

Statistical Methods for Particle Physics

Lecture 2: further topics

<https://indico.fnal.gov/conferenceTimeTable.py?confId=11505>



Lectures on Statistics
HCPSS – Fermilab
11,12 August 2016



Glen Cowan
Physics Department
Royal Holloway, University of London
g.cowan@rhul.ac.uk
www.pp.rhul.ac.uk/~cowan

Outline

Lecture 1: Introduction and review of fundamentals

Probability, random variables, pdfs

Parameter estimation, maximum likelihood

Statistical tests for discovery and limits

→ Lecture 2: Further topics

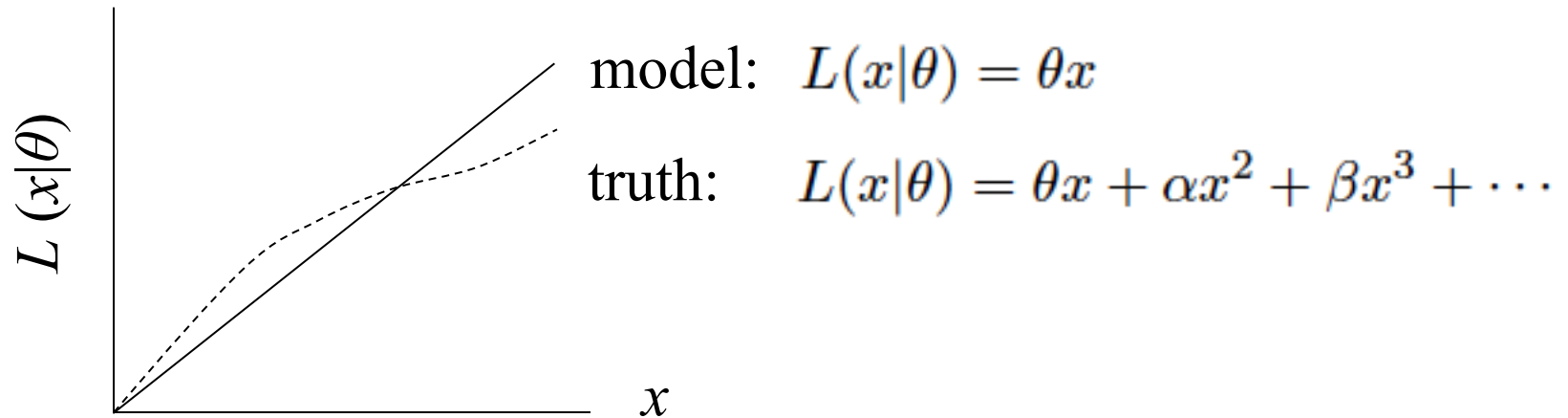
(Brief overview of multivariate methods) → extra slides

Nuisance parameters and systematic uncertainties

Experimental sensitivity

Systematic uncertainties and nuisance parameters

In general our model of the data is not perfect:



Can improve model by including additional adjustable parameters.

$$L(x|\theta) \rightarrow L(x|\theta, \nu)$$

Nuisance parameter \leftrightarrow systematic uncertainty. Some point in the parameter space of the enlarged model should be “true”.

Presence of nuisance parameter decreases sensitivity of analysis to the parameter of interest (e.g., increases variance of estimate).

Example: fitting a straight line

Data: (x_i, y_i, σ_i) , $i = 1, \dots, n$.

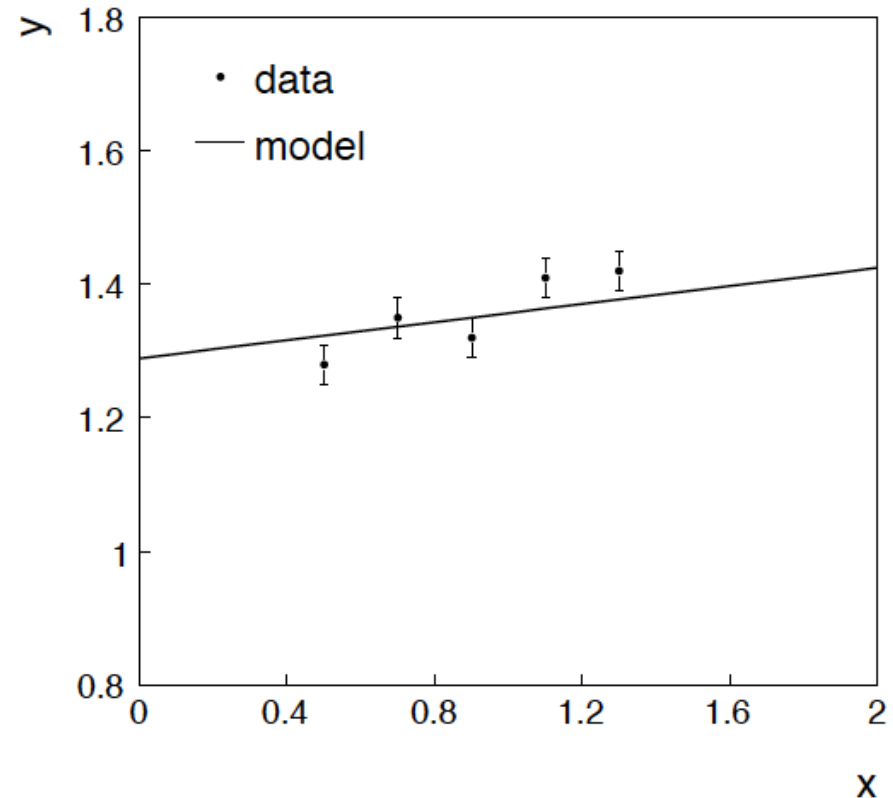
Model: y_i independent and all follow $y_i \sim \text{Gauss}(\mu(x_i), \sigma_i)$

$$\mu(x; \theta_0, \theta_1) = \theta_0 + \theta_1 x,$$

assume x_i and σ_i known.

Goal: estimate θ_0

Here suppose we don't care about θ_1 (example of a “nuisance parameter”)



Maximum likelihood fit with Gaussian data

In this example, the y_i are assumed independent, so the likelihood function is a product of Gaussians:

$$L(\theta_0, \theta_1) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_i} \exp \left[-\frac{1}{2} \frac{(y_i - \mu(x_i; \theta_0, \theta_1))^2}{\sigma_i^2} \right],$$

Maximizing the likelihood is here equivalent to minimizing

$$\chi^2(\theta_0, \theta_1) = -2 \ln L(\theta_0, \theta_1) + \text{const} = \sum_{i=1}^n \frac{(y_i - \mu(x_i; \theta_0, \theta_1))^2}{\sigma_i^2}.$$

i.e., for Gaussian data, ML same as Method of Least Squares (LS)

θ_1 known a priori

$$L(\theta_0) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_i} \exp \left[-\frac{1}{2} \frac{(y_i - \mu(x_i; \theta_0, \theta_1))^2}{\sigma_i^2} \right].$$

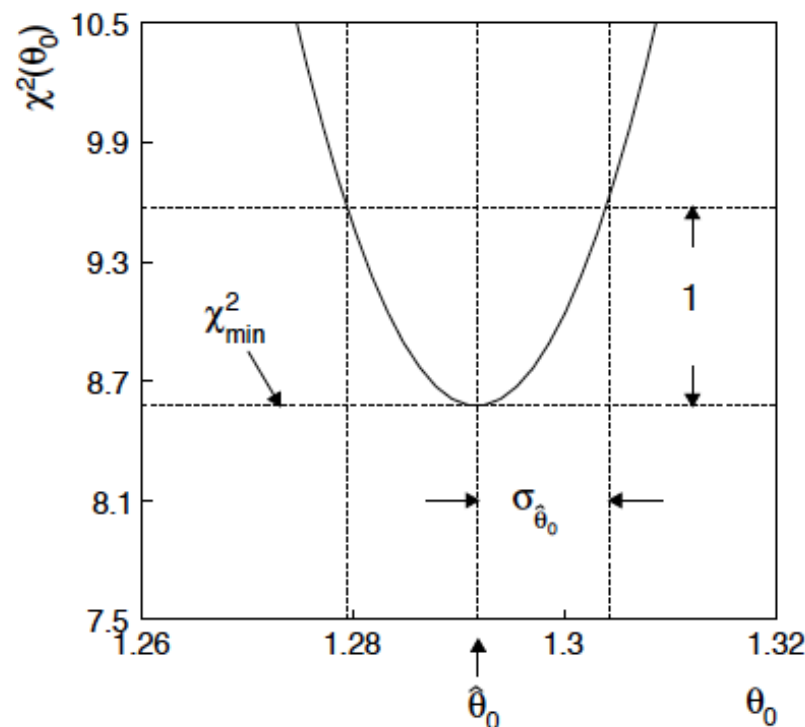
$$\chi^2(\theta_0) = -2 \ln L(\theta_0) + \text{const} = \sum_{i=1}^n \frac{(y_i - \mu(x_i; \theta_0, \theta_1))^2}{\sigma_i^2}.$$

For Gaussian y_i , ML same as LS

Minimize $\chi^2 \rightarrow$ estimator $\hat{\theta}_0$.

Come up one unit from χ_{\min}^2

to find $\sigma_{\hat{\theta}_0}$.



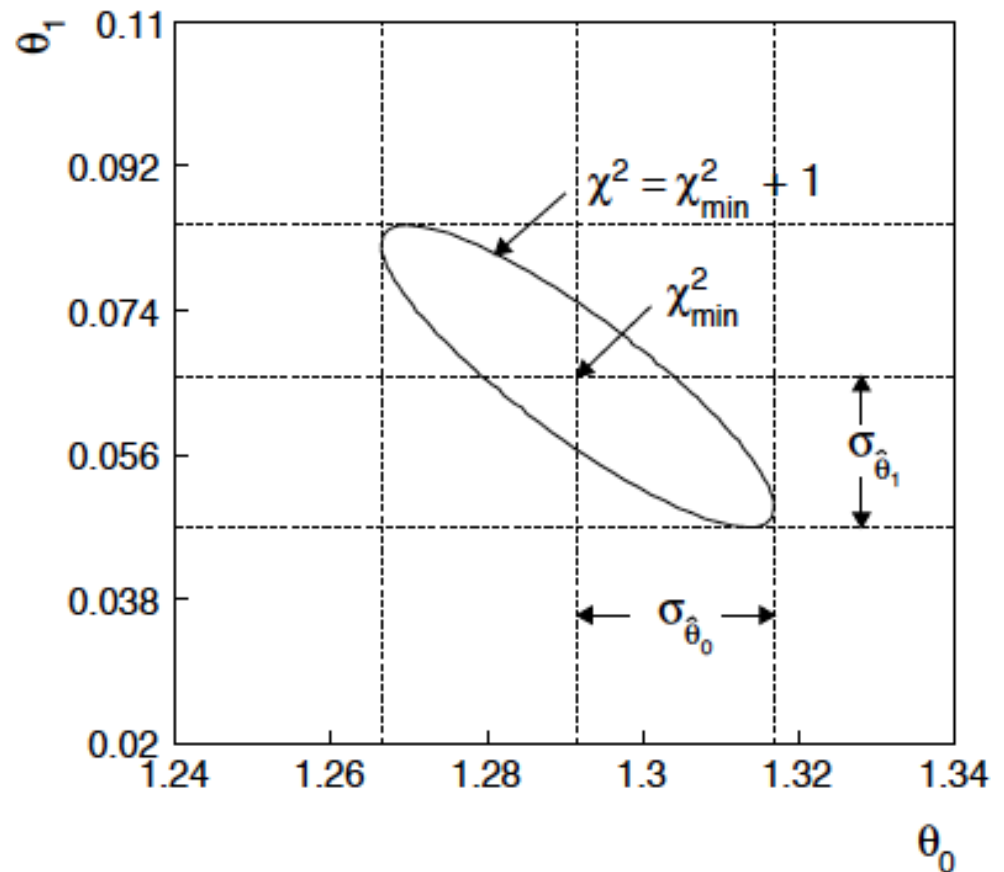
ML (or LS) fit of θ_0 and θ_1

$$\chi^2(\theta_0, \theta_1) = -2 \ln L(\theta_0, \theta_1) + \text{const} = \sum_{i=1}^n \frac{(y_i - \mu(x_i; \theta_0, \theta_1))^2}{\sigma_i^2} .$$

Standard deviations from
tangent lines to contour

$$\chi^2 = \chi_{\min}^2 + 1 .$$

Correlation between
 $\hat{\theta}_0$, $\hat{\theta}_1$ causes errors
to increase.

