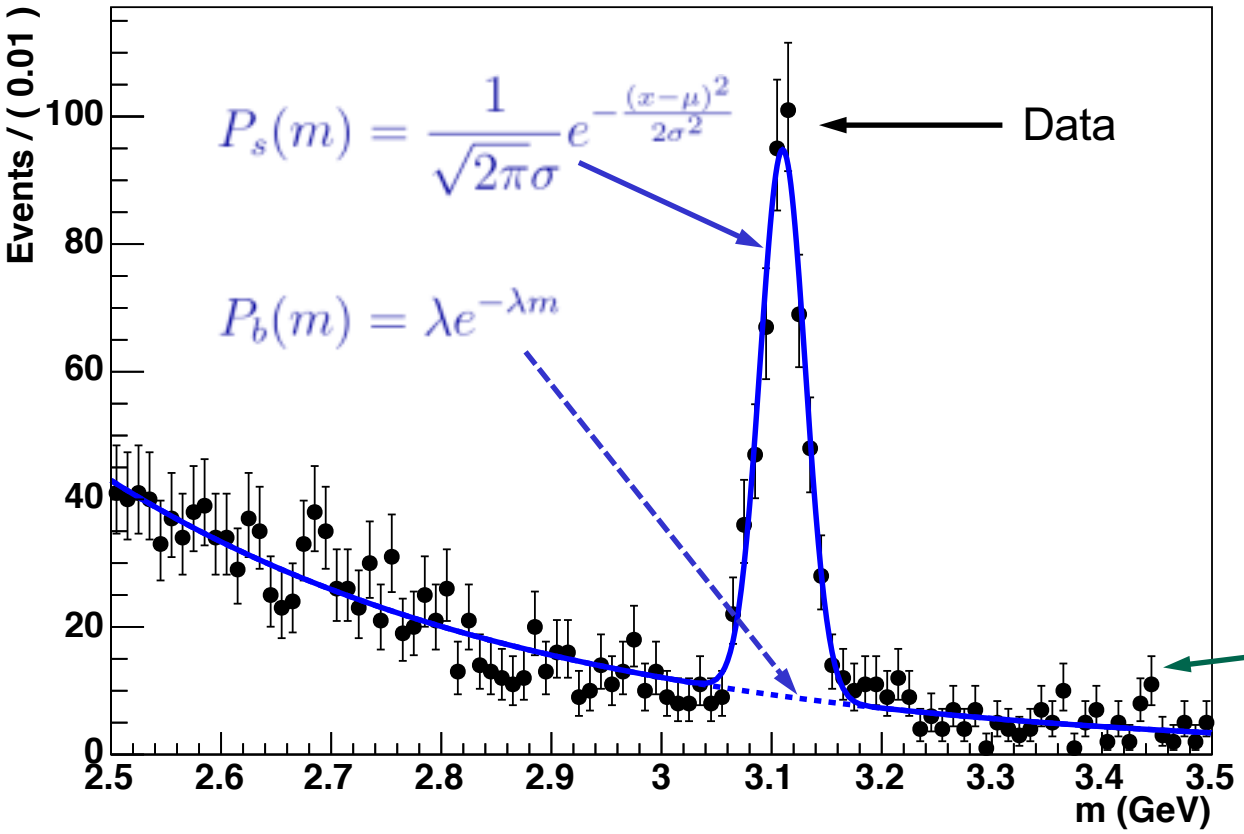
$$-\ln L = s + b - \sum_{i=1}^{n} \ln(s P_s(x_i;\theta) + b P_b(x_i;\theta)) + \ln n!$$

- Sometimes s is replaced by $\mu\, s_0$, where s_0 is the theory estimate and μ is called signal strength

Example of ML fit

- $P_s(m)$: Gaussian peak
- $P_b(m)$: exponential shape

Exponential decay parameter λ, Guassian mean μ and standard deviation σ can be fit together with sig. and bkg. yields s and b.



The additional parameters, beyond the parameters of interest (s in this case), used to model background, resolution, etc. are examples of nuisance parameters

In the plot, data are accumulated into **bins** of a given width

Error bars usually represent uncertainty on each bin count (in this case: Poissonian)

Gaussian case

- If we have n independent measurements all modeled with (or approximated to) the same Gaussian PDF, we have:

$$-2\ln L = \underbrace{\sum_{i=1}^{n} \frac{(x_i-\mu)^2}{\sigma^2}} + n(\ln 2\pi + 2\ln\sigma)$$

(example of a χ^2 variable)

- An analytical minimization of $-2\ln L$ w.r.t μ (assuming σ^2 is known) gives the arithmetic mean as ML estimate of μ:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- If σ^2 is also unknown, the ML estimate of σ^2 is:

$$\hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \hat{\mu})^2$$

- The above estimate can be demonstrated to have an unpleasant feature, called *bias* (→ next slide)

Some estimator properties

- **Consistency**: for large number of measurements the estimator $\hat{\theta}$ should converge, in probability, to the true value θ.
  - ML estimators are consistent

- **Bias**: the bias of a parameter is the average value of its deviation from the true value

$$b(\theta) = \left\langle \hat{\theta} - \theta \right\rangle = \left\langle \hat{\theta} \right\rangle - \theta$$

  - ML estimators may have a bias, but the bias decreases with large number of measurements (if the fit model is correct…!)
  - E.g.: in the case of the estimate of a Gaussian's σ^2, the unbiased estimate is the well known:

$$\hat{\sigma^2}_{\text{unbias.}} = \frac{n}{n-1}\hat{\sigma^2} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

ML method underestimates the variance σ^2

Efficiency of an estimator

- The variance of any consistent estimator is subject a lower bound (Cramér-Rao bound):

$$\mathrm{Var}[\hat{\theta}] \geq \frac{\left(1 + \frac{\partial b(\theta)}{\partial \theta}\right)^2}{\left\langle \left(\frac{\partial \ln L(x_1, \cdots, x_n; \theta)}{\partial \theta} \right)^2 \right\rangle} = V_{\mathrm{CR}}$$

bias of θ

Fisher information

- **Efficiency** can be defined as the ratio of Cramér-Rao bound and the estimator's variance:

$$\varepsilon(\hat{\theta}) = \frac{V_{\mathrm{CR}}}{\mathrm{Var}[\hat{\theta}]}$$

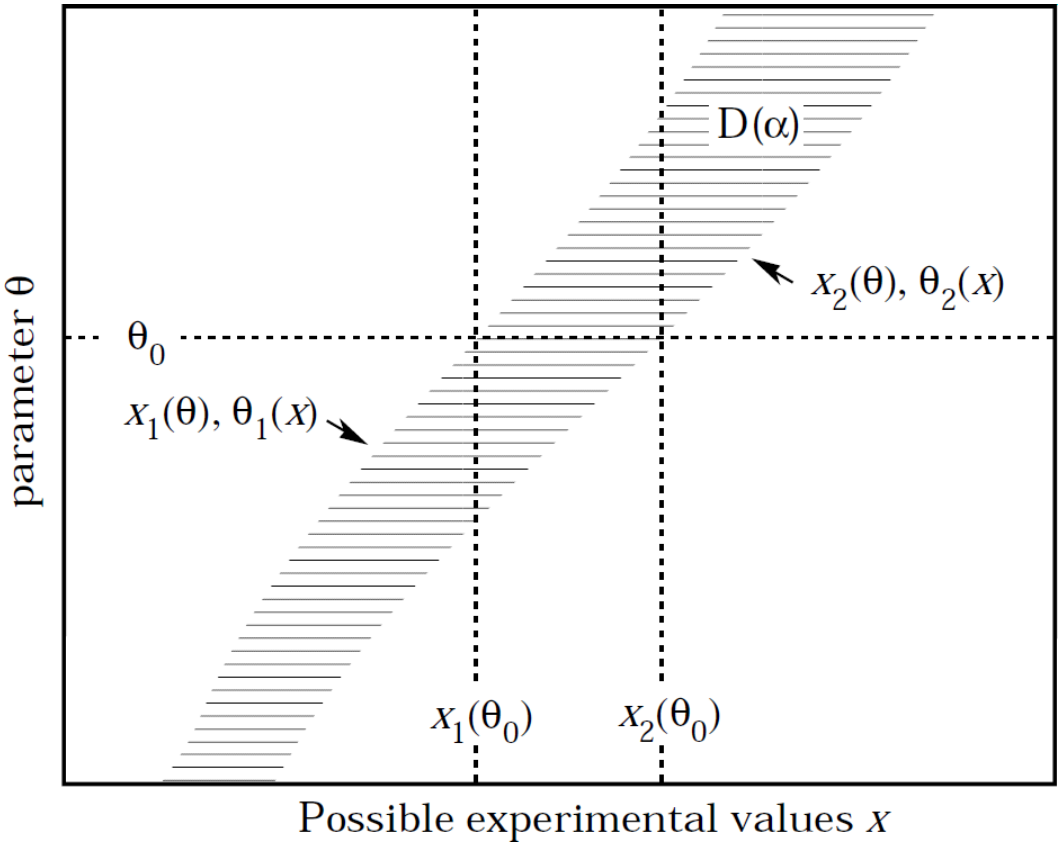
  - Efficiency for ML estimators tends to 1 for large number of measurements
  - I.e.: ML estimates have, asymptotically, the smallest possible variance

Neyman's confidence intervals

Procedure to determine frequentist confidence intervals

- Scan the allowed range of an unknown parameter θ
- Given a value of θ compute the interval $[x_1, x_2]$ that contain x with a probability $1 - \alpha$ equal to 68% (or 90%, 95%)
- **Choice of interval needed!**
- Invert the confidence belt: for an observed value of x, find the interval $[\theta_1, \theta_2]$
- A fraction of the experiments equal to $1 - \alpha$ will measure x such that the corresponding $[\theta_1, \theta_2]$ contains ("covers") the true value of θ ("coverage")
- **Note**: the random variables are $[\theta_1, \theta_2]$, not θ !

Plot from PDG statistics review

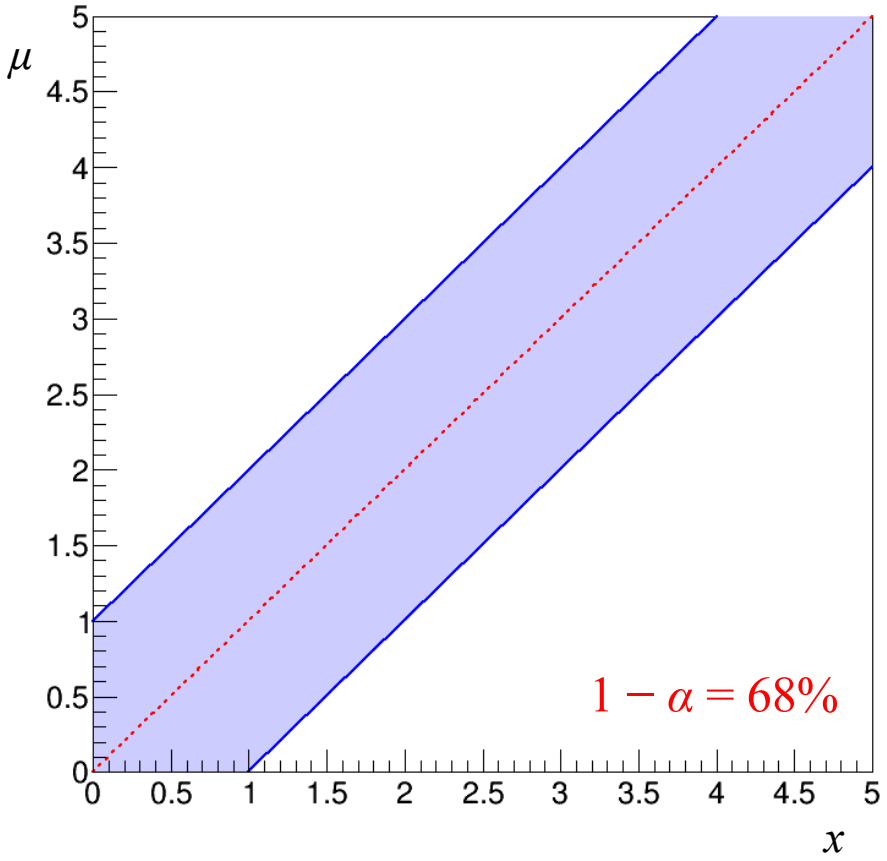


α = significance level

Simplest example: Gaussian case

- Assume a Gaussian distribution with unknown average μ and known $\sigma = 1$
- The belt inversion is trivial and gives the expected result:
  Central value $\hat{\mu} = x$,
  $[\mu_1, \mu_2] = [x - \sigma, x + \sigma]$
- So we can quote:

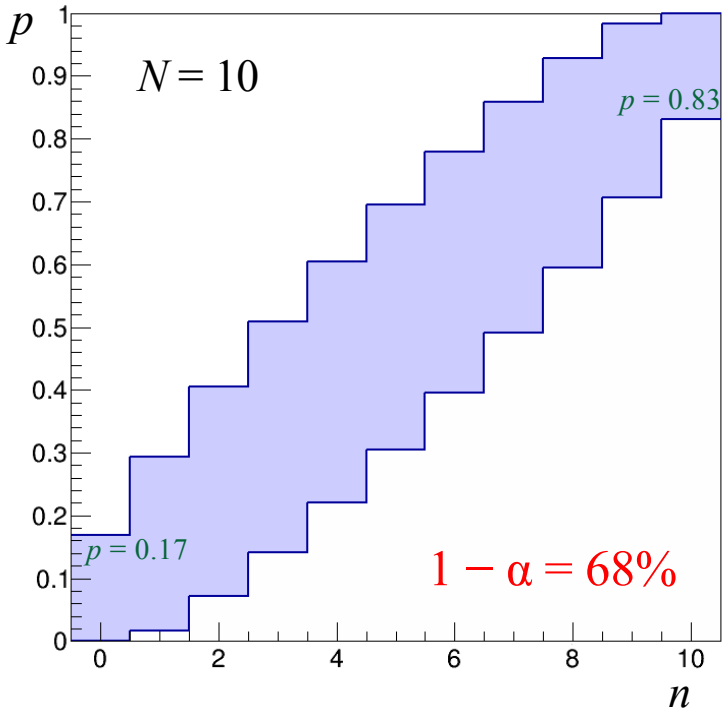
$$\mu = x \pm \sigma$$



Binomial intervals

- The Neyman's belt construction may only guarantee approximate coverage in case of discrete variables
- For a Binomial distribution: find the interval $\{n_{\min}, \ldots, n_{\max}\}$ such that:

$$\sum_{n=n_{\min}}^{n=n_{\max}} \frac{N!}{n!(N-n)!} p^n (1-p)^{N-n} \geq 1-\alpha$$

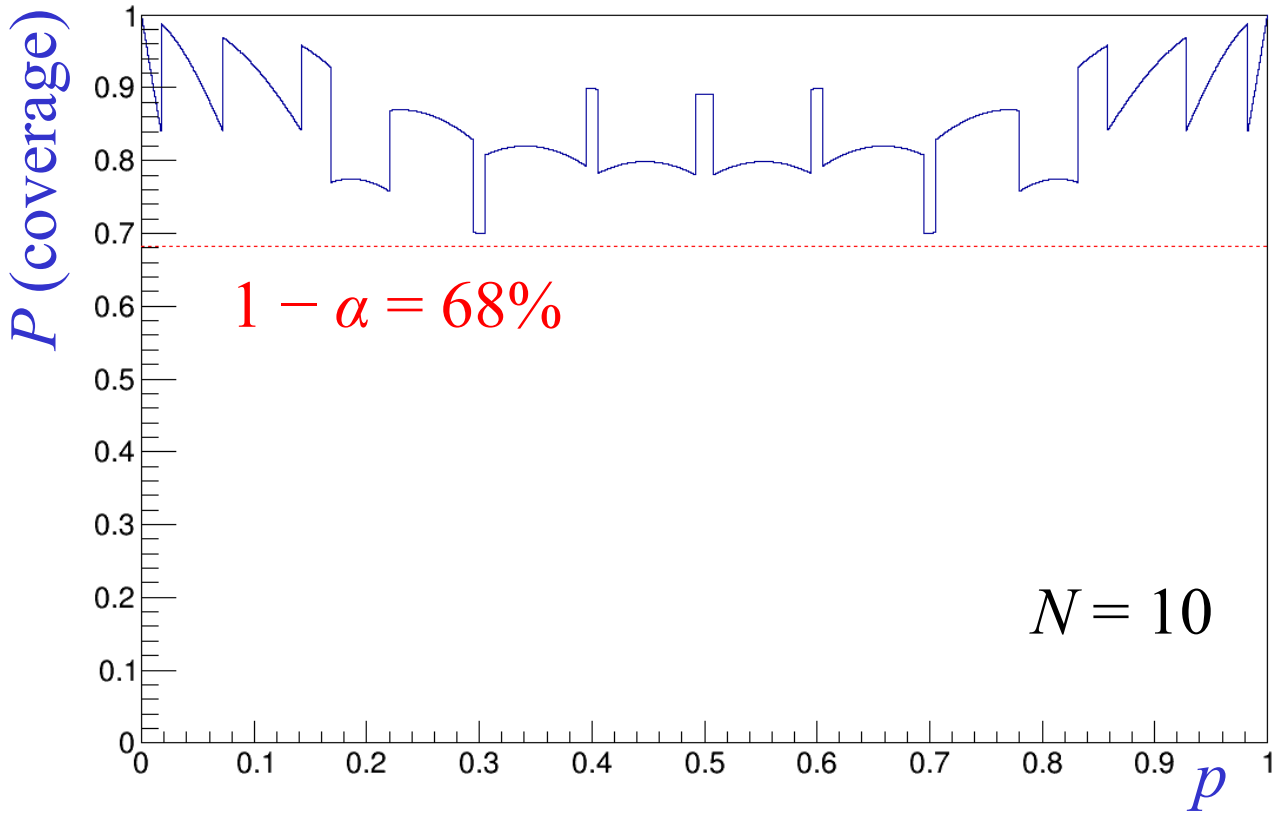


- Clopper and Pearson (1934) solved the belt inversion problem for central intervals
- For an observed $n = k$, find lowest p^{lo} and highest p^{up} such that:
- $P(n \leq k \mid N, p^{\text{lo}}) = \alpha/2, \;\; P(n \geq k \mid N, p^{\text{up}}) = \alpha/2$
- E.g.: $n = N = 10$, $P(N|N) = p^N = \alpha/2$, hence:
  $p^{\text{lo}} = \sqrt[10]{\alpha/2} = 0.83$ (68% CL), 0.74 (90% CL)
- A frequently used approximation, which fails for $n = 0$, N is:

$$\hat{p} = \frac{n}{N}, \; \sigma_{\hat{p}} \simeq \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

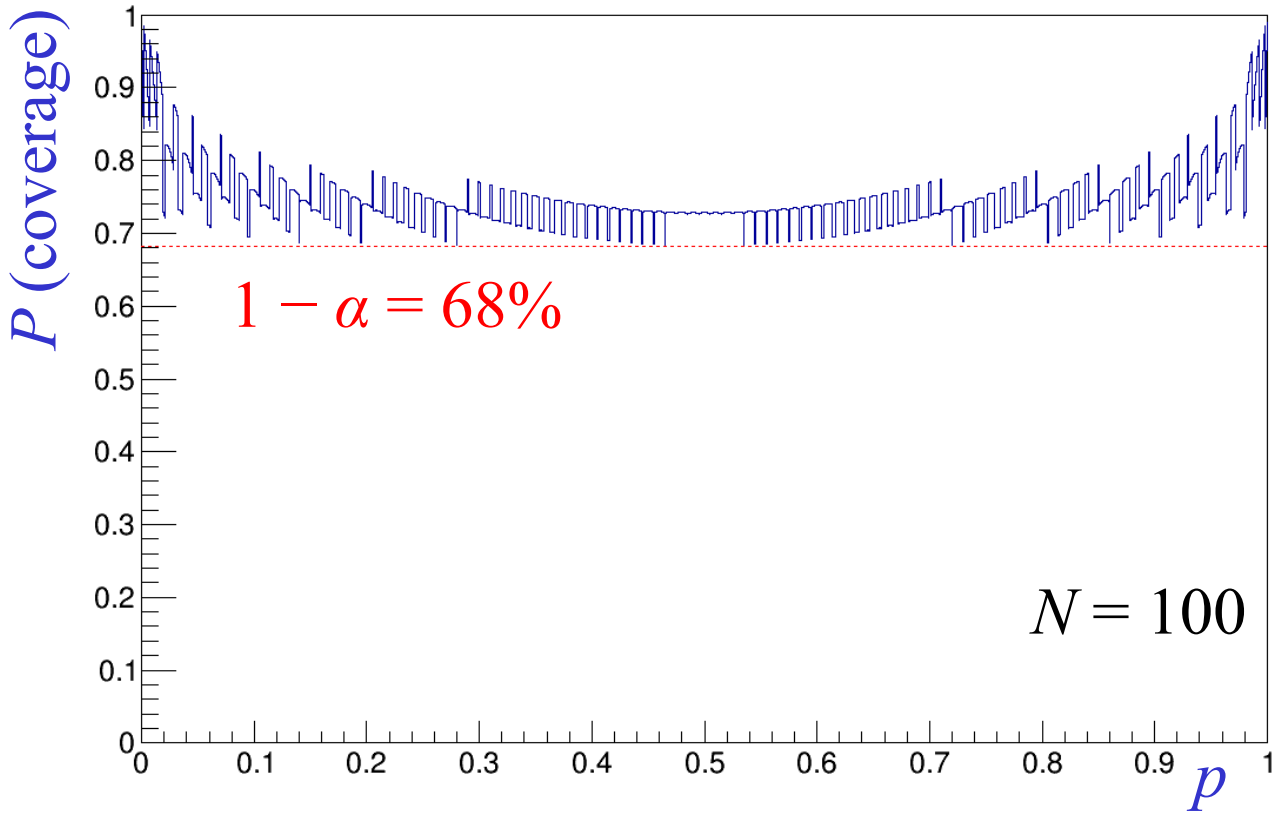
Clopper-Pearson coverage (I)

- CP intervals are often defined as “exact” in literature
- Exact coverage is often impossible to achieve for discrete variables



Clopper-Pearson coverage (II)

- For larger N the "ripple" gets closer to the nominal 68% coverage



Approx. maximum likelihood errors

- A parabolic approximation of $-2\ln L$ around the minimum is equivalent to a Gaussian approximation
  - Sufficiently accurate in many but not all cases

$$-2\ln L = \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{\sigma^2} + \text{const.}$$

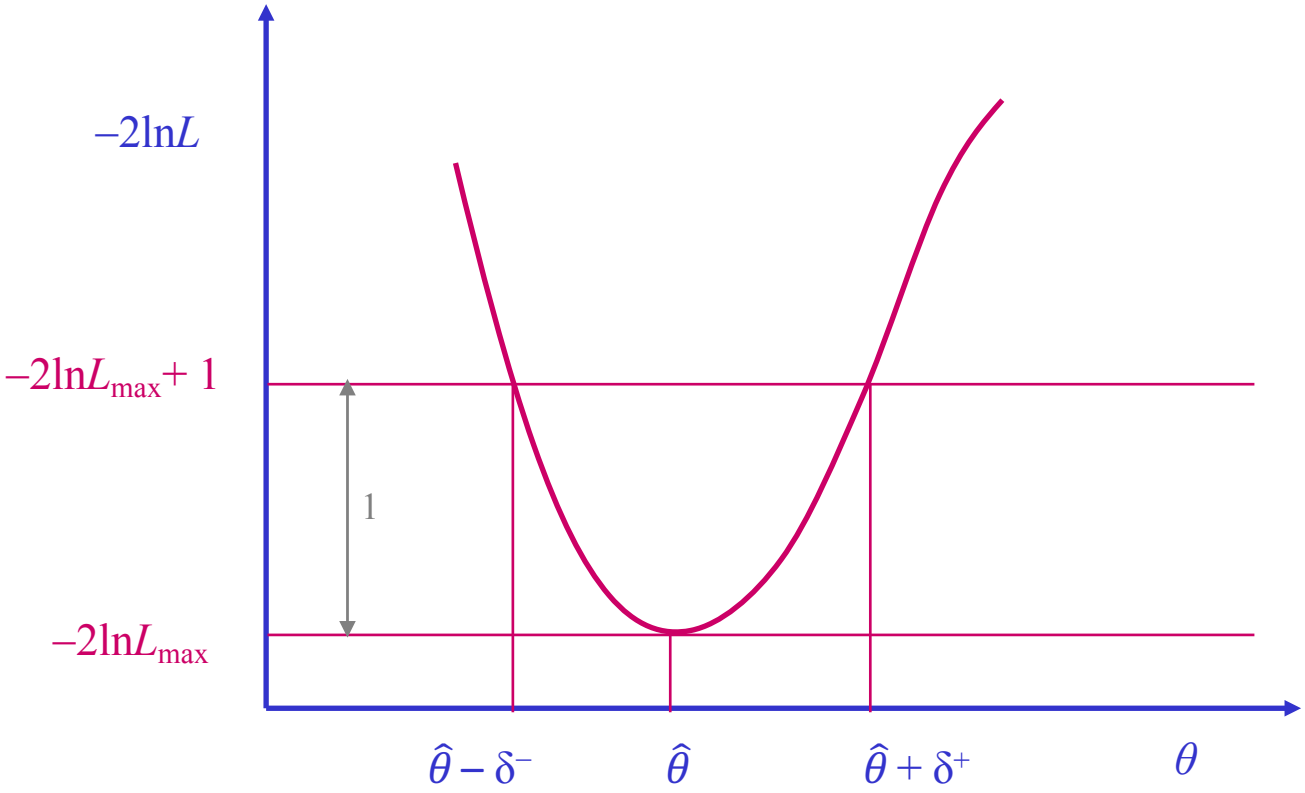
- Estimate of the covariance matrix from 2nd order partial derivatives w.r.t. fit parameters at the minimum:

$$V_{ij}^{-1} = -\left.\frac{\partial^2 \ln L}{\partial\theta_i \partial\theta_j}\right|_{\theta_k=\hat{\theta}_k}$$

- Implemented in Minuit as MIGRAD/HESSE function

Asymmetric errors

- Another approximation alternative to the parabolic one may be to evaluate the excursion range of $-2\ln L$.
- Error ($n\sigma$) determined by the range around the maximum for which $-2\ln L$ increases by $+1$ ($+n^2$ for $n\sigma$ intervals)



- Errors can be asymmetric
- For a Gaussian PDF the result is identical to the 2nd order derivative matrix
- Implemented in Minuit as MINOS function