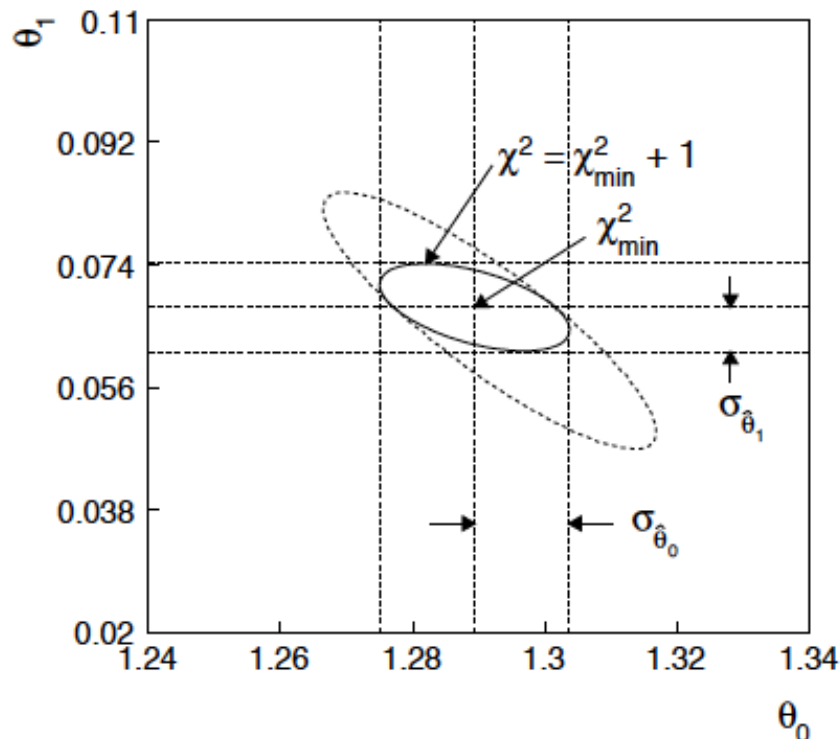


If we have a measurement $t_1 \sim \text{Gauss}(\theta_1, \sigma_{t_1})$

$$L(\theta_0, \theta_1) = \frac{1}{\sqrt{2\pi}\sigma_t} e^{-(t_1 - \theta_1)^2 / 2\sigma_{t_1}^2} \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{1}{2} \frac{(y_i - \mu(x_i; \theta_0, \theta_1))^2}{\sigma_i^2}\right]$$

$$\chi^2(\theta_0, \theta_1) = \sum_{i=1}^n \frac{(y_i - \mu(x_i; \theta_0, \theta_1))^2}{\sigma_i^2} + \frac{(t_1 - \theta_1)^2}{\sigma_{t_1}^2}$$

The information on θ_1
improves accuracy of $\hat{\theta}_0$.



The Bayesian approach

In Bayesian statistics we can associate a probability with a hypothesis, e.g., a parameter value θ .

Interpret probability of θ as ‘degree of belief’ (subjective).

Need to start with ‘**prior pdf**’ $\pi(\theta)$, this reflects degree of belief about θ before doing the experiment.

Our experiment has data x , \rightarrow **likelihood function** $L(x|\theta)$.

Bayes’ theorem tells how our beliefs should be updated in light of the data x :

$$p(\theta|x) = \frac{L(x|\theta)\pi(\theta)}{\int L(x|\theta')\pi(\theta') d\theta'} \propto L(x|\theta)\pi(\theta)$$

Posterior pdf $p(\theta|x)$ contains all our knowledge about θ .

Bayesian method

We need to associate prior probabilities with θ_0 and θ_1 , e.g.,

$$\begin{aligned} \pi(\theta_0, \theta_1) &= \pi_0(\theta_0) \pi_1(\theta_1) && \text{'non-informative', in any} \\ \pi_0(\theta_0) &= \text{const.} && \text{case much broader than } L(\theta_0) \\ \pi_1(\theta_1) &= \frac{1}{\sqrt{2\pi}\sigma_{t_1}} e^{-(\theta_1 - t_1)^2 / 2\sigma_{t_1}^2} && \leftarrow \text{based on previous} \\ &&& \text{measurement} \end{aligned}$$

Putting this into Bayes' theorem gives:

$$p(\theta_0, \theta_1 | \vec{y}) \propto \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_i} e^{-(y_i - \mu(x_i; \theta_0, \theta_1))^2 / 2\sigma_i^2} \pi_0 \frac{1}{\sqrt{2\pi}\sigma_{t_1}} e^{-(\theta_1 - t_1)^2 / 2\sigma_{t_1}^2}$$

↑
↑
↑

posterior
∝
likelihood
×
prior

Bayesian method (continued)

We then integrate (marginalize) $p(\theta_0, \theta_1 | x)$ to find $p(\theta_0 | x)$:

$$p(\theta_0 | x) = \int p(\theta_0, \theta_1 | x) d\theta_1 .$$

In this example we can do the integral (rare). We find

$$p(\theta_0 | x) = \frac{1}{\sqrt{2\pi}\sigma_{\theta_0}} e^{-(\theta_0 - \hat{\theta}_0)^2 / 2\sigma_{\theta_0}^2} \quad \text{with}$$

$$\hat{\theta}_0 = \text{same as ML estimator}$$

$$\sigma_{\theta_0} = \sigma_{\hat{\theta}_0} \text{ (same as before)}$$

Usually need numerical methods (e.g. Markov Chain Monte Carlo) to do integral.

Digression: marginalization with MCMC

Bayesian computations involve integrals like

$$p(\theta_0|x) = \int p(\theta_0, \theta_1|x) d\theta_1 .$$

often high dimensionality and impossible in closed form,
also impossible with ‘normal’ acceptance-rejection Monte Carlo.

Markov Chain Monte Carlo (MCMC) has revolutionized
Bayesian computation.




MCMC (e.g., Metropolis-Hastings algorithm) generates
correlated sequence of random numbers:

cannot use for many applications, e.g., detector MC;
effective stat. error greater than if all values independent .

Basic idea: sample multidimensional $\vec{\theta}$,
look, e.g., only at distribution of parameters of interest.

MCMC basics: Metropolis-Hastings algorithm

Goal: given an n -dimensional pdf $p(\vec{\theta})$,
generate a sequence of points $\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3, \dots$

- 1) Start at some point $\vec{\theta}_0$
- 2) Generate $\vec{\theta} \sim q(\vec{\theta}; \vec{\theta}_0)$  Proposal density $q(\vec{\theta}; \vec{\theta}_0)$
e.g. Gaussian centred
about $\vec{\theta}_0$
- 3) Form Hastings test ratio $\alpha = \min \left[1, \frac{p(\vec{\theta})q(\vec{\theta}_0; \vec{\theta})}{p(\vec{\theta}_0)q(\vec{\theta}; \vec{\theta}_0)} \right]$
- 4) Generate $u \sim \text{Uniform}[0, 1]$
- 5) If $u \leq \alpha$, $\vec{\theta}_1 = \vec{\theta}$,  move to proposed point
else $\vec{\theta}_1 = \vec{\theta}_0$  old point repeated
- 6) Iterate

Metropolis-Hastings (continued)

This rule produces a *correlated* sequence of points (note how each new point depends on the previous one).

For our purposes this correlation is not fatal, but statistical errors larger than if points were independent.

The proposal density can be (almost) anything, but choose so as to minimize autocorrelation. Often take proposal density symmetric: $q(\vec{\theta}; \vec{\theta}_0) = q(\vec{\theta}_0; \vec{\theta})$

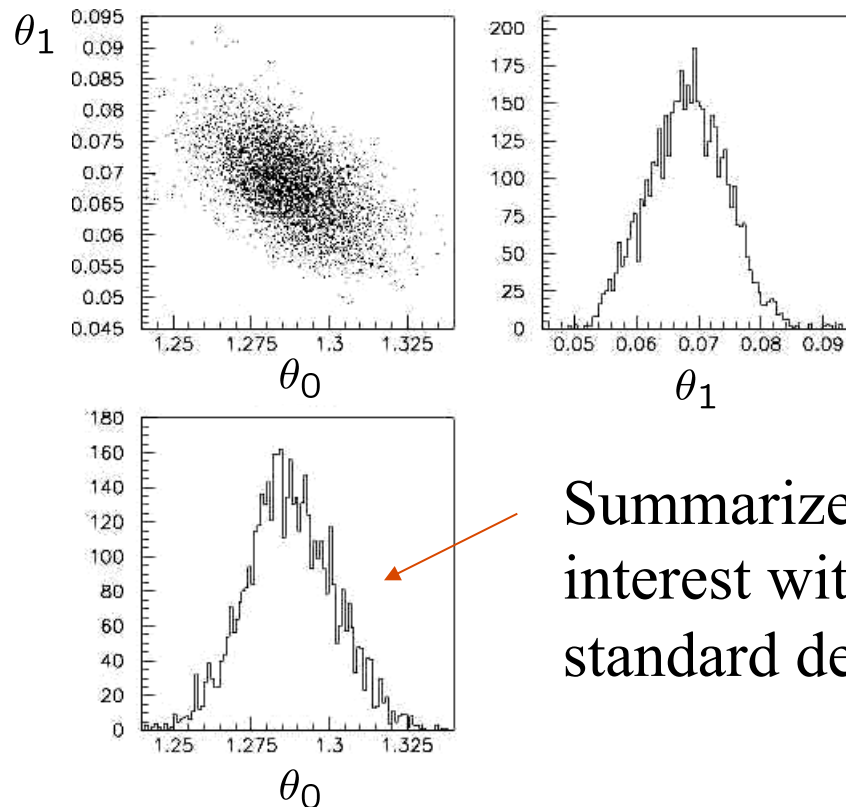
Test ratio is (*Metropolis-Hastings*): $\alpha = \min \left[1, \frac{p(\vec{\theta})}{p(\vec{\theta}_0)} \right]$

I.e. if the proposed step is to a point of higher $p(\vec{\theta})$, take it; if not, only take the step with probability $p(\vec{\theta})/p(\vec{\theta}_0)$.

If proposed step rejected, hop in place.

Example: posterior pdf from MCMC

Sample the posterior pdf from previous example with MCMC:



Summarize pdf of parameter of interest with, e.g., mean, median, standard deviation, etc.