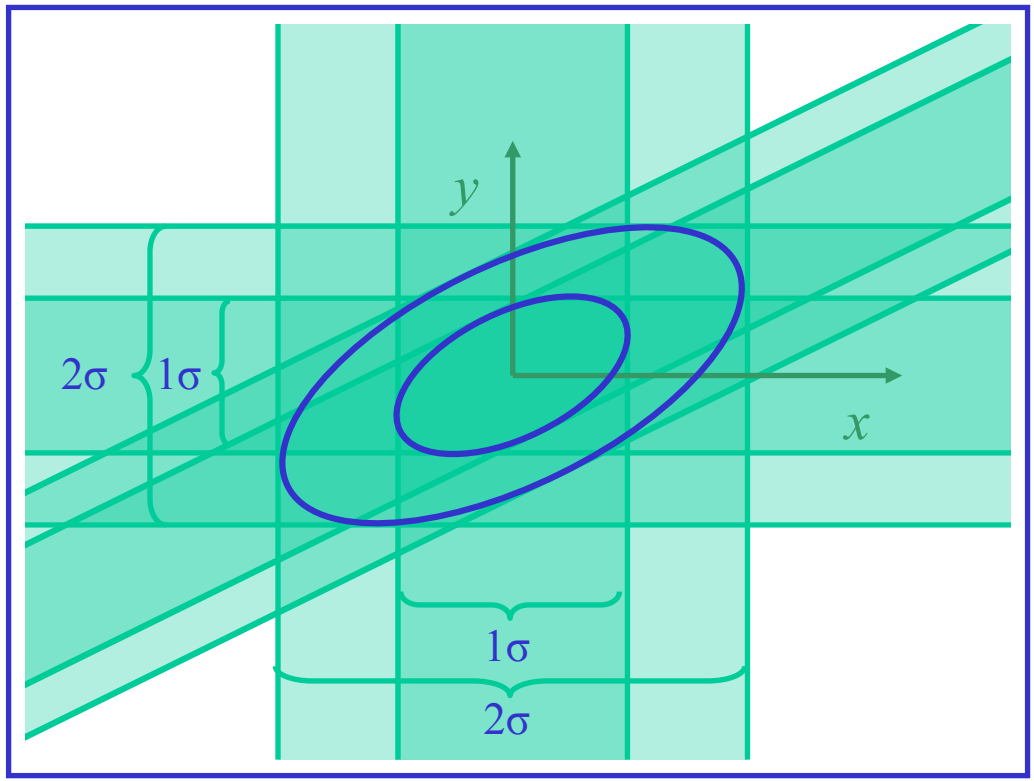
2D intervals

- In more dimensions one can determine 1σ and 2σ contours
- Note: different probability content in 2D compared to one dimension
- 68% and 95% contours are usually preferable

$$P_{1D}(n\sigma) = \sqrt{\frac{2}{\pi}} \int_0^n e^{-\frac{x^2}{2}} \mathrm{d}x = \mathrm{erf}\left(\frac{n}{\sqrt{2}}\right)$$

$$P_{2D}(n\sigma) = \int_0^n e^{-\frac{r^2}{2}} r \mathrm{d}r = 1 - e^{-\frac{n^2}{2}}$$

Width	P_{1D}	P_{2D}
1σ	0.6827	0.3934
2σ	0.9545	0.8647
3σ	0.9973	0.9889
1.515σ		0.6827
2.486σ		0.9545
3.439σ		0.9973

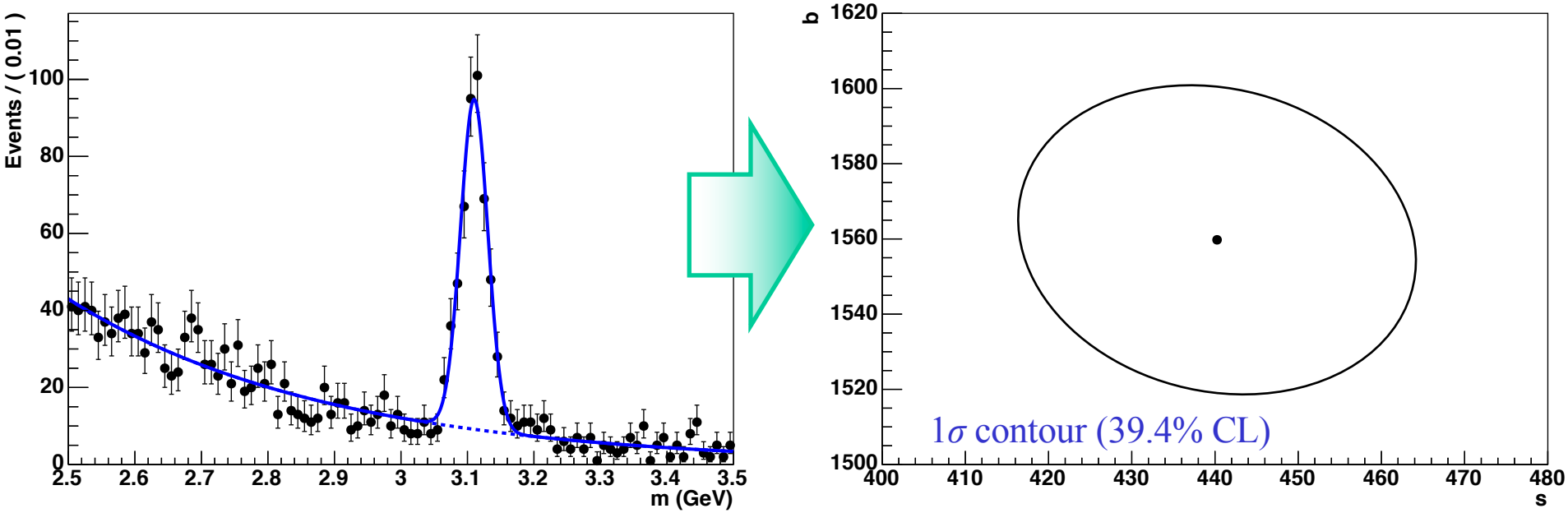


Example of 2D contour

- From previous fit example:
  - $P_s(m)$: Gaussian peak
  - $P_b(m)$: exponential shape

Exponential decay parameter, Gaussian mean and standard deviation are fit together with s and b yields.

The contour shows for this case a mild correlation between s and b



Error propagation

- Assume we estimate from a fit the parameter set: $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ and we know their covariance matrix Θ_{ij}
- We want to determine a new set of parameters that are functions of $\boldsymbol{\theta}$: $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_m)$.
- For small uncertainties, a linear approximation maybe sufficient
- A Taylor expansion around the central values of $\boldsymbol{\theta}$ gives, using the error matrix Θ_{ij}:

$$H_{ij} = \sum_{k,l} \frac{\partial \eta_i}{\partial \theta_k} \frac{\partial \eta_j}{\partial \theta_l} \Theta_{kl}$$

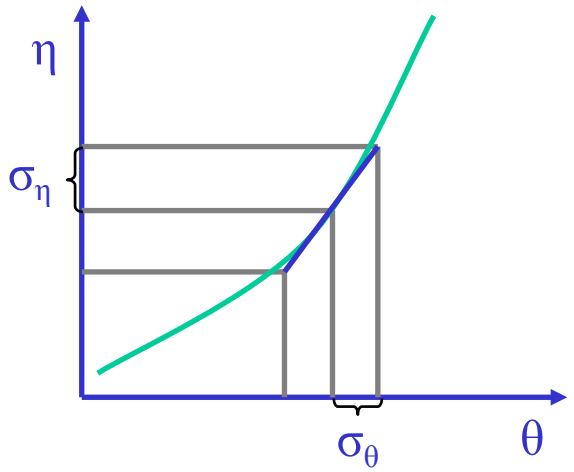
- Few examples in case of no correlation:

$$\sigma_{x+y} = \sigma_{x-y} = \sqrt{\sigma_x^2 + \sigma_y^2}$$

$$\frac{\sigma_{xy}}{xy} = \frac{\sigma_{x/y}}{x/y} = \sqrt{\left(\frac{\sigma_x}{x}\right)^2 + \left(\frac{\sigma_y}{y}\right)^2}$$

$$\sigma_{x^2} = 2x\sigma_x$$

$$\sigma_{\ln x} = \frac{\sigma_x}{\sqrt{x}}$$



Binned likelihood

- Sometimes data are available as binned histogram
  - Most often each bin obeys Poissonian statistics (event counting)
- The likelihood function is the product of Poisson PDFs corresponding to each bin having entries n_i
- The expected number of entries n_i depends on some unknown parameters: $\mu_i = \mu_i(\theta_1, \ldots, \theta_m)$
- The function to minimize is the following $-2 \ln L$:

$$-2 \ln L = -2 \ln \prod_{i=1}^{n_{\text{bins}}} \text{Poiss}(n_i; \mu_i(\theta_1, \cdots, \theta_m))$$

$$= -2 \ln \prod_{i=1}^{n_{\text{bins}}} \frac{e^{-\mu_i(\theta_1, \cdots, \theta_m)} \mu_i(\theta_1, \cdots, \theta_m)^{n_i}}{n_i!}$$

- The expected number of entries μ_i is often approximated by a continuous function $\mu(x)$ evaluated at the center x_i of the bin
- Alternatively, μ_i can be a combination of other histograms ("templates")
  - E.g.: sum of different simulated processes with floating yields as fit parameters

Binned fits: minimum χ^2

- Bin entries can be approximated by Gaussian variables for sufficiently large number of entries with standard deviation equal to n_i (Neyman's χ^2)
- Maximizing L is equivalent to minimize:

$$\chi^2 = \sum_{i=1}^{n_{\text{bins}}} \frac{(n_i - \mu(x_i; \theta_1, \cdots, \theta_m))^2}{n_i}$$

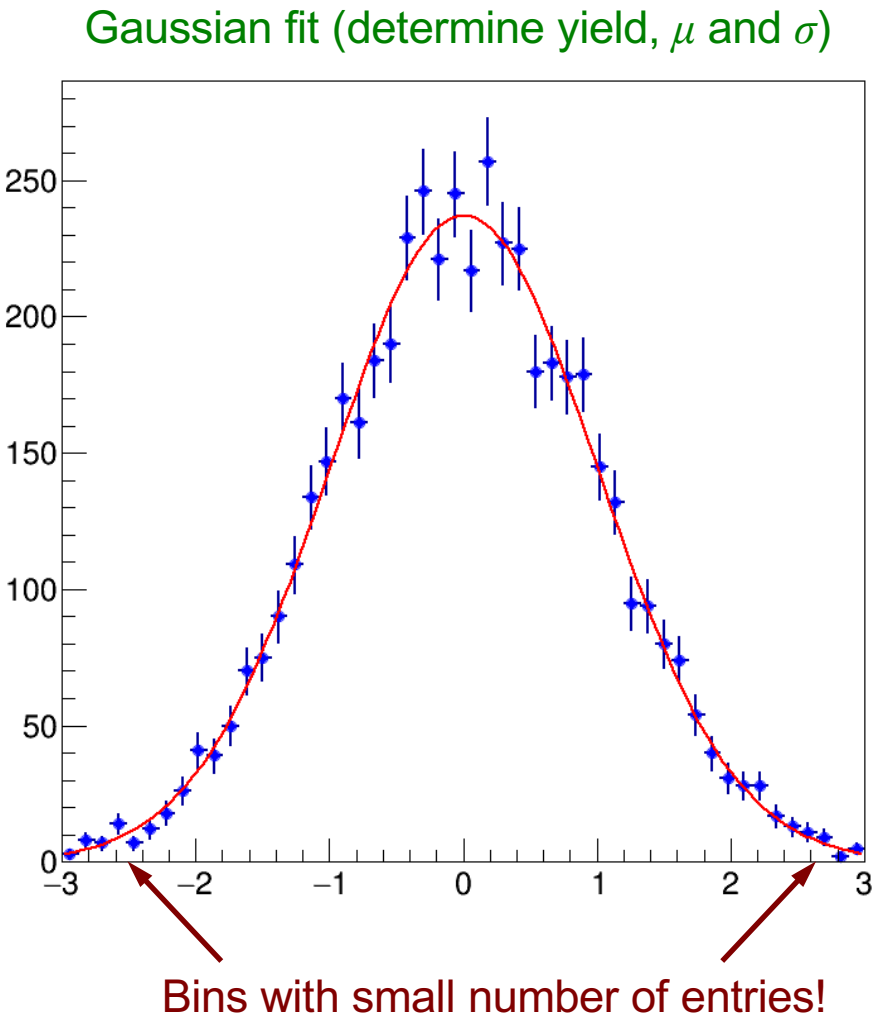
- Sometimes, the denominator n_i is replaced (Pearson's χ^2) by:

$$\mu_i = \mu\,(x_i;\, \theta_1,\, \ldots,\, \theta_m)$$

  in order to avoid cases with zero or small n_i
- Analytic solution exists for linear and other simple problems
  - E.g.: linear fit model
- Most of the cases are treated numerically, as for unbinned ML fits

Binned fit example

- Binned fits are convenient w.r.t. unbinned fits because the number of input variables decreases from the number of entries to the number of bins
  - Usually simpler and faster numerically
  - Unbinned fits become unpractical for very large number of entries
- A fraction of the information is lost, hence a possible loss of precision may occur for small number of entries
- Treat correctly bins with smalll number of entries!



Fit quality (χ^2 test)

- The maximum value of the likelihood function obtained from the fit doesn't usually give information about the goodness of the fit
- The χ^2 of a fit with a Gaussian underlying model is distributed according to a known PDF

$$P(\chi^2; n) = \frac{2^{-\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)} \chi^{n-2} e^{-\frac{\chi^2}{2}}$$

n is the number of degrees of freedom (n. of bins − n. of params.)

  - The cumulative distribution of $P(\chi^2; n)$ follows a uniform distribution between 0 and 1 (p-value)
  - If the model deviates from the assumed distribution, the distribution of the p-value will be more peaked around zero
- Note! p-values are not the "*probability of the fit hypothesis*"
  - This would be a Bayesian probability, with a different meaning, and should be computed in a different way

Binned likelihood ratio

- A better alternative to the (Gaussian-inspired, Neyman and Pearson's) χ^2 has been proposed by Baker and Cousins using the following likelihood ratio:

$$\chi^2_\lambda = -2\ln\prod_i \frac{L(n_i;\mu_i)}{L(n_i;n_i)} = -2\ln\prod_i \frac{e^{-\mu_i}\mu_i^{n_i}}{n_i!}\frac{n_i!}{e^{-n_i}n_i^{n_i}}$$

$$= 2\sum_i \left[\mu_i(\theta_i,\cdots,\theta_m) - n_i + n_i\ln\left(\frac{n_i}{\mu_i(\theta_1,\cdots,\theta_m)}\right)\right]$$

- Same minimum value as from Poisson likelihood function, since a constant term has been added to the log-likelihood function
- In addition, it provides goodness-of-fit information, and asymptotically obeys chi-squared distribution with $n - m$ degrees of freedom
  (Wilks' theorem, see following slides)

S. Baker, R. Cousins NIM 221 (1984) 437

Combining measurements

- Assume two measurements with different uncorrelated (Gaussian) errors: $m_1 \pm \sigma_1, \quad m_2 \pm \sigma_2$

- Build the χ^2: $\chi^2 = \frac{(m - m_1)^2}{\sigma_1^2} + \frac{(m - m_2)^2}{\sigma_2^2}$

- Minimize the χ^2: $0 = \frac{\partial \chi^2}{\partial m} = 2\frac{(m - m_1)}{\sigma_1^2} + 2\frac{(m - m_2)}{\sigma_2^2}$

- Estimate m as: $m = \frac{\frac{m_1}{\sigma_1^2} + \frac{m_2}{\sigma_2^2}}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} = \frac{w_1 m_1 + w_2 m_2}{w_1 + w_2}$ ← Weighted average, $w_i = \sigma_i^{-2}$

- Error estimate: $\frac{1}{\sigma_m^2} = -\frac{\partial^2 \ln L}{\partial m^2} = \frac{1}{2}\frac{\partial^2 \chi^2}{\partial m^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$

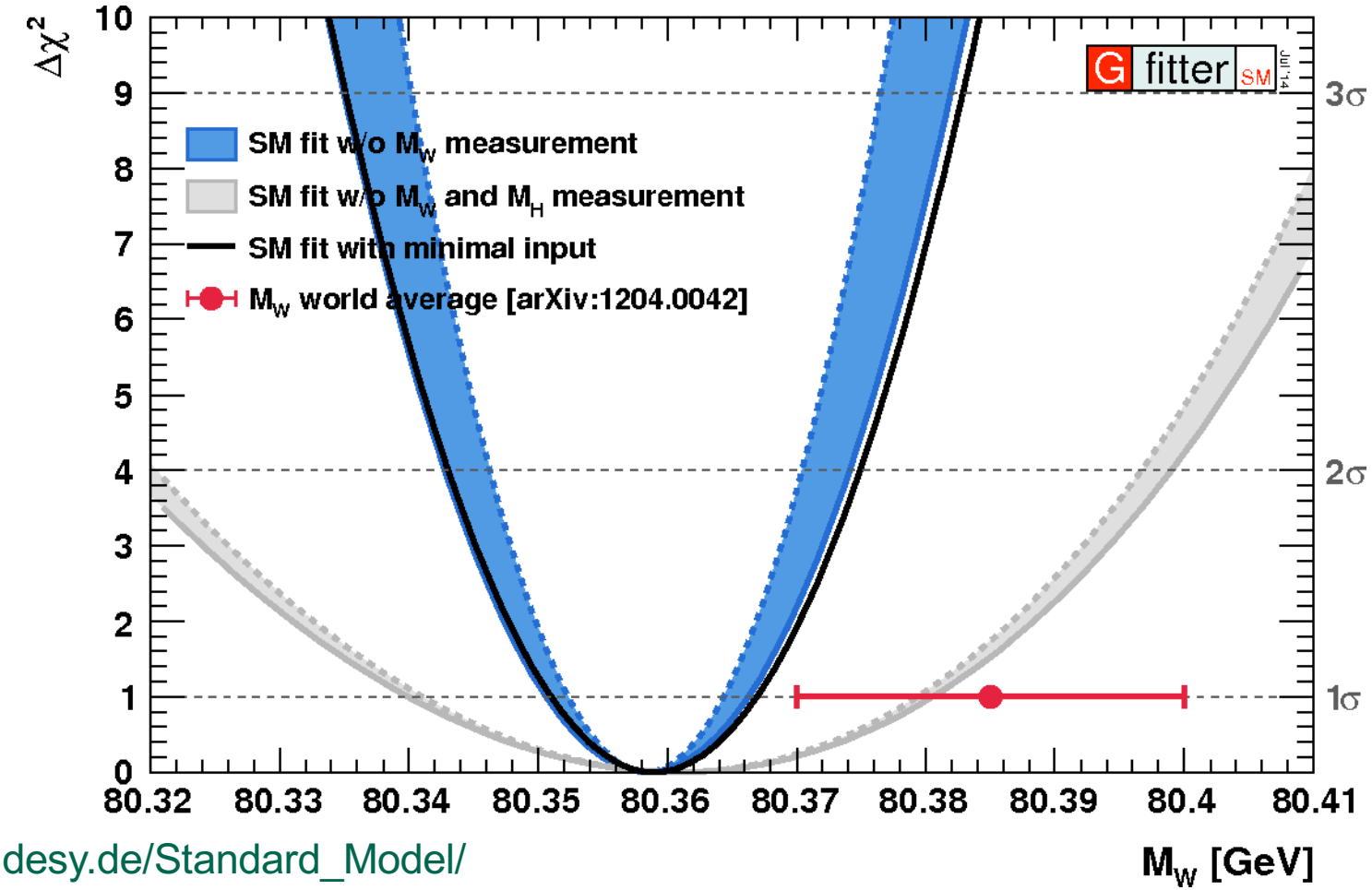
Generalization of χ^2 to n dimensions

- We have n measurements, $(m_1, \ldots, m_n)$ with a $n \times n$ covariance matrix (C_{ij})
- Expected values for $m_1, \ldots, m_n$, $M_1, \ldots, M_n$ may depend on some theory parameter(s) θ
- The following χ^2 can be minimized to have an estimate of the parameter(s) θ:

$$\begin{aligned}
\chi^2 &= \sum_{i,j=1}^{n} (m_i - M_i(\theta)) C_{ij}^{-1} (m_j - M_j(\theta)) \\
&= \begin{pmatrix} m_1 - M_1, & \cdots, & m_n - M_n \end{pmatrix} \begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}^{-1} \begin{pmatrix} m_1 - M_1 \\ \cdots \\ m_n - M_n \end{pmatrix} \\
&= (\boldsymbol{m} - \boldsymbol{M}(\theta))^T \boldsymbol{C}^{-1} (\boldsymbol{m} - \boldsymbol{M}(\theta))
\end{aligned}$$

Global electroweak fit

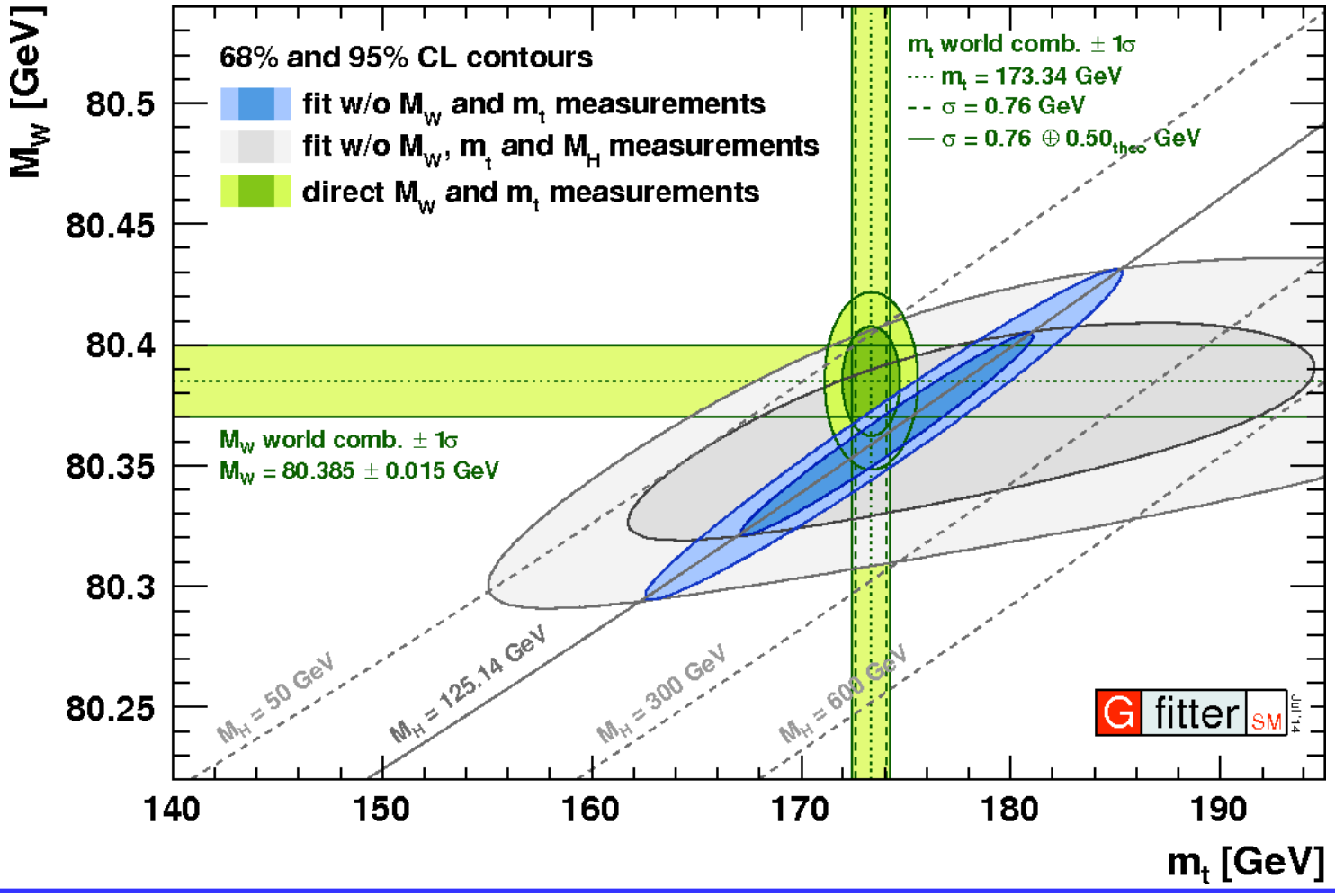
- A Global χ^2 fit to electroweak measurements predicts the W mass allowing a comparison with direct measurements



Details on:
http://gfitter.desy.de/Standard_Model/

More on electroweak fit

- W mass vs top-quark mass from global electroweak fit



Combining correlated measurements

- Correlation coefficient $\rho \neq 0$:

$$m_1 \pm \sigma_1, \quad m_2 \pm \sigma_2$$

- Build χ^2 including correlation terms:

$$\chi^2 = \begin{pmatrix} m - m_1 & m - m_2 \end{pmatrix} \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}^{-1} \begin{pmatrix} m - m_1 \\ m - m_2 \end{pmatrix}$$

- The χ^2 minimization gives:

$$m = \frac{m_1(\sigma_2^2 - \rho\sigma_1\sigma_2) + m_2(\sigma_1^2 - \rho\sigma_1\sigma_2)}{\sigma_1^2 - 2\rho\sigma_1\sigma_2 + \sigma_2^2}$$

$$\sigma_m^2 = \frac{\sigma_1^2\sigma_2^2(1-\rho)^2}{\sigma_1^2 - 2\rho\sigma_1\sigma_2 + \sigma_2^2}$$

a.k.a "**BLUE**":
Best **L**inear **U**nbiased **E**stimator

Correlated errors

- The "common error" σ_C is defined as: $\sigma_C^2 = \rho\sigma_1\sigma_2$
- Using error propagation, this also implies that:

$$\sigma^2_{m_1-m_2} = \left(\frac{\partial(m_1-m_2)}{\partial m_1}\right)^2 \sigma_1^2 + \left(\frac{\partial(m_1-m_2)}{\partial m_2}\right)^2 \sigma_2^2 + 2\left(\frac{\partial(m_1-m_2)}{\partial m_1}\right)\left(\frac{\partial(m_1-m_2)}{\partial m_2}\right)\rho\sigma_1\sigma_2$$

$$\sigma^2_{m_1-m_2} = (\sigma_1^2 - \sigma_C^2) + (\sigma_2^2 - \sigma_C^2)$$

- The previous formulas can be written as a weighted average:

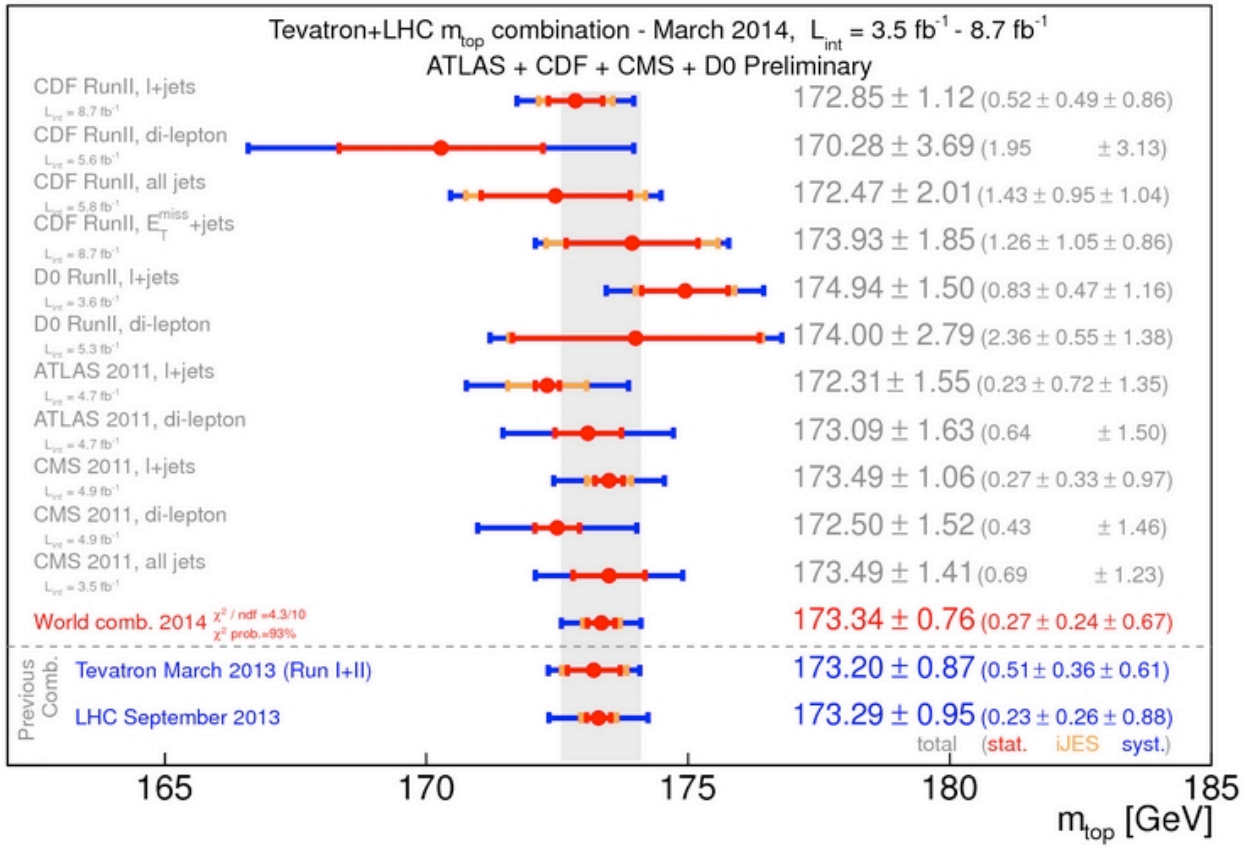
$$m = \frac{\frac{m_1}{\sigma_1^2-\sigma_C^2} + \frac{m_2}{\sigma_2^2-\sigma_C^2}}{\frac{1}{\sigma_1^2-\sigma_C^2} + \frac{1}{\sigma_2^2-\sigma_C^2}} = \frac{w_1 m_1 + w_2 m_2}{w_1 + w_2}$$

← Note: weights may be negative!