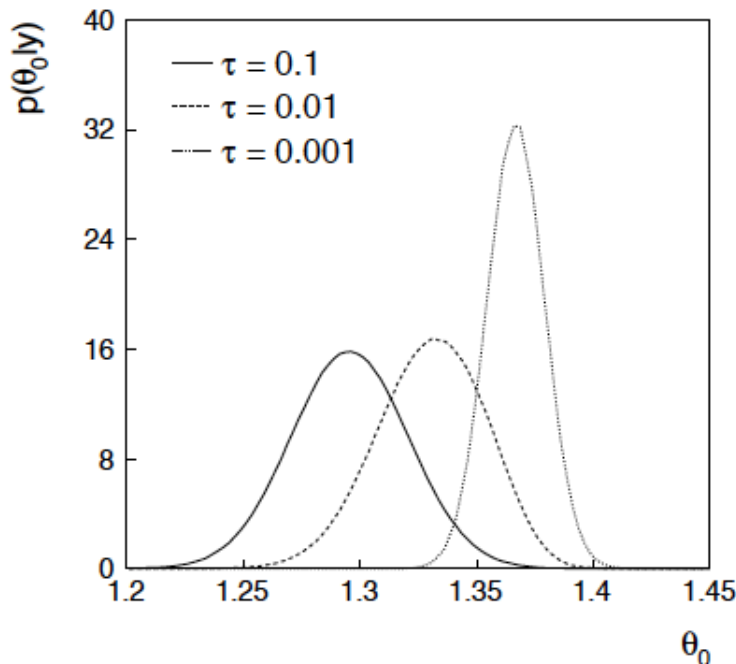
Although numerical values of answer here same as in frequentist case, interpretation is different (sometimes unimportant?)

Bayesian method with alternative priors

Suppose we don't have a previous measurement of θ_1 but rather, e.g., a theorist says it should be positive and not too much greater than 0.1 "or so", i.e., something like

$$\pi_1(\theta_1) = \frac{1}{\tau} e^{-\theta_1/\tau}, \quad \theta_1 \geq 0, \quad \tau = 0.1 .$$

From this we obtain (numerically) the posterior pdf for θ_0 :



This summarizes all knowledge about θ_0 .

Look also at result from variety of priors.

Expected discovery significance for counting experiment with background uncertainty

I. Discovery sensitivity for counting experiment with b known:

(a) $\frac{s}{\sqrt{b}}$

(b) Profile likelihood ratio test & Asimov: $\sqrt{2 \left((s + b) \ln \left(1 + \frac{s}{b} \right) - s \right)}$

II. Discovery sensitivity with uncertainty in b , σ_b :

(a) $\frac{s}{\sqrt{b + \sigma_b^2}}$

(b) Profile likelihood ratio test & Asimov:

$$\left[2 \left((s + b) \ln \left[\frac{(s + b)(b + \sigma_b^2)}{b^2 + (s + b)\sigma_b^2} \right] - \frac{b^2}{\sigma_b^2} \ln \left[1 + \frac{\sigma_b^2 s}{b(b + \sigma_b^2)} \right] \right) \right]^{1/2}$$

Counting experiment with known background

Count a number of events $n \sim \text{Poisson}(s+b)$, where

s = expected number of events from signal,

b = expected number of background events.

To test for discovery of signal compute p -value of $s = 0$ hypothesis,

$$p = P(n \geq n_{\text{obs}}|b) = \sum_{n=n_{\text{obs}}}^{\infty} \frac{b^n}{n!} e^{-b} = 1 - F_{\chi^2}(2b; 2n_{\text{obs}})$$

Usually convert to equivalent significance: $Z = \Phi^{-1}(1 - p)$
where Φ is the standard Gaussian cumulative distribution, e.g.,
 $Z > 5$ (a 5 sigma effect) means $p < 2.9 \times 10^{-7}$.

To characterize sensitivity to discovery, give expected (mean or median) Z under assumption of a given s .

s/\sqrt{b} for expected discovery significance

For large $s + b$, $n \rightarrow x \sim \text{Gaussian}(\mu, \sigma)$, $\mu = s + b$, $\sigma = \sqrt{s + b}$.

For observed value x_{obs} , p -value of $s = 0$ is $\text{Prob}(x > x_{\text{obs}} | s = 0)$,:

$$p_0 = 1 - \Phi\left(\frac{x_{\text{obs}} - b}{\sqrt{b}}\right)$$

Significance for rejecting $s = 0$ is therefore

$$Z_0 = \Phi^{-1}(1 - p_0) = \frac{x_{\text{obs}} - b}{\sqrt{b}}$$

Expected (median) significance assuming signal rate s is

$$\text{median}[Z_0 | s + b] = \frac{s}{\sqrt{b}}$$

Better approximation for significance

Poisson likelihood for parameter s is

$$L(s) = \frac{(s+b)^n}{n!} e^{-(s+b)}$$

For now
no nuisance
params.

To test for discovery use profile likelihood ratio:

$$q_0 = \begin{cases} -2 \ln \lambda(0) & \hat{s} \geq 0, \\ 0 & \hat{s} < 0. \end{cases} \quad \lambda(s) = \frac{L(s, \hat{\theta}(s))}{L(\hat{s}, \hat{\theta})}$$

So the likelihood ratio statistic for testing $s = 0$ is

$$q_0 = -2 \ln \frac{L(0)}{L(\hat{s})} = 2 \left(n \ln \frac{n}{b} + b - n \right) \quad \text{for } n > b, \quad 0 \text{ otherwise}$$

Approximate Poisson significance (continued)

For sufficiently large $s + b$, (use Wilks' theorem),

$$Z = \sqrt{2 \left(n \ln \frac{n}{b} + b - n \right)} \quad \text{for } n > b \text{ and } Z = 0 \text{ otherwise.}$$

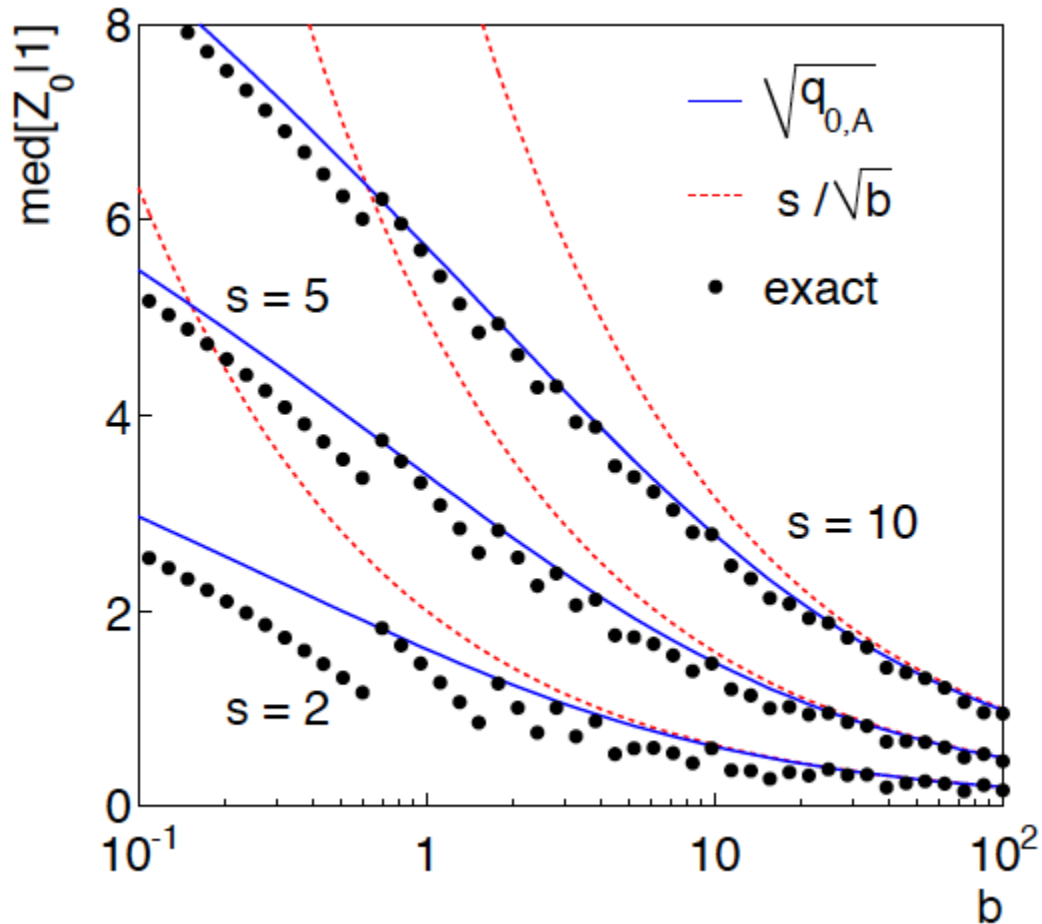
To find $\text{median}[Z|s]$, let $n \rightarrow s + b$ (i.e., the Asimov data set):

$$Z_A = \sqrt{2 \left((s + b) \ln \left(1 + \frac{s}{b} \right) - s \right)}$$

This reduces to s/\sqrt{b} for $s \ll b$.

$n \sim \text{Poisson}(s+b)$, median significance,
assuming s , of the hypothesis $s = 0$

CCGV, EPJC 71 (2011) 1554, arXiv:1007.1727



“Exact” values from MC,
jumps due to discrete data.

Asimov $\sqrt{q_{0,A}}$ good approx.
for broad range of s, b .

s/\sqrt{b} only good for $s \ll b$.

Extending s/\sqrt{b} to case where b uncertain

The intuitive explanation of s/\sqrt{b} is that it compares the signal, s , to the standard deviation of n assuming no signal, \sqrt{b} .

Now suppose the value of b is uncertain, characterized by a standard deviation σ_b .

A reasonable guess is to replace \sqrt{b} by the quadratic sum of \sqrt{b} and σ_b , i.e.,

$$\text{med}[Z|s] = \frac{s}{\sqrt{b + \sigma_b^2}}$$

This has been used to optimize some analyses e.g. where σ_b cannot be neglected.

Profile likelihood with b uncertain

This is the well studied “on/off” problem: Cranmer 2005; Cousins, Linnemann, and Tucker 2008; Li and Ma 1983,...

Measure two Poisson distributed values:

$n \sim \text{Poisson}(s+b)$ (primary or “search” measurement)

$m \sim \text{Poisson}(\tau b)$ (control measurement, τ known)

The likelihood function is

$$L(s, b) = \frac{(s+b)^n}{n!} e^{-(s+b)} \frac{(\tau b)^m}{m!} e^{-\tau b}$$

Use this to construct profile likelihood ratio (b is nuisance parameter):

$$\lambda(0) = \frac{L(0, \hat{b}(0))}{L(\hat{s}, \hat{b})}$$

Ingredients for profile likelihood ratio

To construct profile likelihood ratio from this need estimators:

$$\hat{s} = n - m/\tau ,$$

$$\hat{b} = m/\tau ,$$

$$\hat{b}(s) = \frac{n + m - (1 + \tau)s + \sqrt{(n + m - (1 + \tau)s)^2 + 4(1 + \tau)sm}}{2(1 + \tau)} .$$

and in particular to test for discovery ($s = 0$),

$$\hat{b}(0) = \frac{n + m}{1 + \tau}$$

Asymptotic significance

Use profile likelihood ratio for q_0 , and then from this get discovery significance using asymptotic approximation (Wilks' theorem):

$$Z = \sqrt{q_0} \\ = \left[-2 \left(n \ln \left[\frac{n+m}{(1+\tau)n} \right] + m \ln \left[\frac{\tau(n+m)}{(1+\tau)m} \right] \right) \right]^{1/2}$$

for $n > \hat{b}$ and $Z = 0$ otherwise.

Essentially same as in:

Robert D. Cousins, James T. Linnemann and Jordan Tucker, NIM A 595 (2008) 480–501; arXiv:physics/0702156.

Tipei Li and Yuqian Ma, Astrophysical Journal 272 (1983) 317–324.

Asimov approximation for median significance

To get median discovery significance, replace n , m by their expectation values assuming background-plus-signal model:

$$n \rightarrow s + b$$

$$m \rightarrow \tau b$$

$$Z_A = \left[-2 \left((s + b) \ln \left[\frac{s + (1 + \tau)b}{(1 + \tau)(s + b)} \right] + \tau b \ln \left[1 + \frac{s}{(1 + \tau)b} \right] \right) \right]^{1/2}$$

Or use the variance of $\hat{b} = m/\tau$, $V[\hat{b}] \equiv \sigma_b^2 = \frac{b}{\tau}$, to eliminate τ :

$$Z_A = \left[2 \left((s + b) \ln \left[\frac{(s + b)(b + \sigma_b^2)}{b^2 + (s + b)\sigma_b^2} \right] - \frac{b^2}{\sigma_b^2} \ln \left[1 + \frac{\sigma_b^2 s}{b(b + \sigma_b^2)} \right] \right) \right]^{1/2}$$

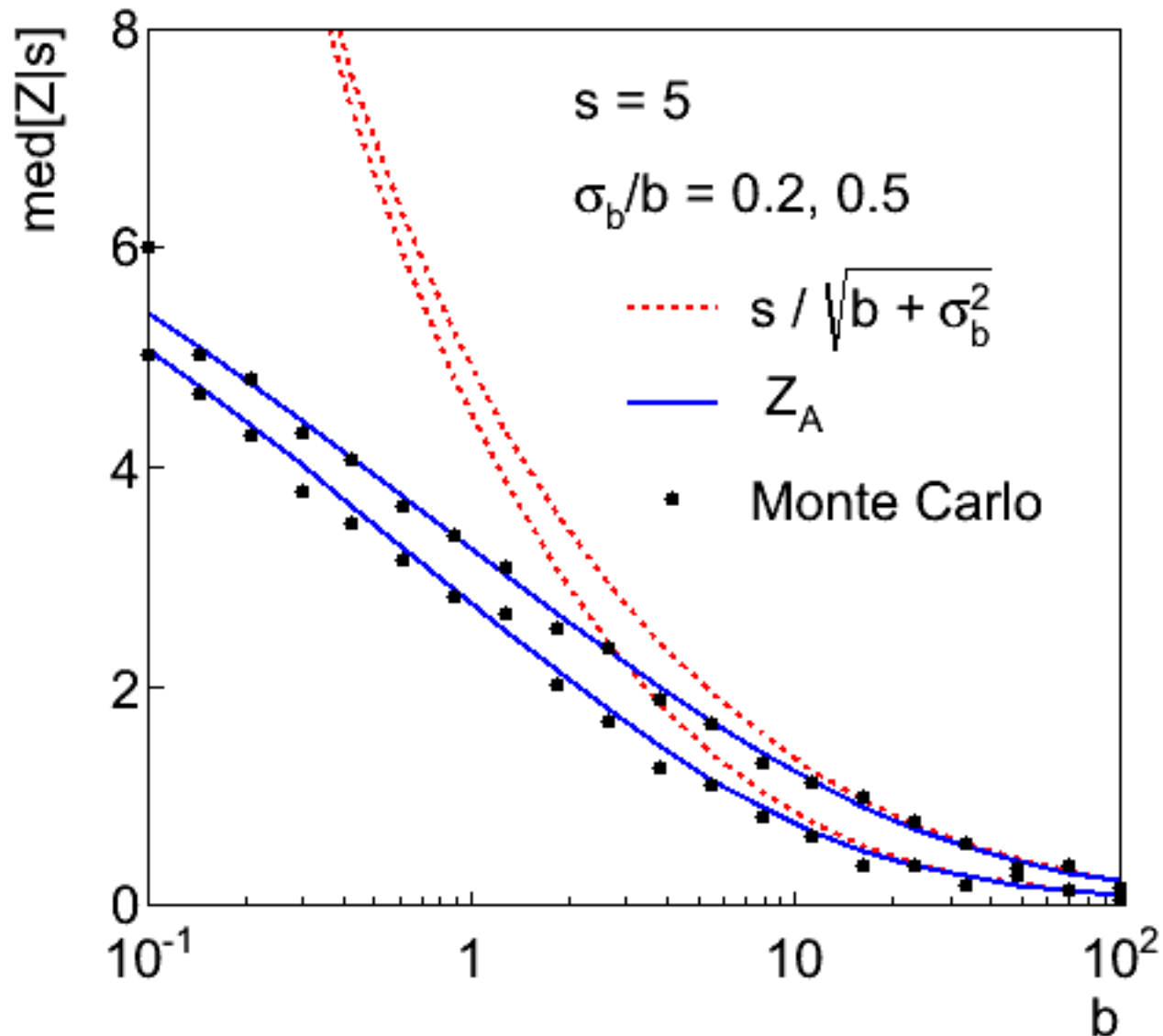
Limiting cases

Expanding the Asimov formula in powers of s/b and σ_b^2/b ($= 1/\tau$) gives

$$Z_A = \frac{s}{\sqrt{b + \sigma_b^2}} \left(1 + \mathcal{O}(s/b) + \mathcal{O}(\sigma_b^2/b) \right)$$

So the “intuitive” formula can be justified as a limiting case of the significance from the profile likelihood ratio test evaluated with the Asimov data set.

Testing the formulae: $s = 5$



Using sensitivity to optimize a cut

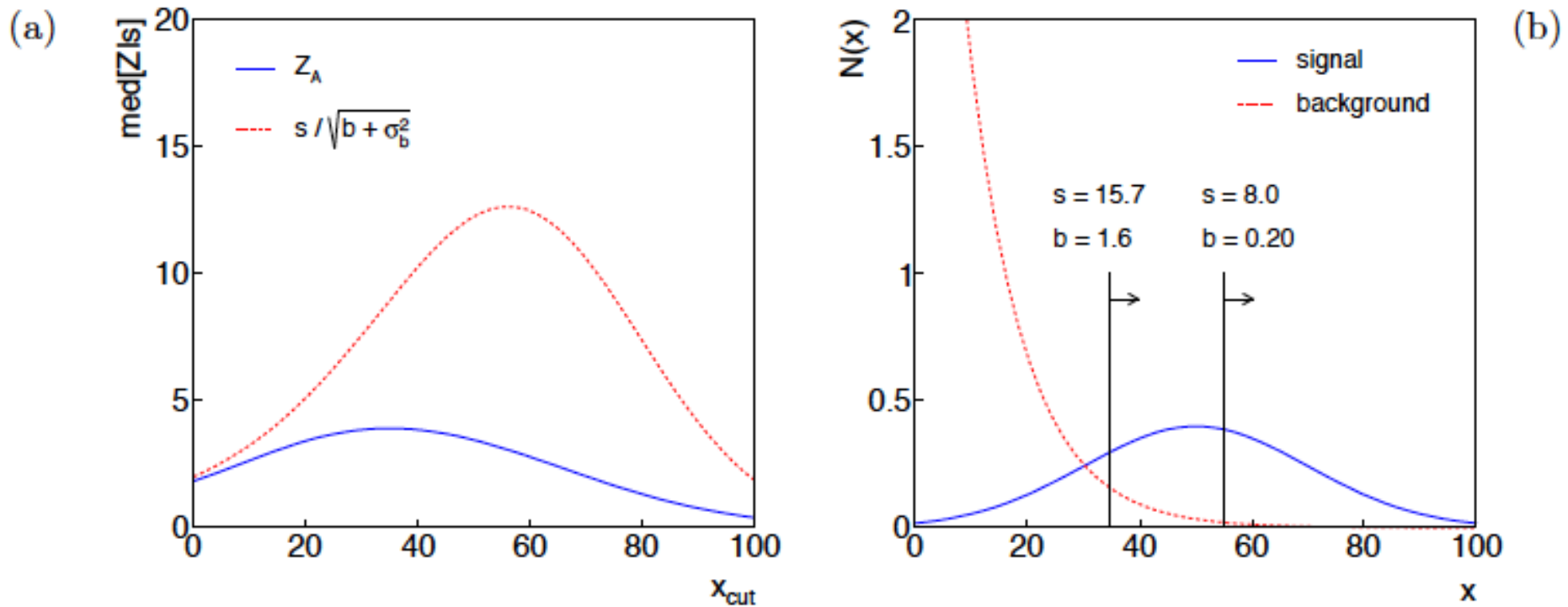


Figure 1: (a) The expected significance as a function of the cut value x_{cut} ; (b) the distributions of signal and background with the optimal cut value indicated.

Extra slides

Resources on multivariate methods

C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006

T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2nd ed., Springer, 2009

R. Duda, P. Hart, D. Stork, Pattern Classification, 2nd ed., Wiley, 2001

A. Webb, Statistical Pattern Recognition, 2nd ed., Wiley, 2002.

Ilya Narsky and Frank C. Porter, *Statistical Analysis Techniques in Particle Physics*, Wiley, 2014.

朱永生 (编著), 实验数据多元统计分析, 科学出版社, 北京, 2009。

Software

Rapidly growing area of development – two important resources:

TMVA, Höcker, Stelzer, Tegenfeldt, Voss, Voss, [physics/0703039](#)