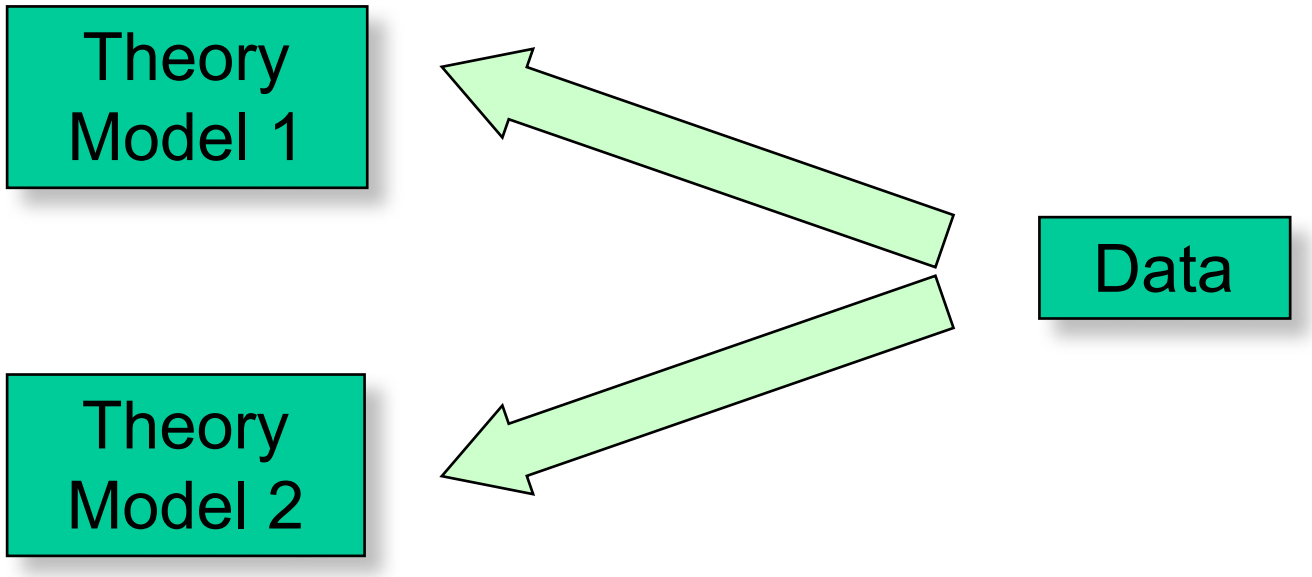
$$\sigma_m^2 = \frac{1}{\frac{1}{\sigma_1^2-\sigma_C^2} + \frac{1}{\sigma_2^2-\sigma_C^2}} + \sigma_C^2$$

Top-quark mass combination

- World average of top-quark mass measurements combined with BLUE
- Outdated by more recent measurements: determining and agreeing on all uncertainty contributions and correlation is a complex procedure and takes time…



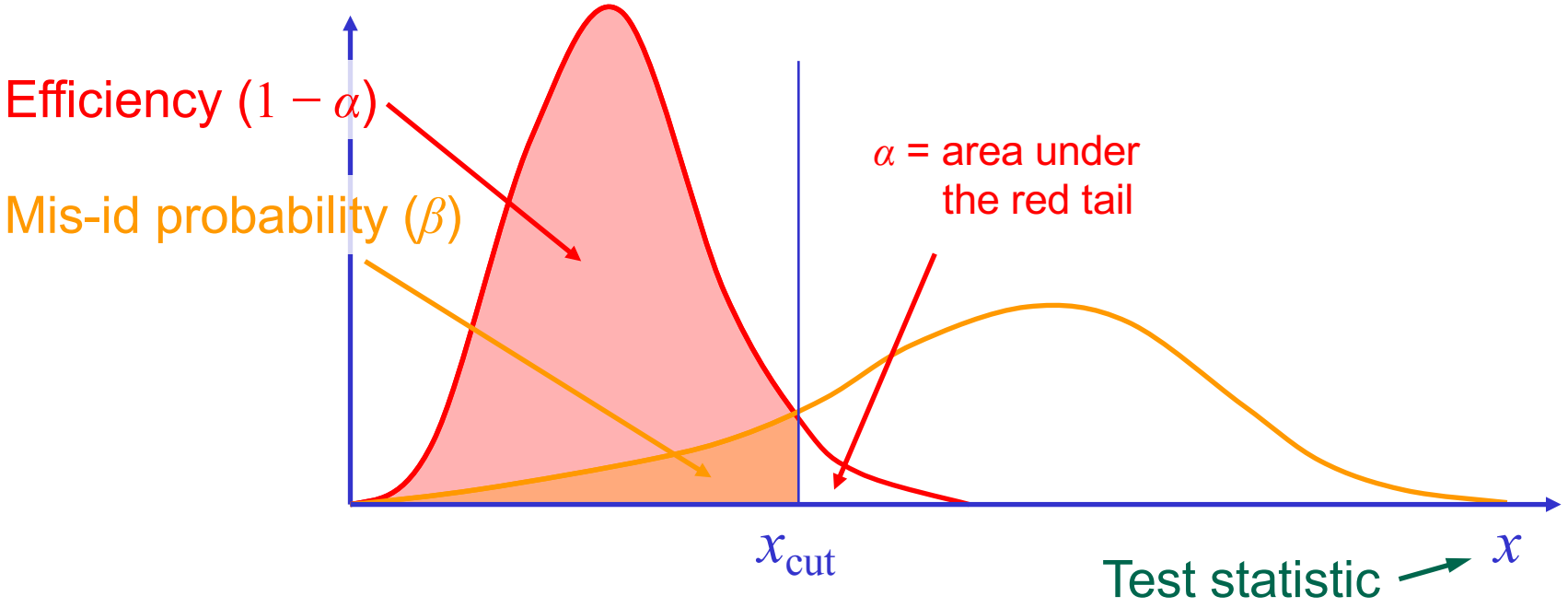
Hypothesis testing



Which hypothesis is the most consistent with the experimental data?

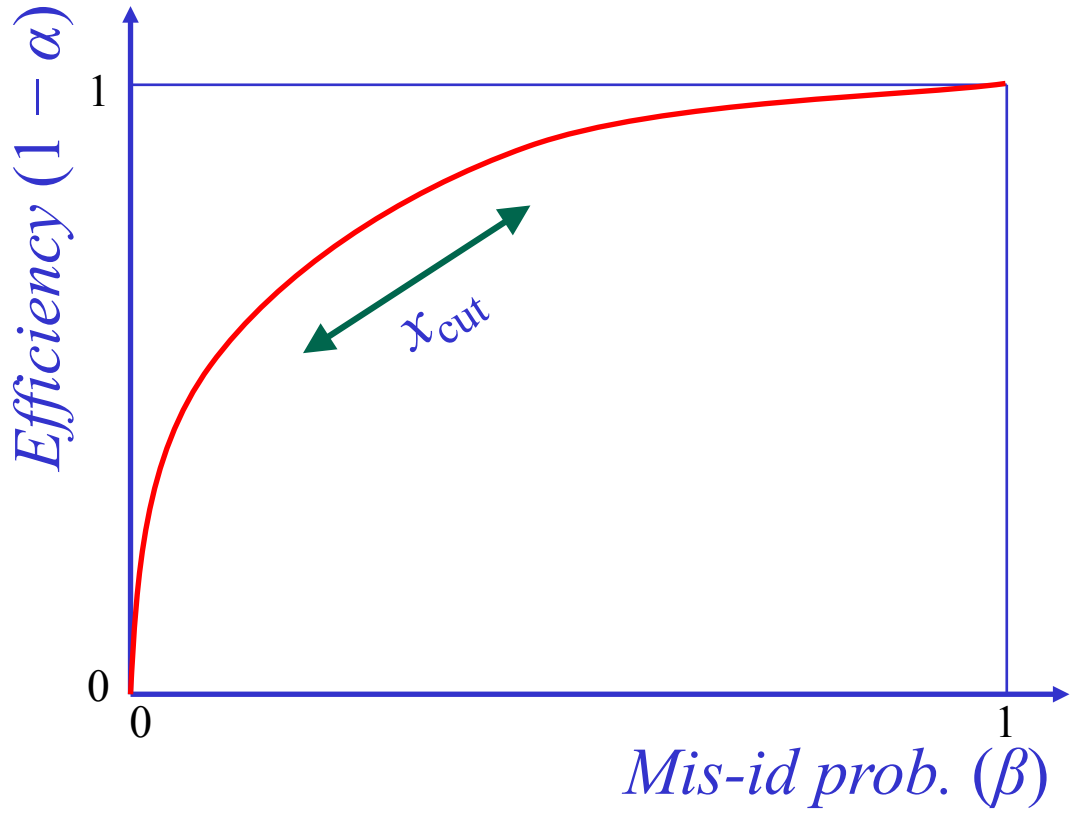
Simplest case: cut analysis

- Selection ("cut") on one (or more) variable(s):
  - If $x \leq x_{\text{cut}} \Rightarrow$ signal
  - Else, if $x > x_{\text{cut}} \Rightarrow$ background



Efficiency *vs* mis-id

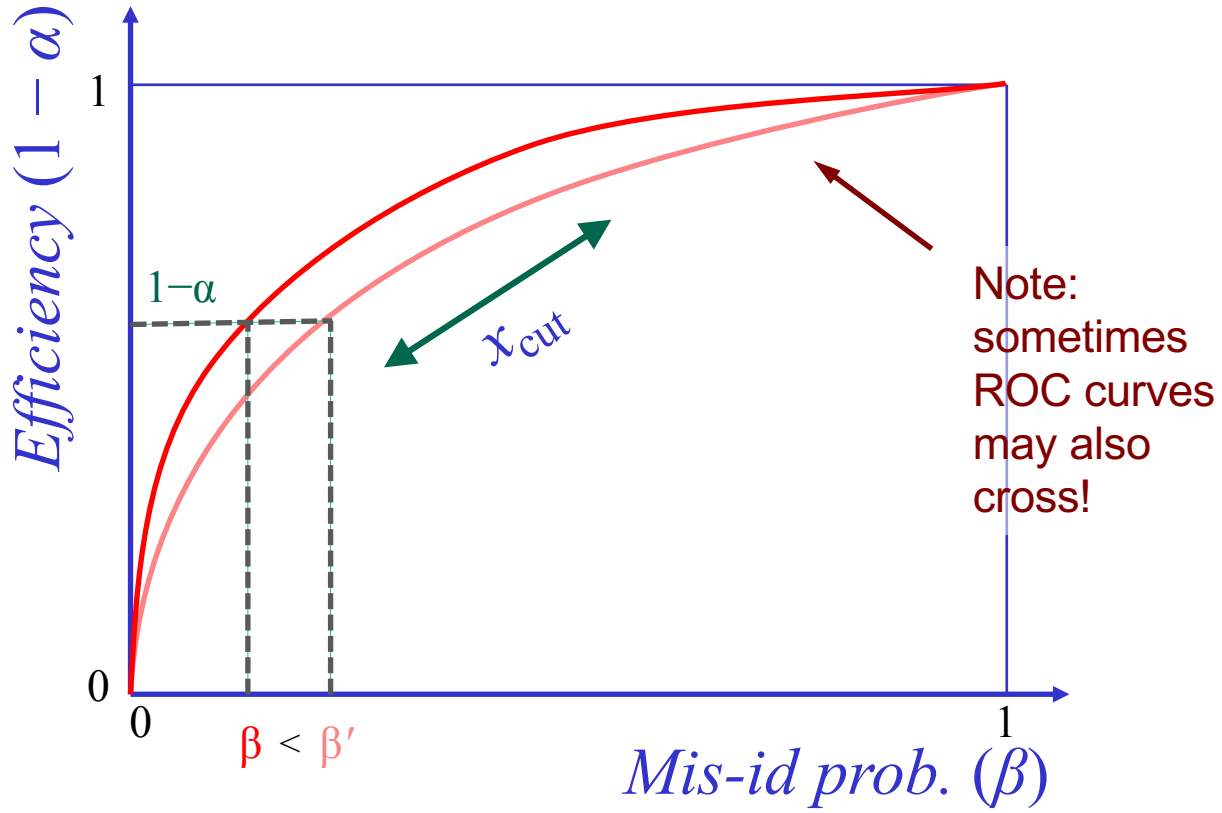
- Varying the applied cut on the test statistic both the efficiency and mis-id probability change



Sometimes also referred to as **ROC curve** (*Receiver Operating Characteristic*)