From `tmva.sourceforge.net`, also distributed with ROOT

Variety of classifiers

Good manual, widely used in HEP

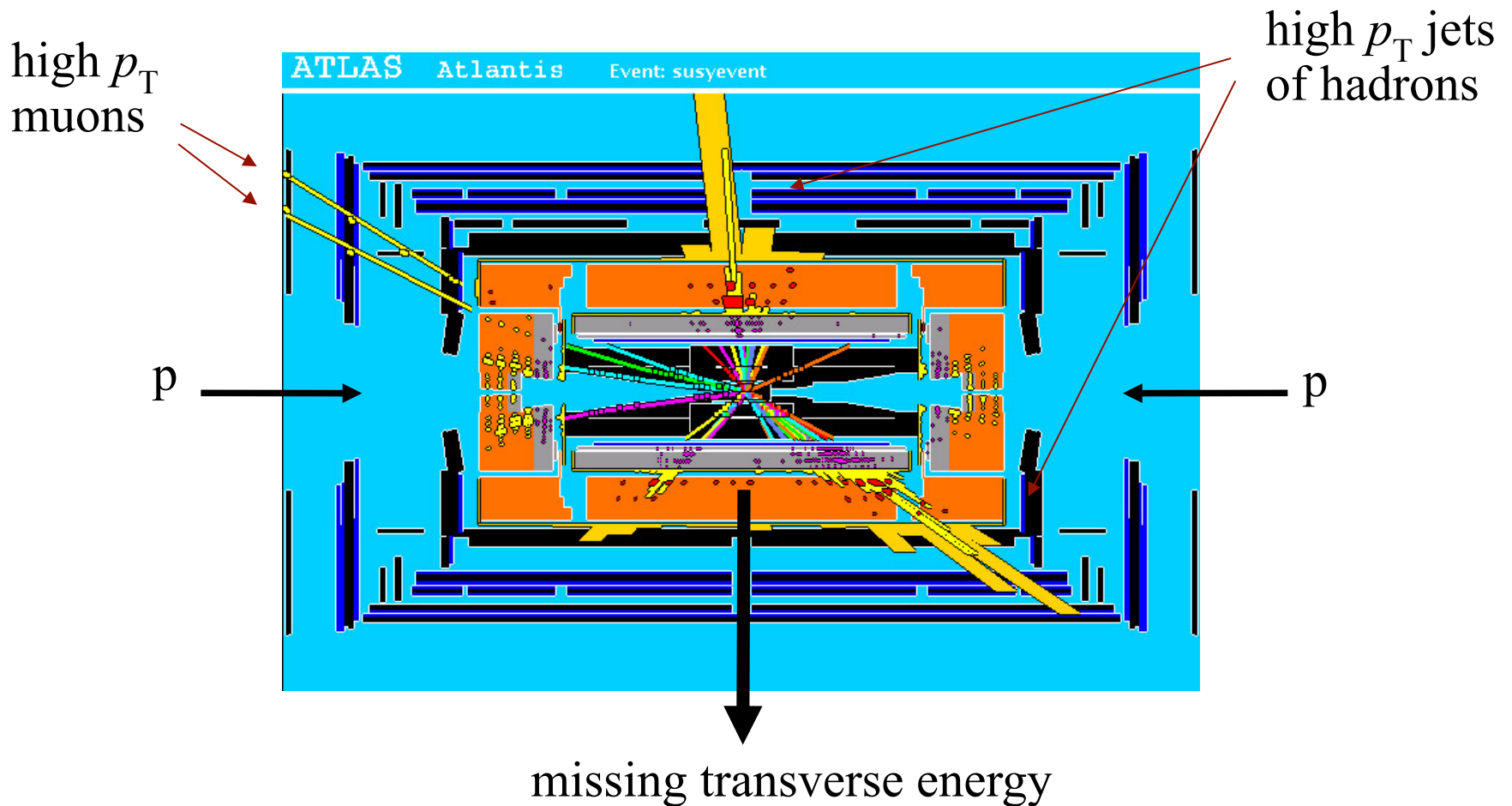
scikit-learn

Python-based tools for Machine Learning

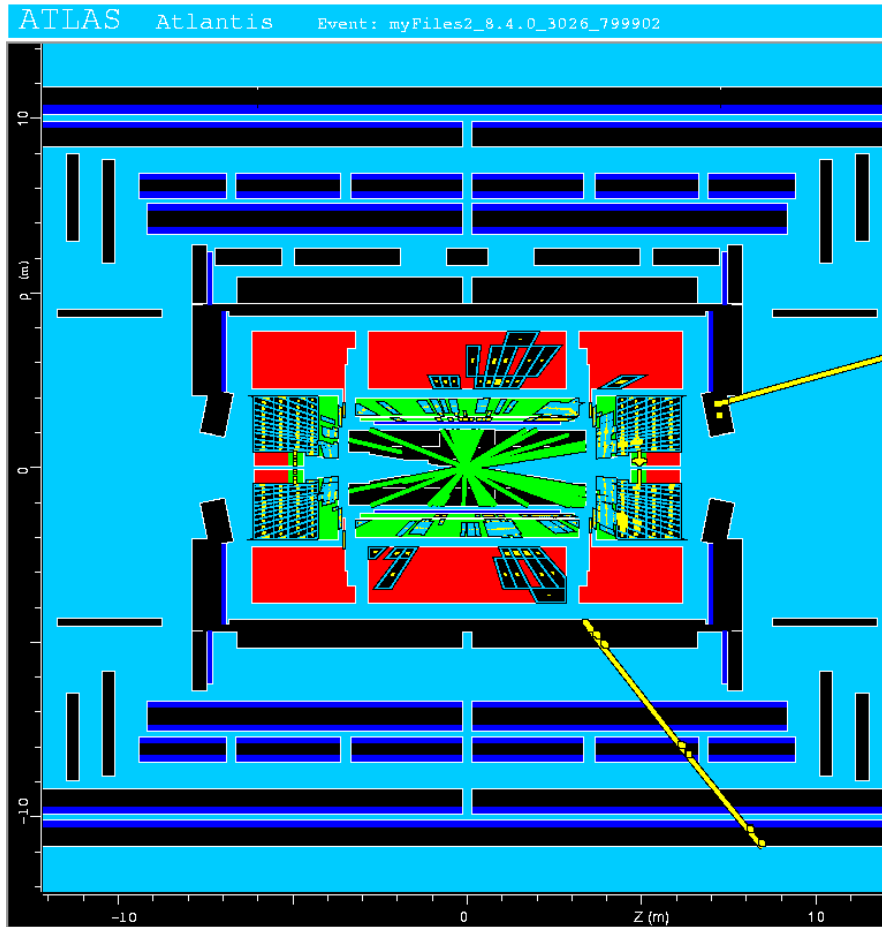
`scikit-learn.org`

Large user community

A simulated SUSY event in ATLAS



Background events



This event from Standard Model $t\bar{t}$ production also has high p_T jets and muons, and some missing transverse energy.

→ can easily mimic a SUSY event.

Defining a multivariate critical region

For each event, measure, e.g.,

$$x_1 = \text{missing energy}, x_2 = \text{electron } p_T, x_3 = \dots$$

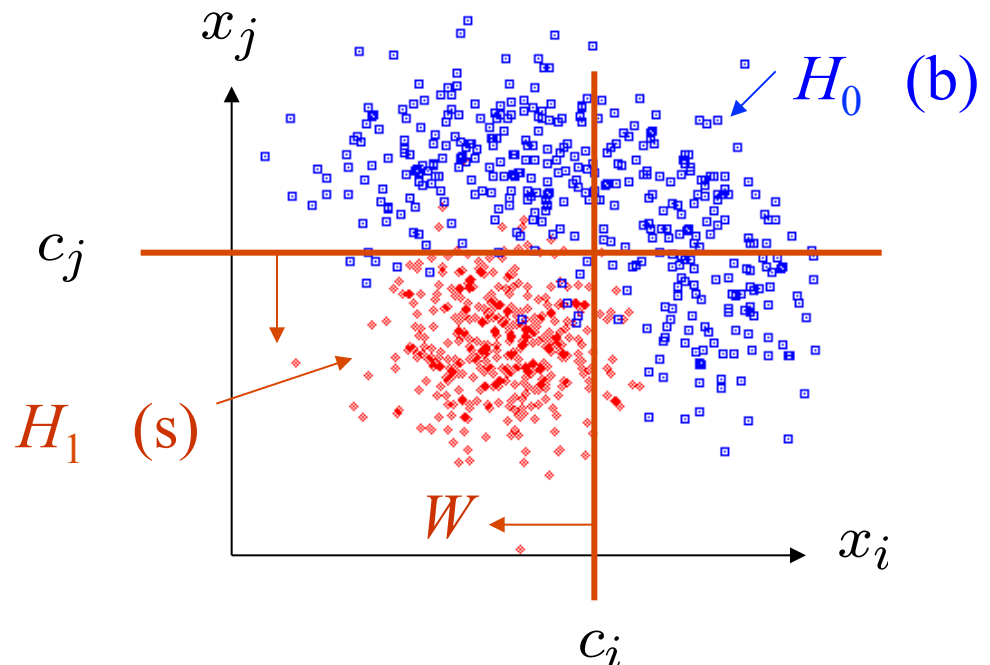
Each event is a point in n -dimensional \mathbf{x} -space; critical region is now defined by a ‘decision boundary’ in this space.

What is best way to determine the boundary?

Perhaps with ‘cuts’:

$$x_i < c_i$$

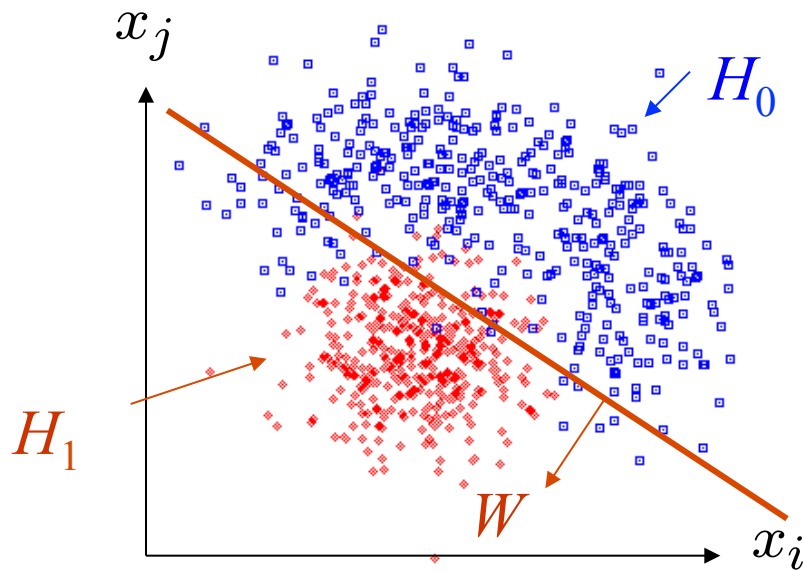
$$x_j < c_j$$



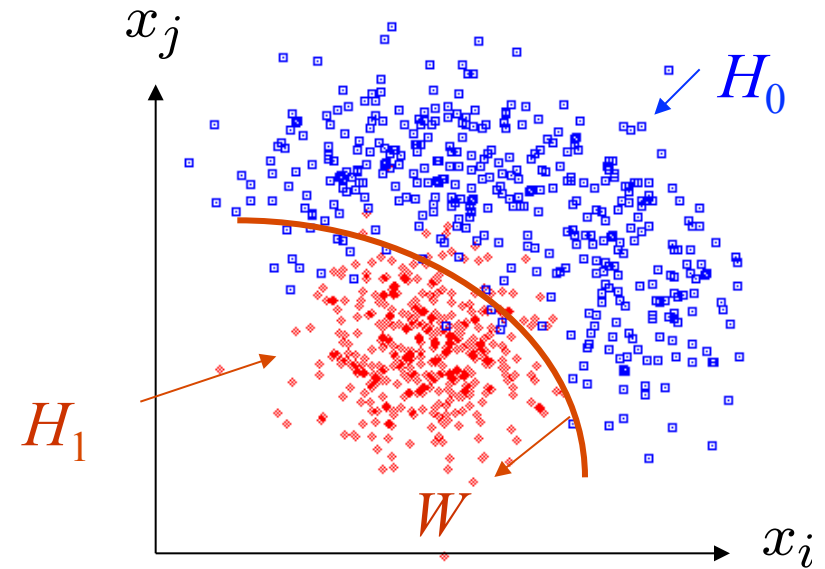
Other multivariate decision boundaries

Or maybe use some other sort of decision boundary:

linear



or nonlinear



Multivariate methods for finding optimal critical region have become a Big Industry (neural networks, boosted decision trees,...), benefitting from recent advances in Machine Learning.

Test statistics

The boundary of the critical region for an n -dimensional data space $\mathbf{x} = (x_1, \dots, x_n)$ can be defined by an equation of the form

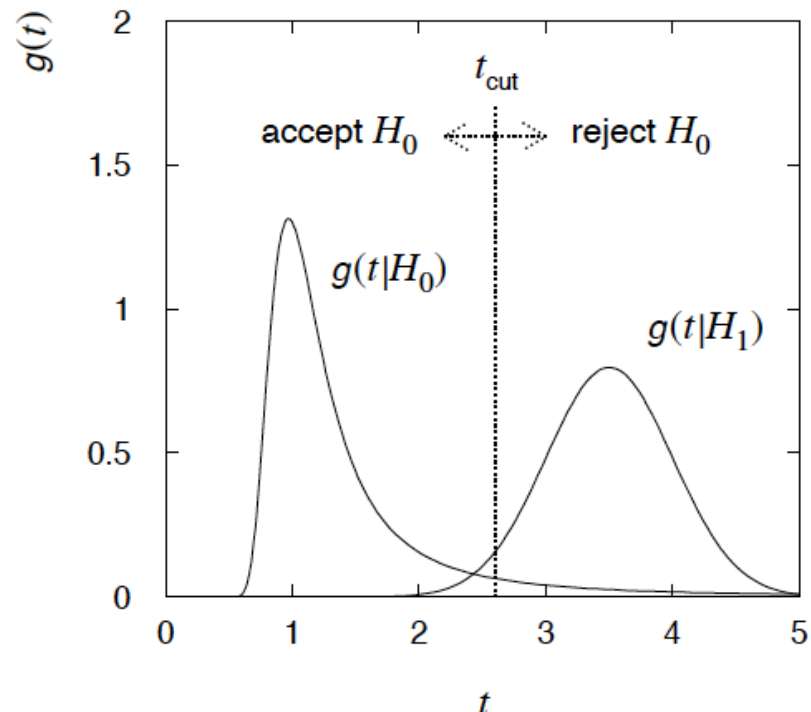
$$t(x_1, \dots, x_n) = t_{\text{cut}}$$

where $t(x_1, \dots, x_n)$ is a scalar **test statistic**.

We can work out the pdfs $g(t|H_0)$, $g(t|H_1)$, \dots

Decision boundary is now a single 'cut' on t , defining the critical region.

So for an n -dimensional problem we have a corresponding 1-d problem.



Test statistic based on likelihood ratio

How can we choose a test's critical region in an 'optimal way'?

Neyman-Pearson lemma states:

To get the highest power for a given significance level in a test of H_0 , (background) versus H_1 , (signal) the critical region should have

$$\frac{f(\mathbf{x}|H_1)}{f(\mathbf{x}|H_0)} > c$$

inside the region, and $\leq c$ outside, where c is a constant chosen to give a test of the desired size.

Equivalently, optimal scalar test statistic is

$$t(\mathbf{x}) = \frac{f(\mathbf{x}|H_1)}{f(\mathbf{x}|H_0)}$$

N.B. any monotonic function of this is leads to the same test.

Neyman-Pearson doesn't usually help

We usually don't have explicit formulae for the pdfs $f(\mathbf{x}|s)$, $f(\mathbf{x}|b)$, so for a given \mathbf{x} we can't evaluate the likelihood ratio

$$t(\mathbf{x}) = \frac{f(\mathbf{x}|s)}{f(\mathbf{x}|b)}$$

Instead we may have Monte Carlo models for signal and background processes, so we can produce simulated data:

generate $\mathbf{x} \sim f(\mathbf{x}|s)$ \rightarrow $\mathbf{x}_1, \dots, \mathbf{x}_N$

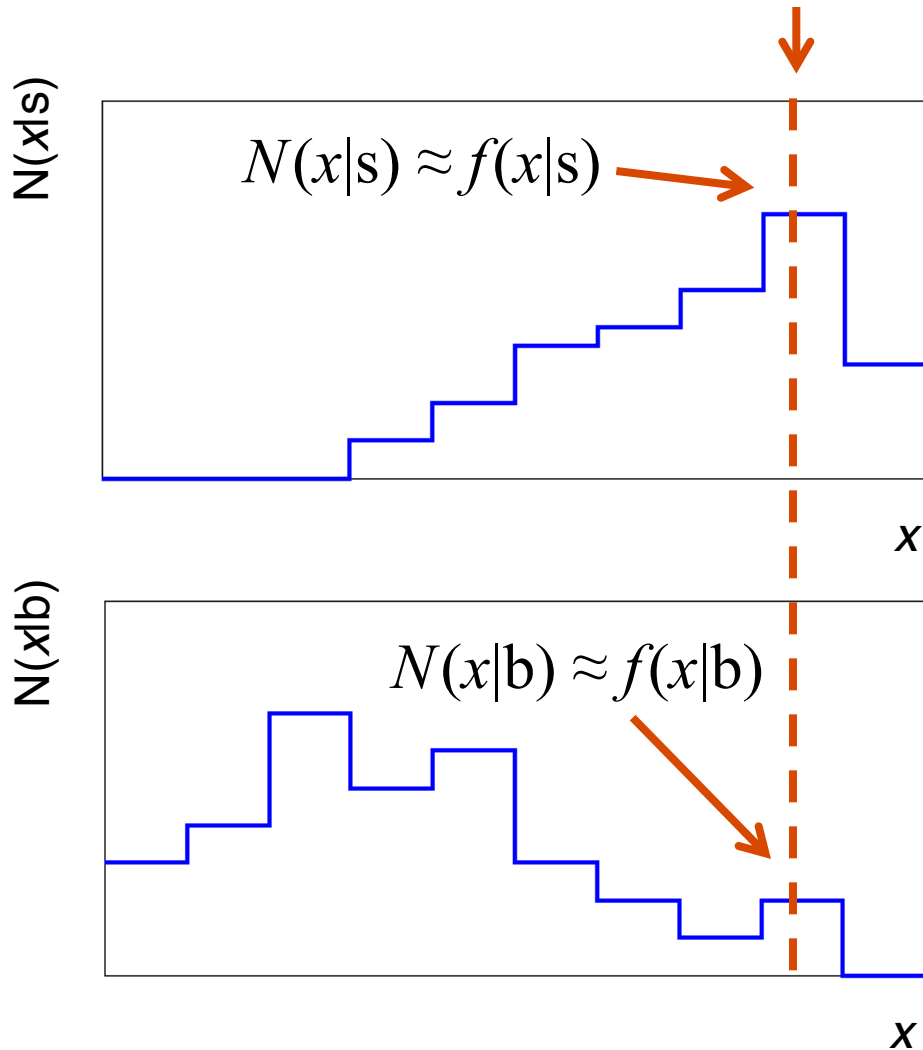
generate $\mathbf{x} \sim f(\mathbf{x}|b)$ \rightarrow $\mathbf{x}_1, \dots, \mathbf{x}_N$

This gives samples of “training data” with events of known type.

Can be expensive (1 fully simulated LHC event \sim 1 CPU minute).

Approximate LR from histograms

Want $t(x) = f(x|s)/f(x|b)$ for x here



One possibility is to generate MC data and construct histograms for both signal and background.

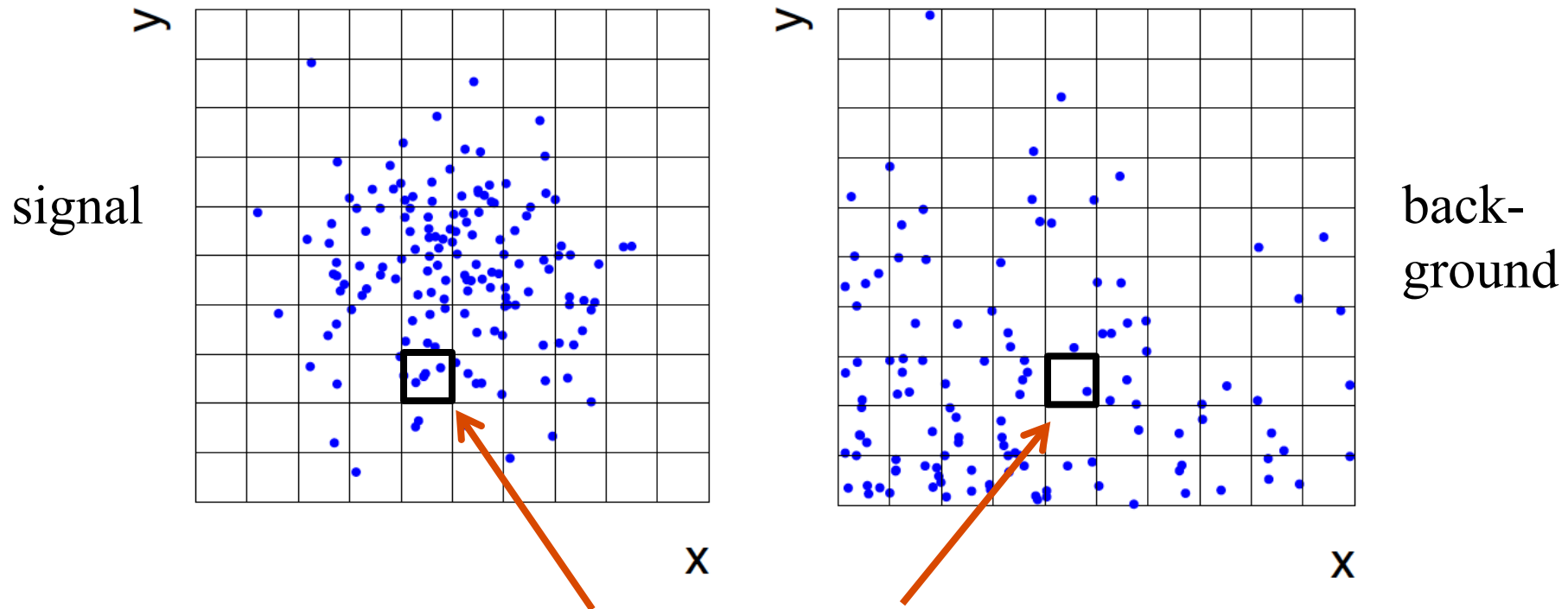
Use (normalized) histogram values to approximate LR:

$$t(x) \approx \frac{N(x|s)}{N(x|b)}$$

Can work well for single variable.

Approximate LR from 2D-histograms

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



Approximate pdfs using $N(x,y|s)$, $N(x,y|b)$ 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.

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

Strategies for multivariate analysis

Neyman-Pearson lemma gives optimal answer, but cannot be used directly, because we usually don't have $f(\mathbf{x}|\mathbf{s})$, $f(\mathbf{x}|\mathbf{b})$.

Histogram method with M bins for n variables requires that we estimate M^n parameters (the values of the pdfs in each cell), so this is rarely practical.

A compromise solution is to assume a certain functional form for the test statistic $t(\mathbf{x})$ with fewer parameters; determine them (using MC) to give best separation between signal and background.

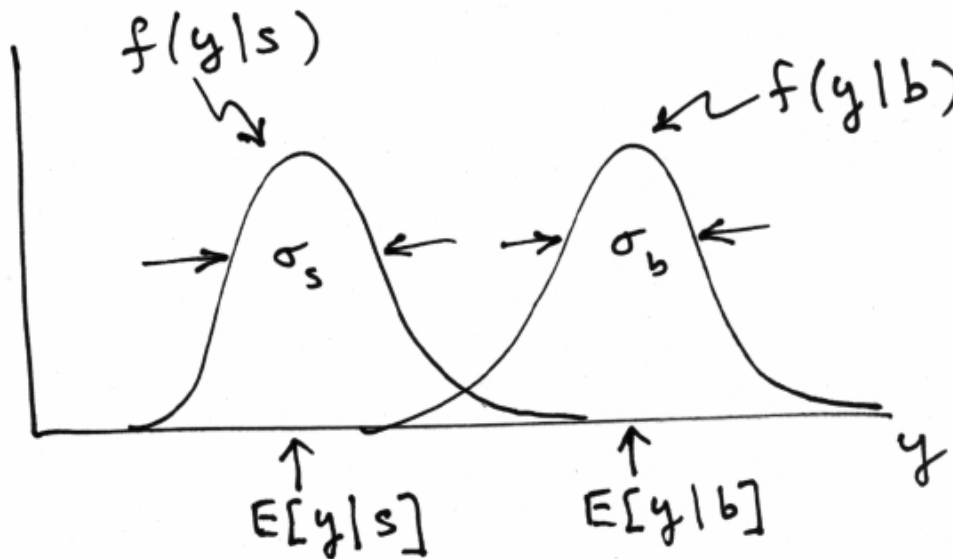
Alternatively, try to estimate the probability densities $f(\mathbf{x}|\mathbf{s})$ and $f(\mathbf{x}|\mathbf{b})$ (with something better than histograms) and use the estimated pdfs to construct an approximate likelihood ratio.