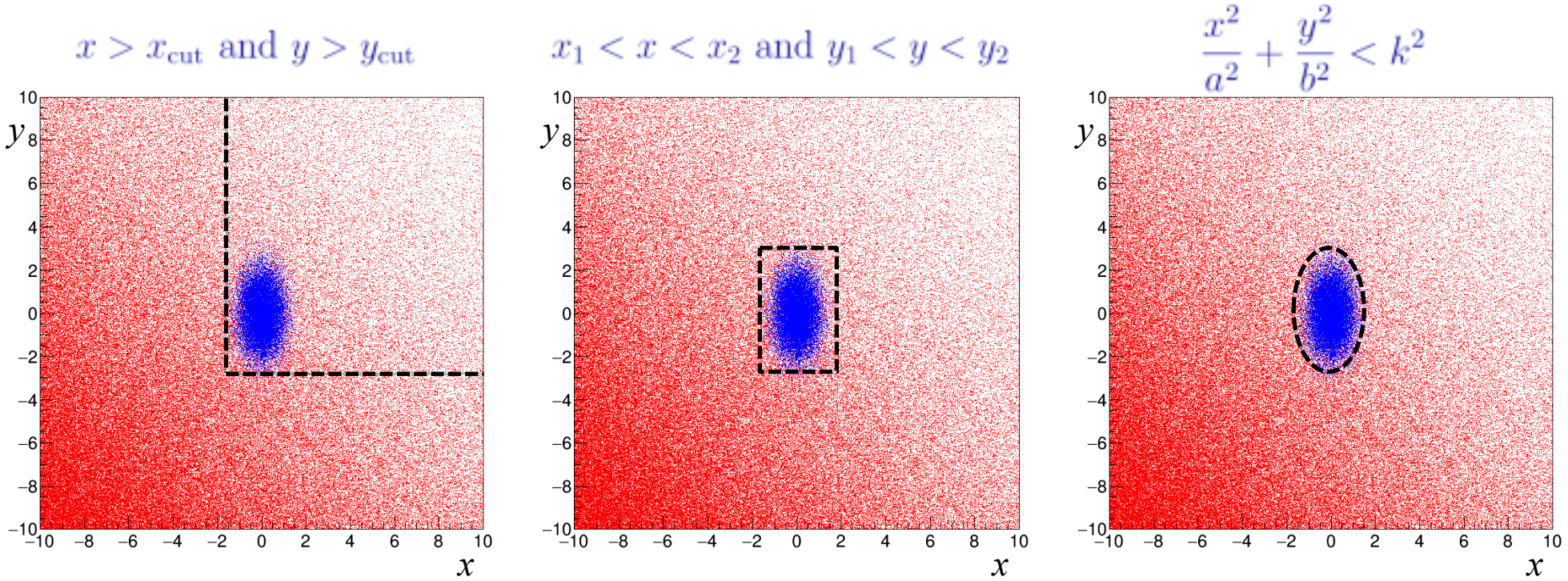
Performance comparison

- One test is preferable to another if, for the same level of efficiency ($1-\alpha$), it has lower mis-id probability (β)



Multidimensional case

- In case of multiple discriminating variables (multivariate), the choice of the optimal selection is not always straightforward



$P_s(x, y) = \text{Gauss}(x; 0, \sigma_x) \times \text{Gauss}(y; 0, \sigma_y)$, $P_b(x, y) = \alpha\, e^{-\alpha x} \times \beta\, e^{-\beta y}$

- In many cases it's convenient to find a single variable (test statistic) that 'summarizes' all the sample information

Terminology

- Statisticians' terminology is sometimes not very natural for physics applications, but it has become popular among physicists as well:

- H_0 = **null hypothesis**
  - Ex. 1: *"a sample contains only background"*
  - Ex. 2: *"a particle is a pion"*
- H_1 = **alternative hypothesis**
  - Ex. 1: *"a sample contains background + signal"*
  - Ex. 2: *"a particle is a muon"*

- **Test statistic**: a variable computed from our sample that discriminates between the two hypotheses H_0 and H_1. Usually a 'summary' of the information available in the sample
- α **= significance level**: probability to reject H_1 if $H_0$is assumed to be true (error of first kind, false positive)
  - α = 1 – selection efficiency
- β **= misidentification probability**, i.e.: probability to reject H_0 if H_1 is assumed to be true (error of second kind, false negative)
  - $1 - \beta$ = **power of the test**
- ***p*-value**: probability, assuming H_0, of observing a result at least as extreme as the observed test statistic

The Neyman-Pearson lemma

- For a fixed significance level (α) or signal efficiency ($1 - \alpha$), a selection based on the likelihood ratio gives the lowest possible mis-id probability (β):

$$\lambda(x) = \frac{L(x|H_1)}{L(x|H_0)} > k_\alpha$$

- The likelihood function can't always be determined exactly
- If we can't determine the exact likelihood function, we can choose other discriminators as test statistics that approximates the exact likelihood
- Neural Networks, Boosted Decision Trees and other machine-learning algorithms are example of discriminators that may closely approximate the performances of the exact likelihood ratio approaching the Neyman-Pearson limit

Multivariate discrimination

- In general, when we consider algorithms that provide a test statistic for samples with multiple variables we talk about multivariate discriminators
  - Simple mathematical algorithms exist, as well as complex implementation based on extensive CPU computations
- In general, the algorithms are 'trained' using input samples whose nature is known (training samples)
  - I.e.: where H_0 or H_1 is know to be true
  - Example: use data samples simulated with computer algorithms (Monte Carlo)
- Some of the most common problems:
  - The size of samples is finite, hence the true distributions for the considered hypotheses can't be determined exactly
  - The distribution of the input samples does not reproduce exactly the true distribution of real data (e.g.: systematic uncertainties)

Projective likelihood ratio

- The likelihood function is approximated by the product of projective PDF in each variable

$$\lambda(x) = \frac{L(x_1, \cdots, x_n|H_1)}{L(x_1, \cdots, x_n|H_0)} \simeq \frac{\prod_{i=1}^{n} f_i(x_i|H_1)}{\prod_{i=1}^{n} f_i(x_i|H_0)}$$

$x_1, \ldots, x_n$ approximately considered independent variables

- Exact only in case of independent variables
- The approximation may be improved if the variables are first rotated in order to eliminate correlation (principal component analysis)
  - Find eigenvectors of the covariance matrix
  - **Note**: uncorrelated variables are not necessarily independent

Fisher discriminant

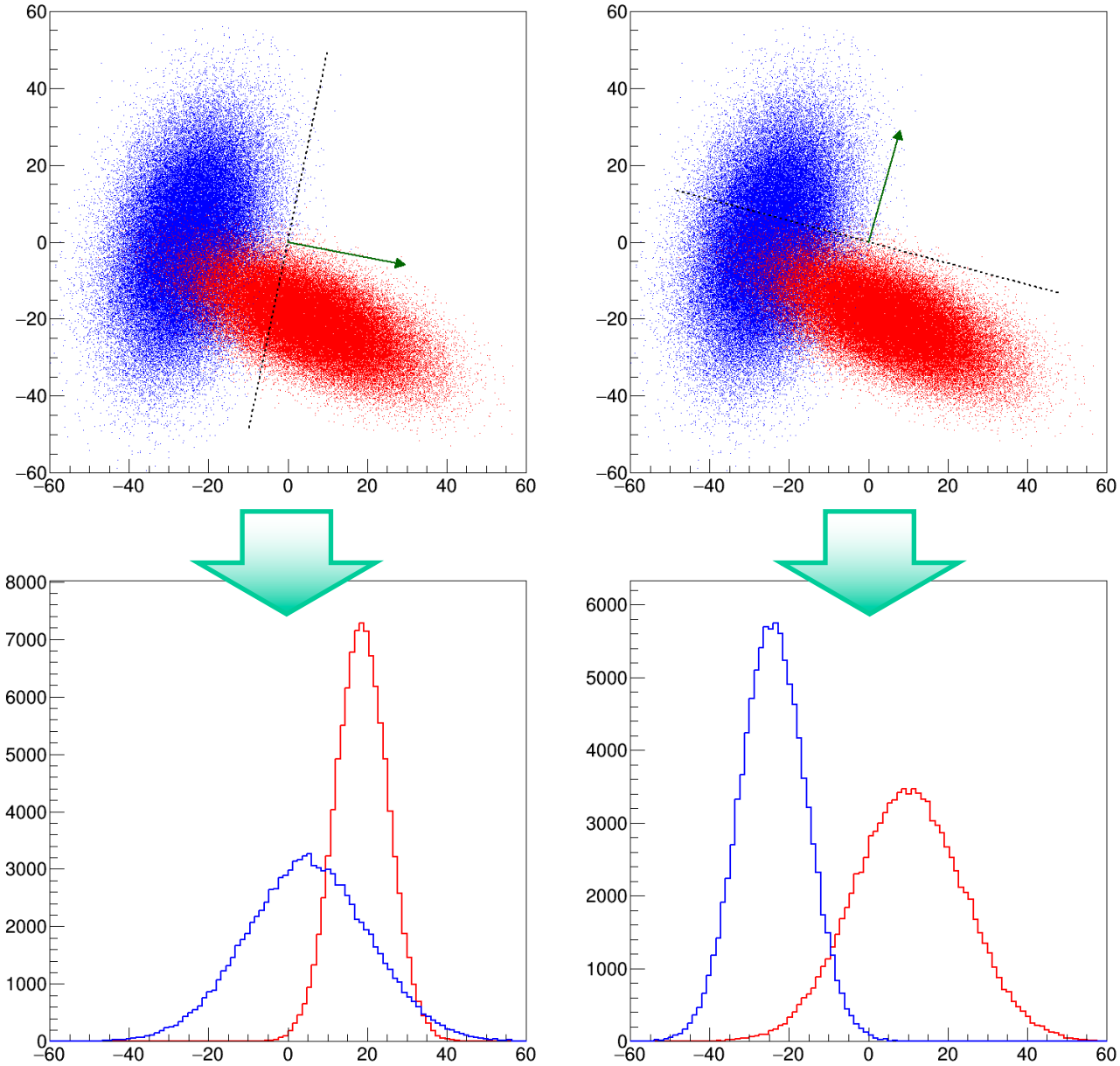
- Linear combination of input variables that maximizes the distance of the means of the two classes while minimizing the variance projected along a direction $\mathbf{w}$:

$$J(\mathbf{w}) = \frac{|\mu_0 - \mu_1|^2}{\sigma_0^2 + \sigma_1^2} = \frac{\mathbf{w}^T \cdot (\mathbf{m}_0 - \mathbf{m}_1)}{\mathbf{w}^T (\Sigma_0 + \Sigma_1)\mathbf{w}}$$

Sir Ronald Aylmer Fisher (1890-1962)

- The selection is achieved by requiring $J(\mathbf{w}) > J_{\text{cut}}$, which determines an hyperplane perpendicular to $\mathbf{w}$ that separates the two samples
- The maximization problem can be solved analytically using linear algebra

Fisher discriminant

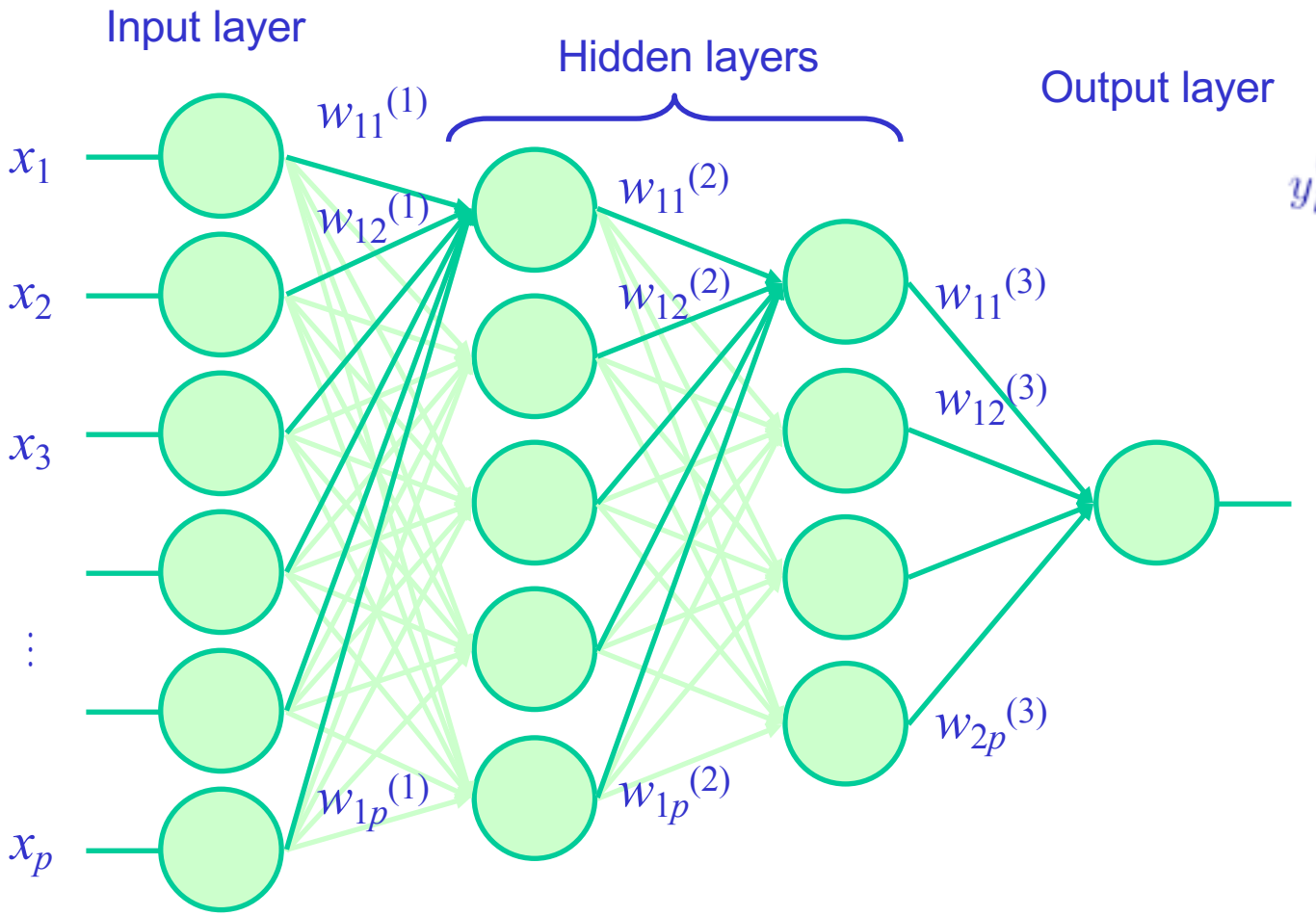


Projections along different directions achieve different overlap level

Maximum separation achieved by maximizing Fisher discriminant

Artificial Neural Networks

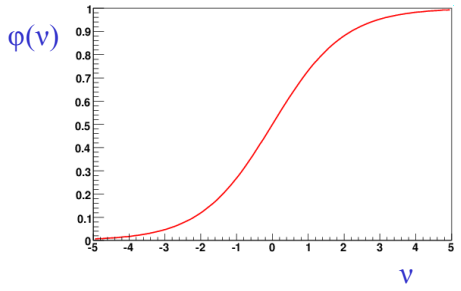
- Artificial simplified model of how neuron cells work: multilayer perceptron



$$y_k^{(n)}(\vec{x}) = \varphi\left(\sum_{j=1}^{p} w_{kj}^{(n)} x_j\right)$$

$\varphi(v)$ = Activation function

$$\varphi(\nu) = \frac{1}{1+e^{-\lambda\nu}}$$



Network training

- Find the optimal set of network parameters $w_{ij}^{(n)}$ that minimize the "loss function" defined on a set of N training events:

$$L(w) = \sum_{i=1}^{N} (y_i^{\text{true}} - y(\vec{x_i}))^2$$

- Where $y_i^{\text{true}} = 1$ for signal (H_1), 0 for background (H_0)
- Usually achieved with stochastic gradient descent: weights are modified for each training event (back propagation):

$$w_{ij} \rightarrow w_{ij} - \eta \frac{\partial L(w)}{\partial w_{ij}}$$

- The parameter η controls the learning rate
- Variations on the training methods exist