Linear test statistic

Suppose there are n input variables: $\mathbf{x} = (x_1, \dots, x_n)$.

Consider a linear function:
$$y(\mathbf{x}) = \sum_{i=1}^n w_i x_i$$

For a given choice of the coefficients $\mathbf{w} = (w_1, \dots, w_n)$ we will get pdfs $f(y|s)$ and $f(y|b)$:



Linear test statistic

Fisher: to get large difference between means and small widths for $f(y|s)$ and $f(y|b)$, maximize the difference squared of the expectation values divided by the sum of the variances:

$$J(\mathbf{w}) = \frac{(E[y|s] - E[y|b])^2}{V[y|s] + V[y|b]}$$

Setting $\partial J / \partial w_i = 0$ gives for $\mathbf{w} = (w_1, \dots, w_n)$:

$$\mathbf{w} \propto W^{-1}(\boldsymbol{\mu}_b - \boldsymbol{\mu}_s)$$

$$W_{ij} = \text{cov}[x_i, x_j|s] + \text{cov}[x_i, x_j|b]$$

$$\mu_{i,s} = E[x_i|s], \quad \mu_{i,b} = E[x_i|b]$$

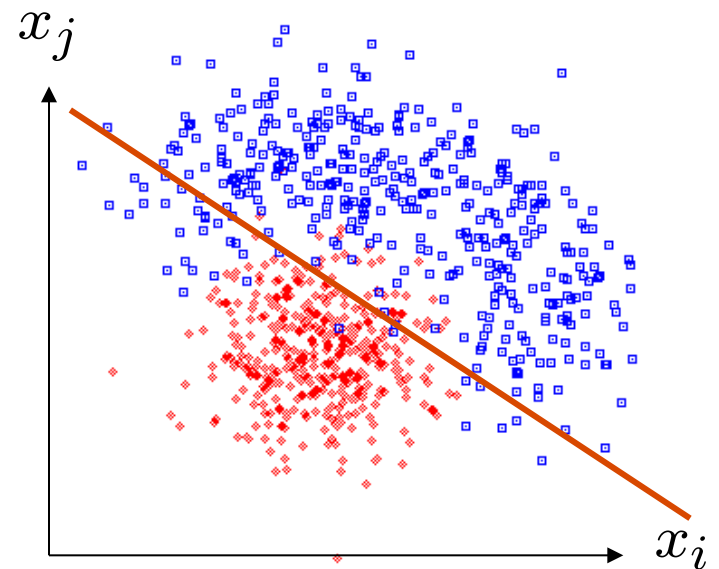
The Fisher discriminant

The resulting coefficients w_i define a Fisher discriminant.

Coefficients defined up to multiplicative constant; can also add arbitrary offset, i.e., usually define test statistic as

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^n w_i x_i$$

Boundaries of the test's critical region are surfaces of constant $y(\mathbf{x})$, here linear (hyperplanes):



Fisher discriminant for Gaussian data

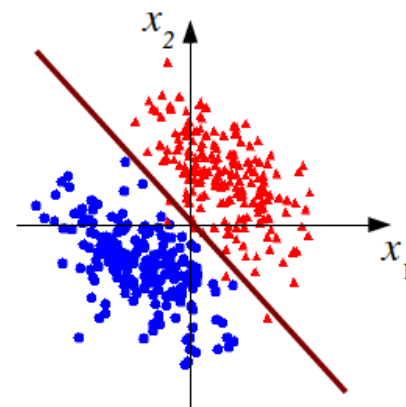
Suppose the pdfs of the input variables, $f(\mathbf{x}|\mathbf{s})$ and $f(\mathbf{x}|\mathbf{b})$, are both multivariate Gaussians with same covariance but different means:

$$f(\mathbf{x}|\mathbf{s}) = \text{Gauss}(\boldsymbol{\mu}_s, V)$$

$$f(\mathbf{x}|\mathbf{b}) = \text{Gauss}(\boldsymbol{\mu}_b, V)$$

Same covariance

$$V_{ij} = \text{cov}[x_i, x_j]$$



In this case it can be shown that the Fisher discriminant is

$$y(\mathbf{x}) \sim \ln \frac{f(\mathbf{x}|\mathbf{s})}{f(\mathbf{x}|\mathbf{b})}$$

i.e., it is a monotonic function of the likelihood ratio and thus leads to the same critical region. So in this case the Fisher discriminant provides an optimal statistical test.

Transformation of inputs

If the data are not Gaussian with equal covariance, a linear decision boundary is not optimal. But we can try to subject the data to a transformation

$$\phi_1(\vec{x}), \dots, \phi_m(\vec{x})$$

and then treat the ϕ_i as the new input variables. This is often called “feature space” and the ϕ_i are “basis functions”. The basis functions can be fixed or can contain adjustable parameters which we optimize with training data (cf. neural networks).

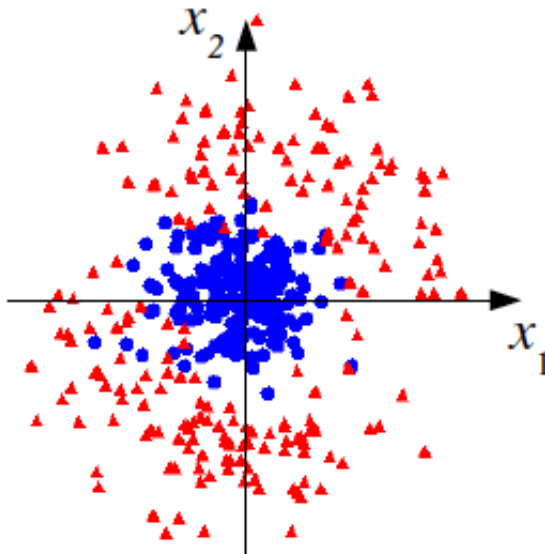
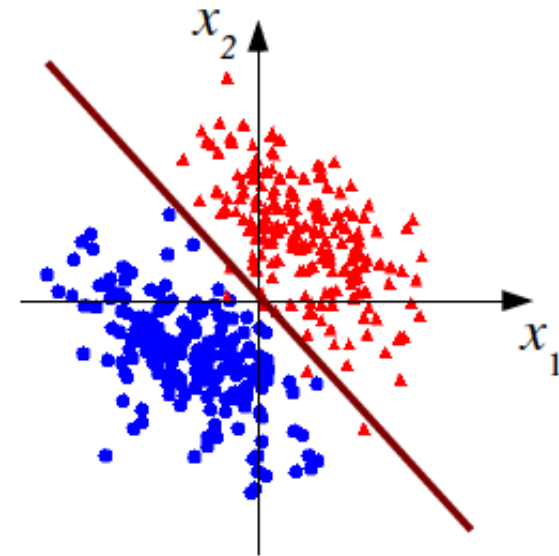
In other cases we will see that the basis functions only enter as dot products

$$\vec{\phi}(\vec{x}_i) \cdot \vec{\phi}(\vec{x}_j) = K(\vec{x}_i, \vec{x}_j)$$

and thus we will only need the “kernel function” $K(\mathbf{x}_i, \mathbf{x}_j)$

Linear decision boundaries

A linear decision boundary is only optimal when both classes follow multivariate Gaussians with equal covariances and different means.



For some other cases a linear boundary is almost useless.

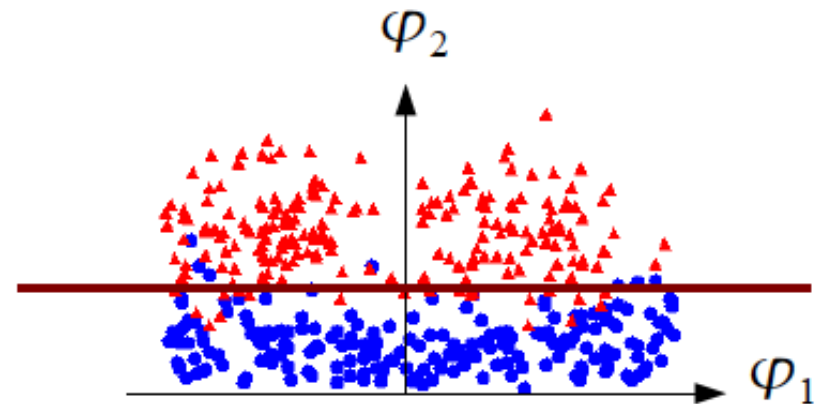
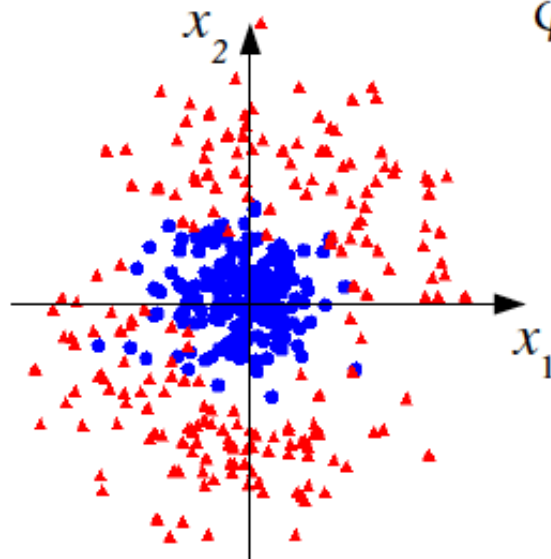
Nonlinear transformation of inputs

We can try to find a transformation, $x_1, \dots, x_n \rightarrow \varphi_1(\vec{x}), \dots, \varphi_m(\vec{x})$ so that the transformed “feature space” variables can be separated better by a linear boundary:

$$\varphi_1 = \tan^{-1}(x_2/x_1)$$

$$\varphi_2 = \sqrt{x_1^2 + x_2^2}$$

Here, guess fixed basis functions (no free parameters)



Neural networks

Neural networks originate from attempts to model neural processes (McCulloch and Pitts, 1943; Rosenblatt, 1962).

Widely used in many fields, and for many years the only “advanced” multivariate method popular in HEP.

We can view a neural network as a specific way of parametrizing the basis functions used to define the feature space transformation.

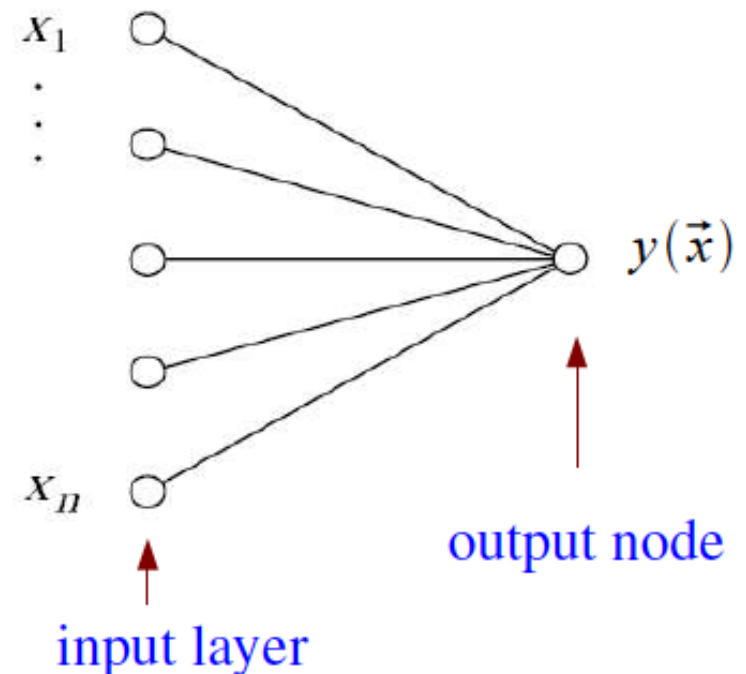
The training data are then used to adjust the parameters so that the resulting discriminant function has the best performance.