ANN and function modeling

- Artificial neural network with a single hidden layer may approximate any analytical function within a given approximation if the number of neurons is sufficiently high
- Demonstration in:
  - H. N. Mhaskar, Neural Computation, Vol. 8, No. 1, Pages 164-177 (1996), *Neural Networks for Optimal Approximation of Smooth and Analytic Functions*:

"We prove that neural networks with a single hidden layer are capable of providing an optimal order of approximation for functions assumed to possess a given number of derivatives, if the activation function evaluated by each principal element satisfies certain technical conditions"

- Anyway, the finite number of layer may lead to reaching this goal only approximately
- Deep learning: networks with several hidden layers
  - Can manage complex variables combinations, e.g.: exploiting invariant mass distributions using four-vectors as input!
  - Almost untreatable in the past, a lot of progress has been done recently, with better training algorithms and more easily available CPU power

Decision Trees

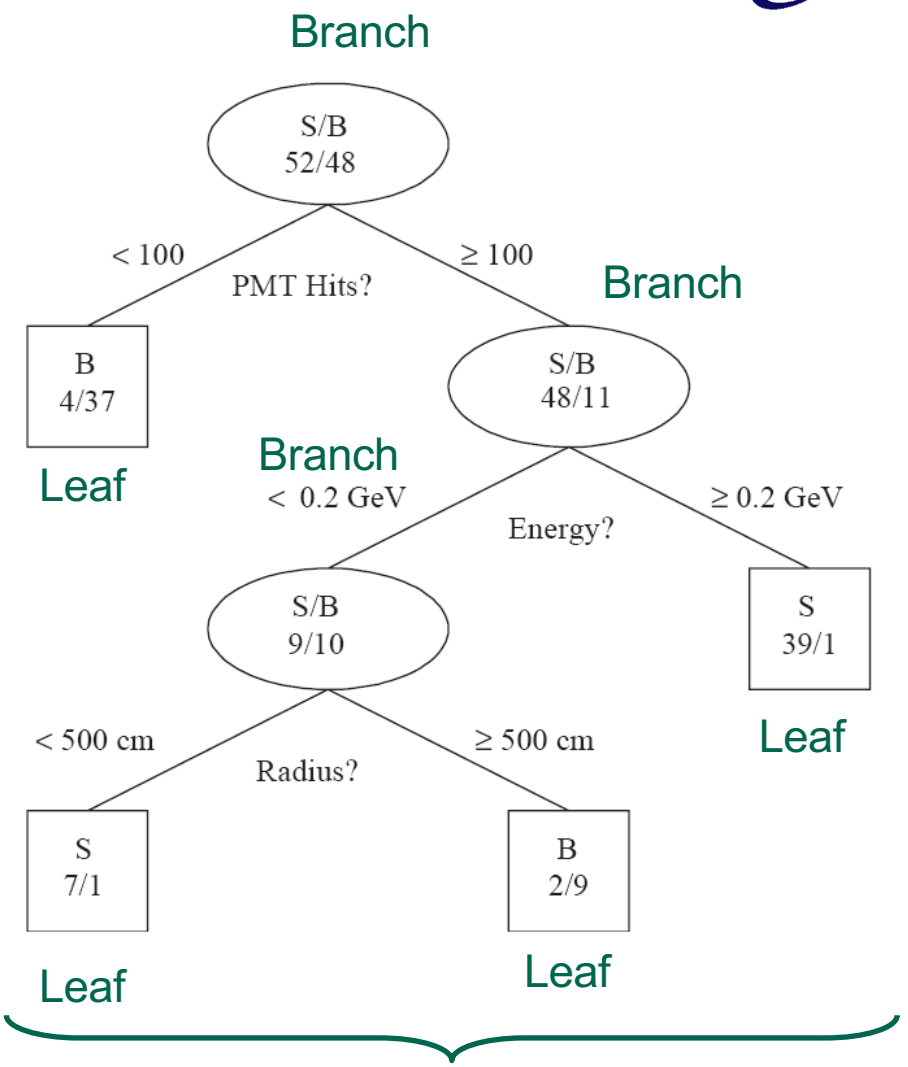
- Cuts are applied sequentially
- Each cut splits the sample into nodes
  - Nodes where signal or background is largely dominant are classified as leafs
    - Alternatively: stop splitting if too few events per note, total number of nodes too high, …
  - One branch = one sequence of cuts
- Cuts can be optimized to achieve the best split level: maximize for each node the gain of Gini index:

$$G = P(1-P)$$

P = node purity

$G = 0$ for nodes with all S or all B events

$$\text{Gain} = N_{\text{parent}} G_{\text{parent}} - N_{\text{left ch.}} G_{\text{left ch.}} - N_{\text{left ch.}} G_{\text{left ch.}}$$



- Alternative metrics exist
  E.g.: cross entropy $= -(P \ln P + (1-P)\ln(1-P))$, …

From trees to forests

- Boosted Decision Trees
  - Combine a large number of decision trees (forest) using different weights. Usually: $\mathcal{O}(1000)$ trees used
  - More performant and stable than a single optimized tree
- Boosting achieved by iteratively reweighting training sample according to classifier output in previous iteration

1. Reweight events using previous iteration's classifier result
2. Build and optimize a new tree with reweighted events
3. Give a score to each tree
4. The final BDT classifier result is the weighted average over all trees, using the given scores as weights:

$$y(\vec{x}) = \sum_{k=1}^{N_c} w_i C^{(i)}(\vec{x})$$

Adaptive boosting

- Misclassified events are reweight according to the fraction of classification errors of the previous tree:

$$\frac{1-f}{f}, f = \frac{N_{\text{misclassified}}}{N_{\text{tot}}}$$

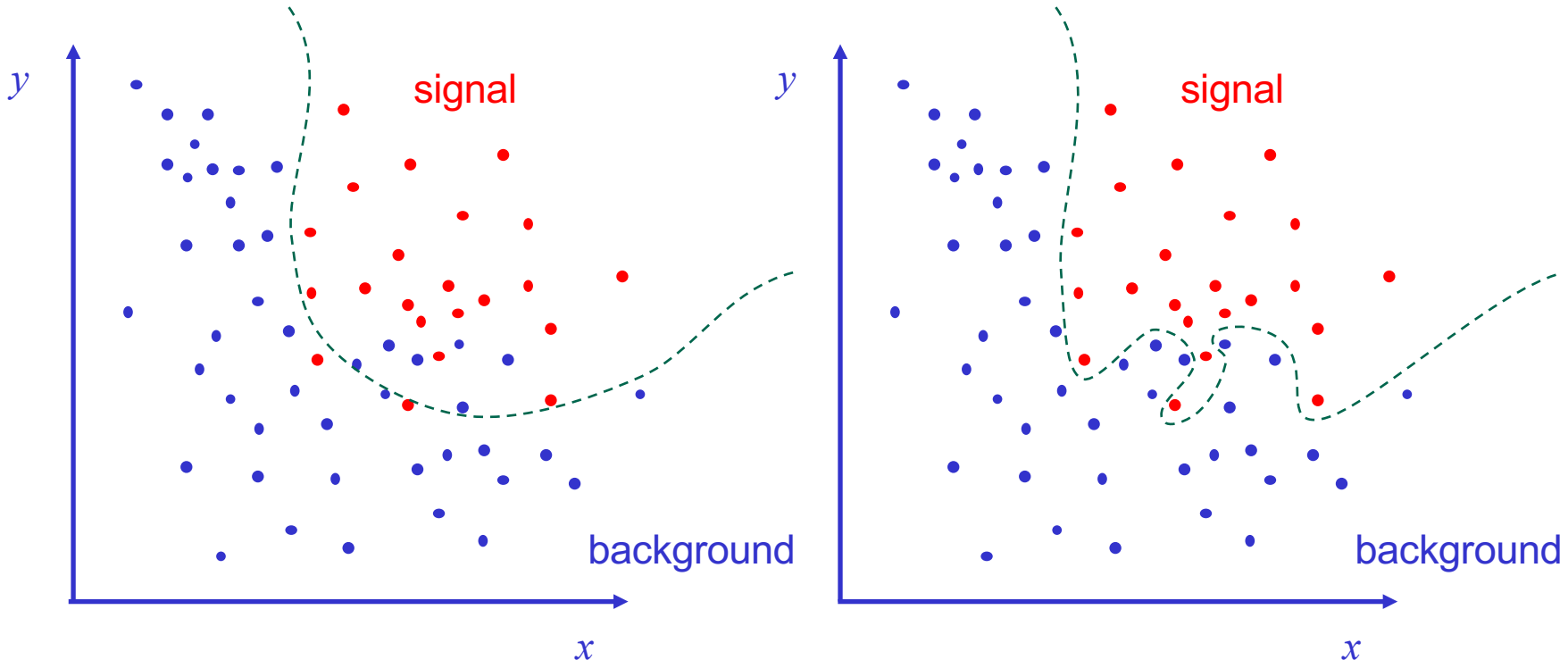
- Also use (log of) misclassification fraction as score for each tree:

$$y(\vec{x}) = \sum_{k=1}^{N_c} \log\left(\frac{1-f^{(i)}}{f^{(i)}}\right) C^{(i)}(\vec{x})$$

- Next iteration will better perform on events poorly classified in the previous iteration
- Further variations and more algorithms available

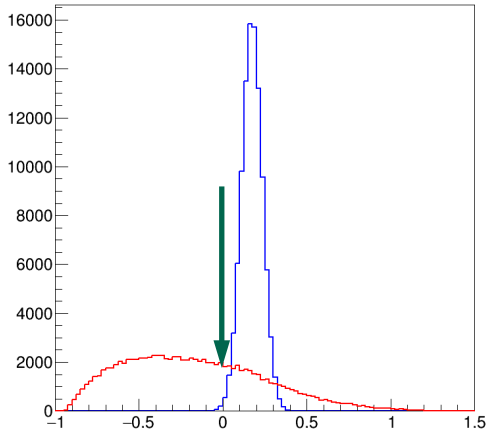
Overtraining

- Algorithms may learn too much from the training sample, exploiting features that are only due to random fluctuations
- Check for overtraining comparing the discriminator's distributions for the training sample and for an independent test sample (consistent distributions = no overtraining)

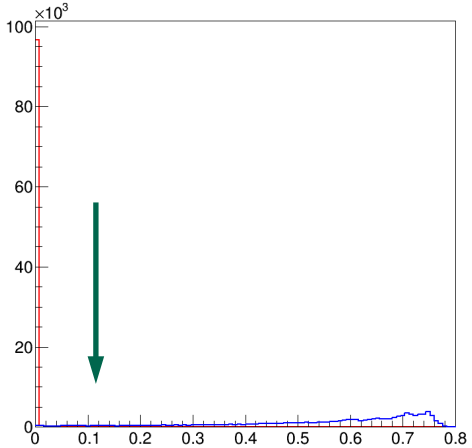


Comparing discriminants

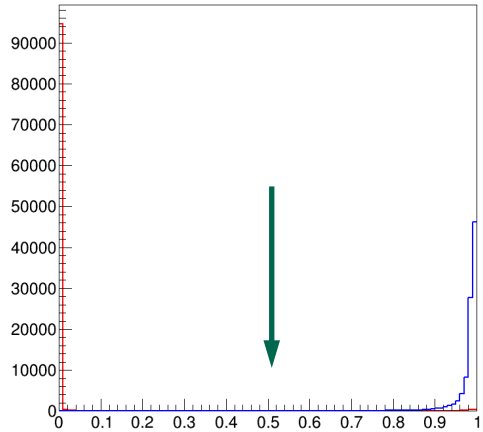
Fisher



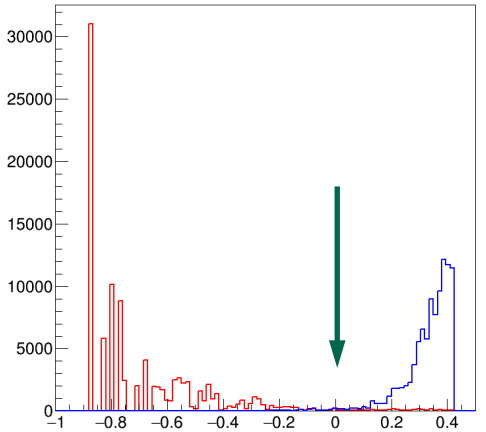
Proj. likelihood ratio



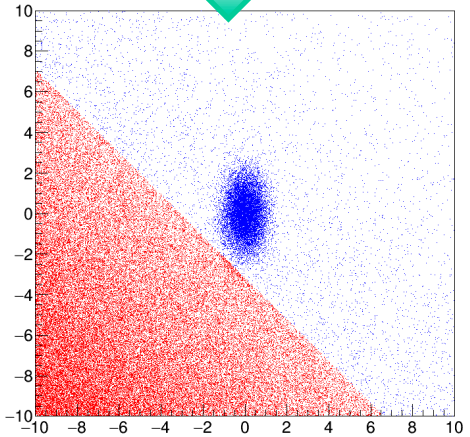
Neural Network



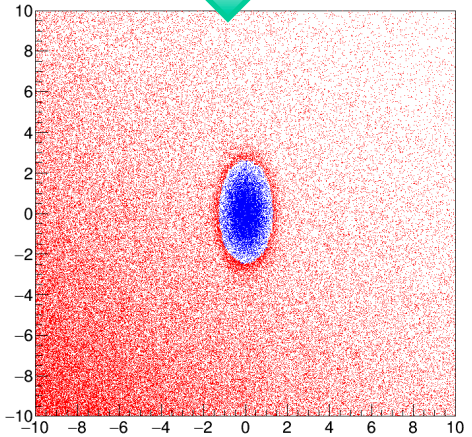
BDT



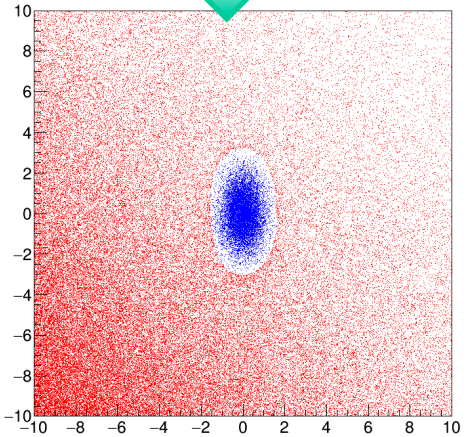
F > 0



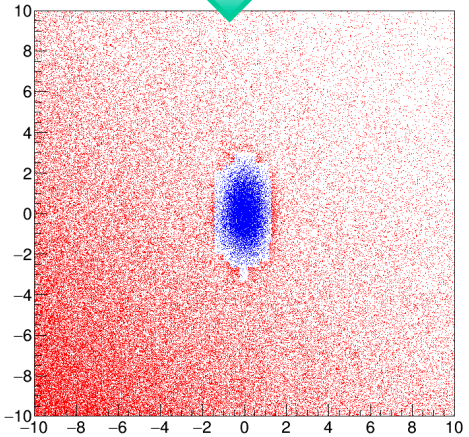
L > 0.1



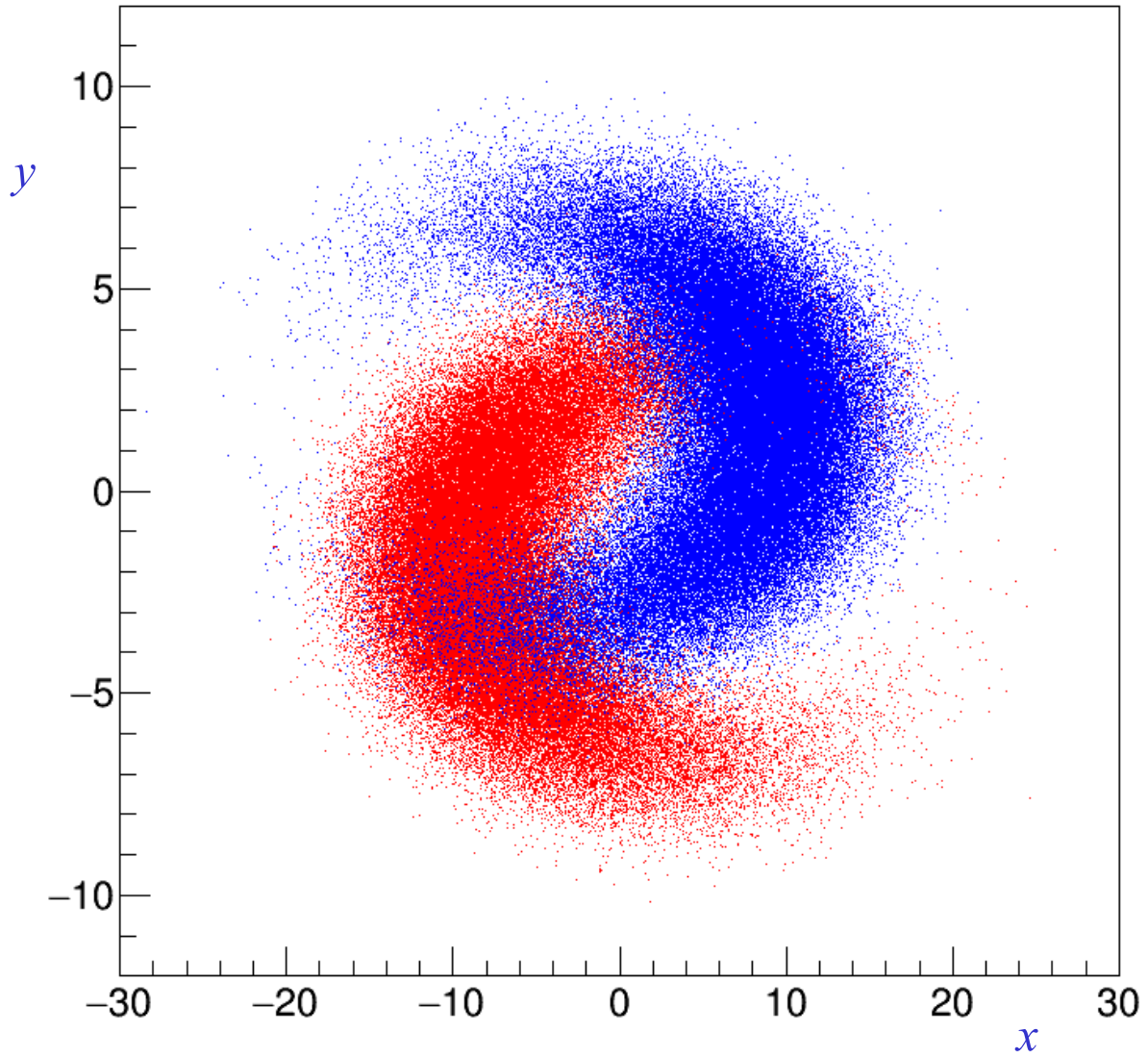
NN > 0.5



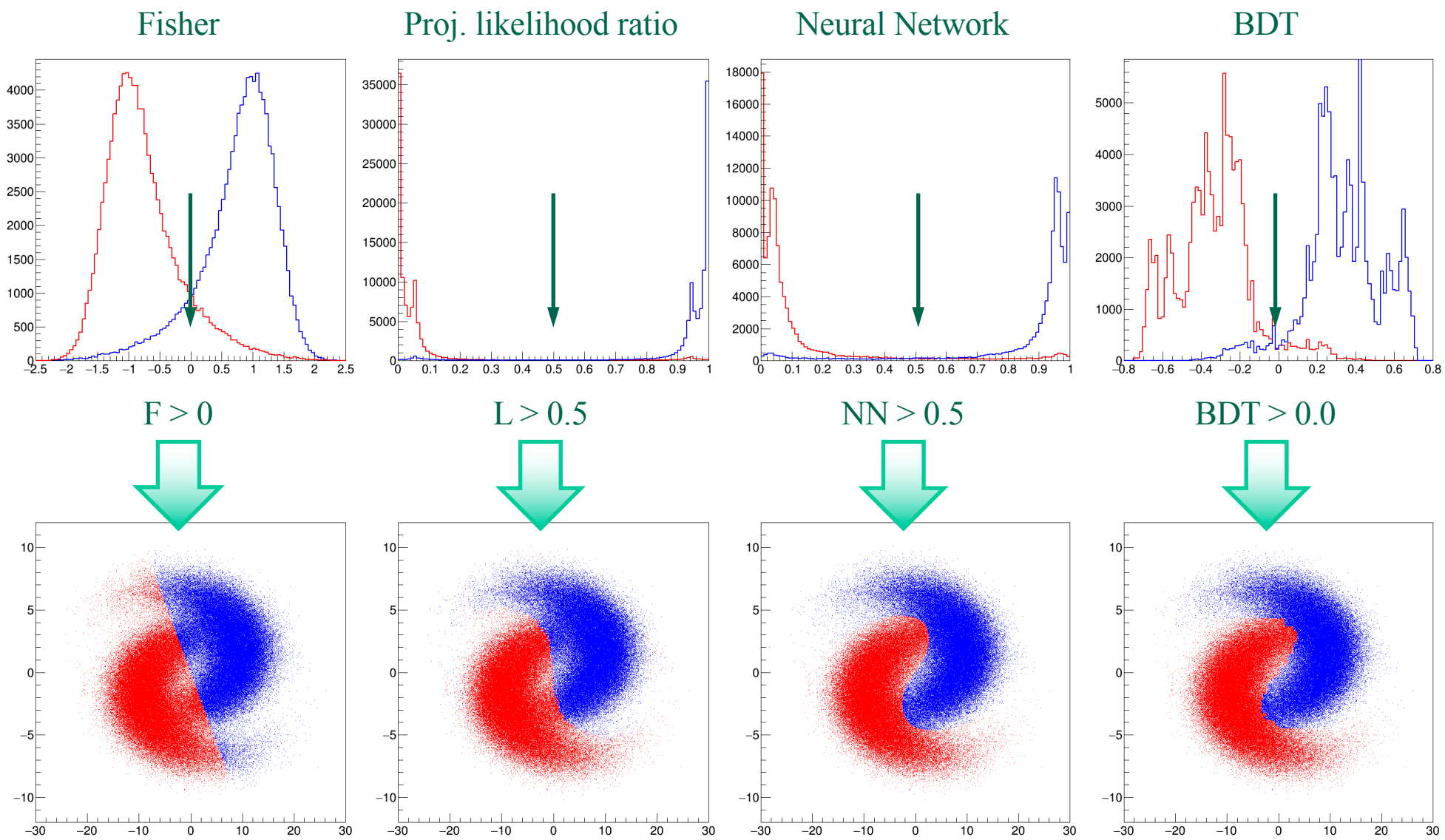
BDT > 0.0



Strongly nonlinear example

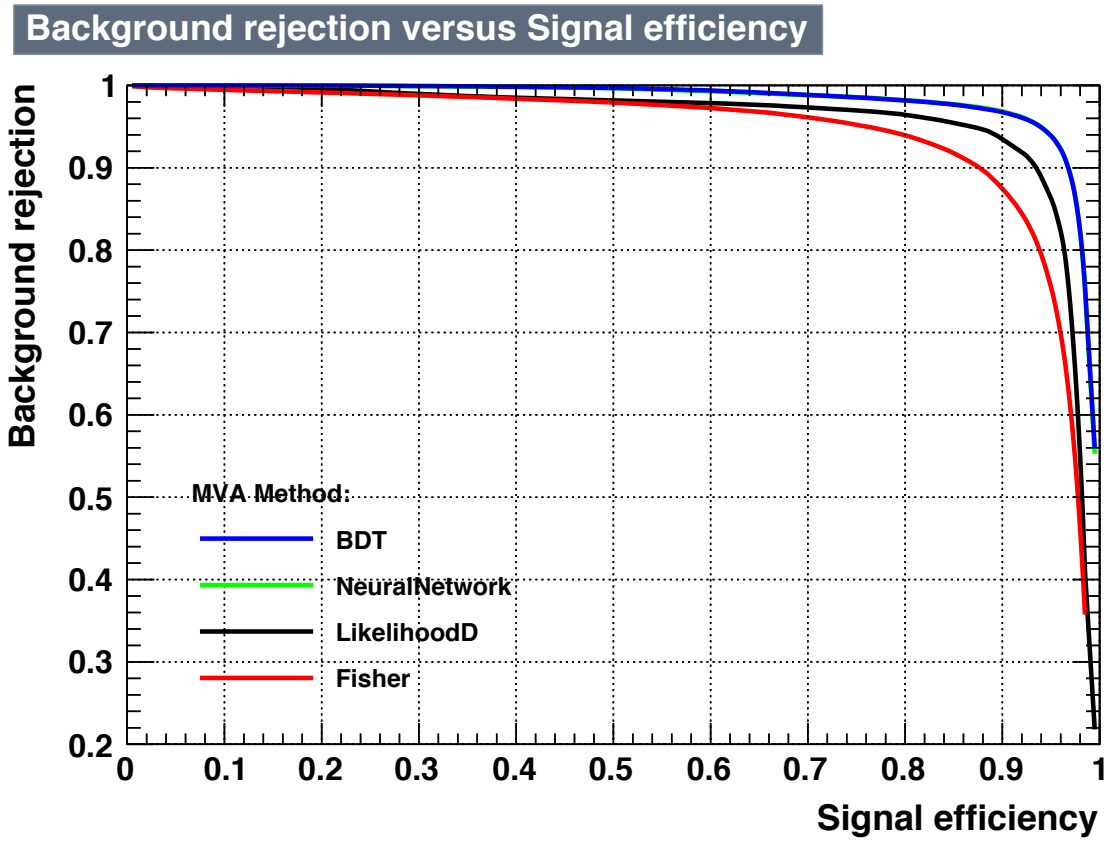


Performances



ROC curve

- TMVA implementation under ROOT
- GUI provided for standard set of plots



ROC curve from the standard TMVA GUI available in ROOT