The single layer perceptron

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

where h is a nonlinear, monotonic **activation function**; we can use e.g. the logistic sigmoid $h(x) = (1 + e^{-x})^{-1}$.

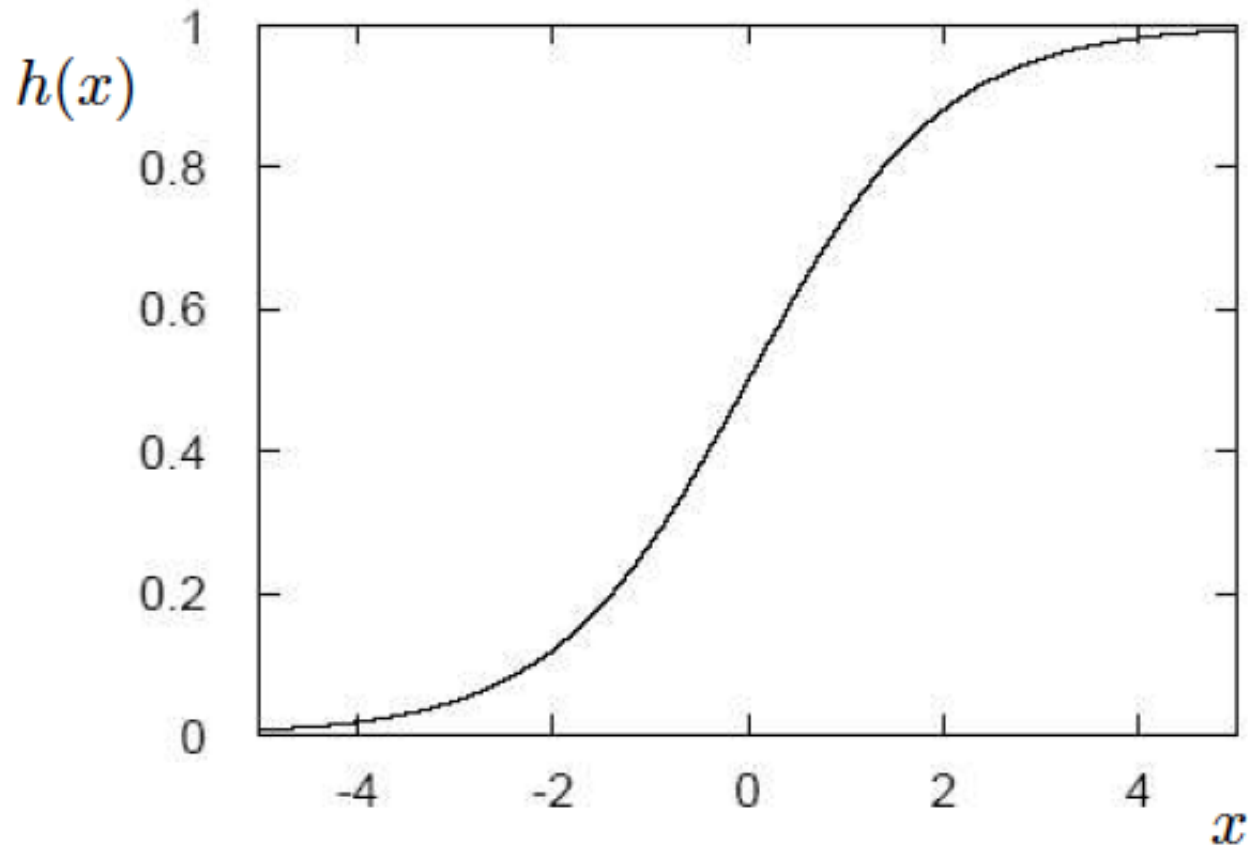
If the activation function is monotonic, the resulting $y(\mathbf{x})$ is equivalent to the original linear discriminant. This is an example of a “generalized linear model” called the **single layer perceptron**.



The activation function

For activation function $h(\cdot)$ often use logistic sigmoid:

$$h(x) = \frac{1}{1 + e^{-x}}$$



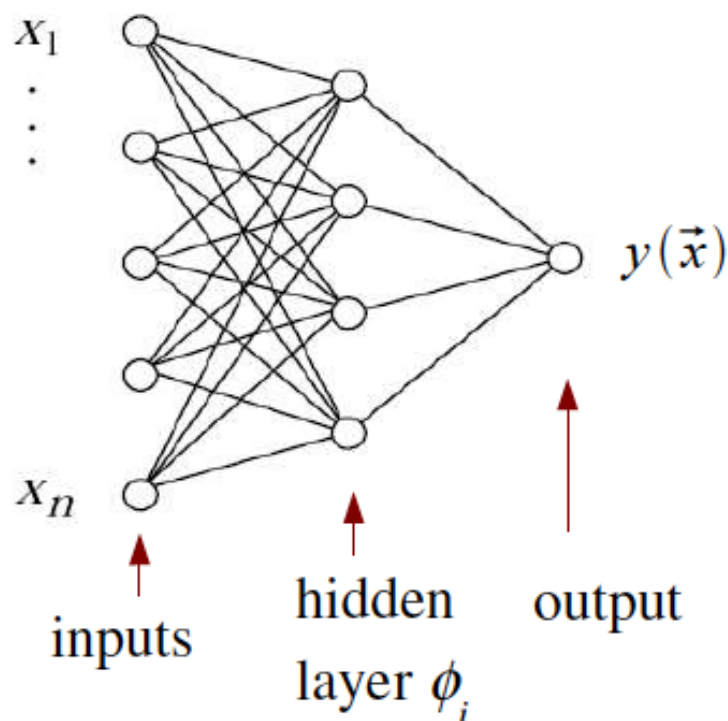
The multilayer perceptron

Now use this idea to define not only the output $y(\mathbf{x})$, but also the set of transformed inputs $\varphi_1(\vec{x}), \dots, \varphi_m(\vec{x})$ that form a “hidden layer”:

Superscript for weights indicates layer number

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

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



This is the **multilayer perceptron**, our basic neural network model; straightforward to generalize to multiple hidden layers.

Network architecture

Theorem: An MLP with a single hidden layer having a sufficiently large number of nodes can approximate arbitrarily well the optimal decision boundary.

Holds for any continuous non-polynomial activation function

Leshno, Lin, Pinkus and Schocken (1993) *Neural Networks* 6, 861-867

However, the number of required nodes may be very large; cannot train well with finite samples of training data.

Recent advances in *Deep Neural Networks* have shown important advantages in having multiple hidden layers.

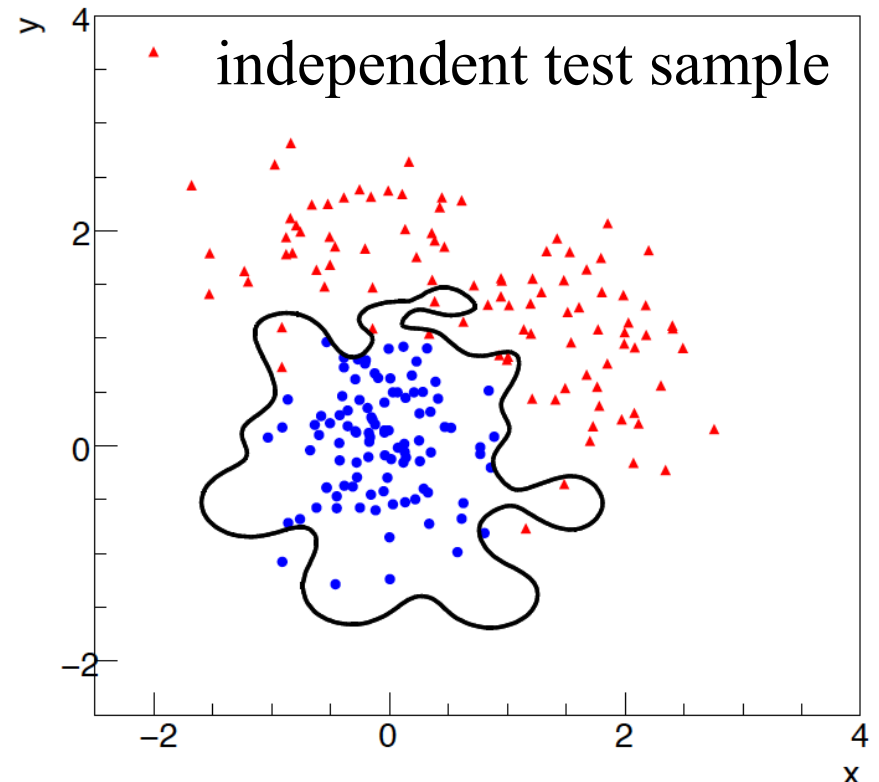
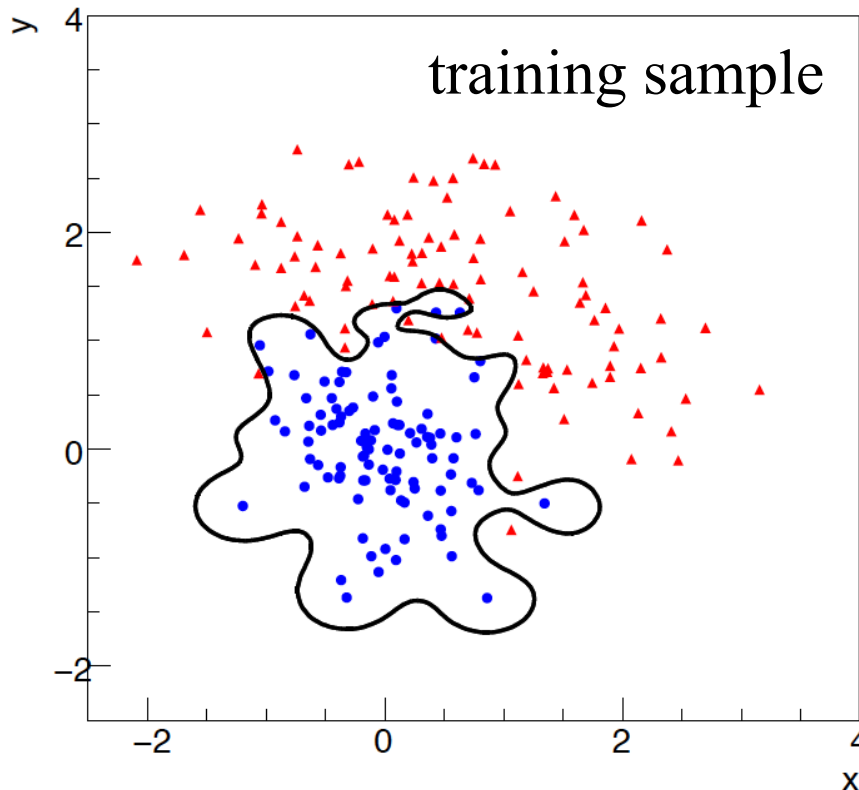
For a particle physics application of Deep Learning, see e.g.

Baldi, Sadowski and Whiteson, *Nature Communications* 5 (2014); arXiv:1402.4735.

Overtraining

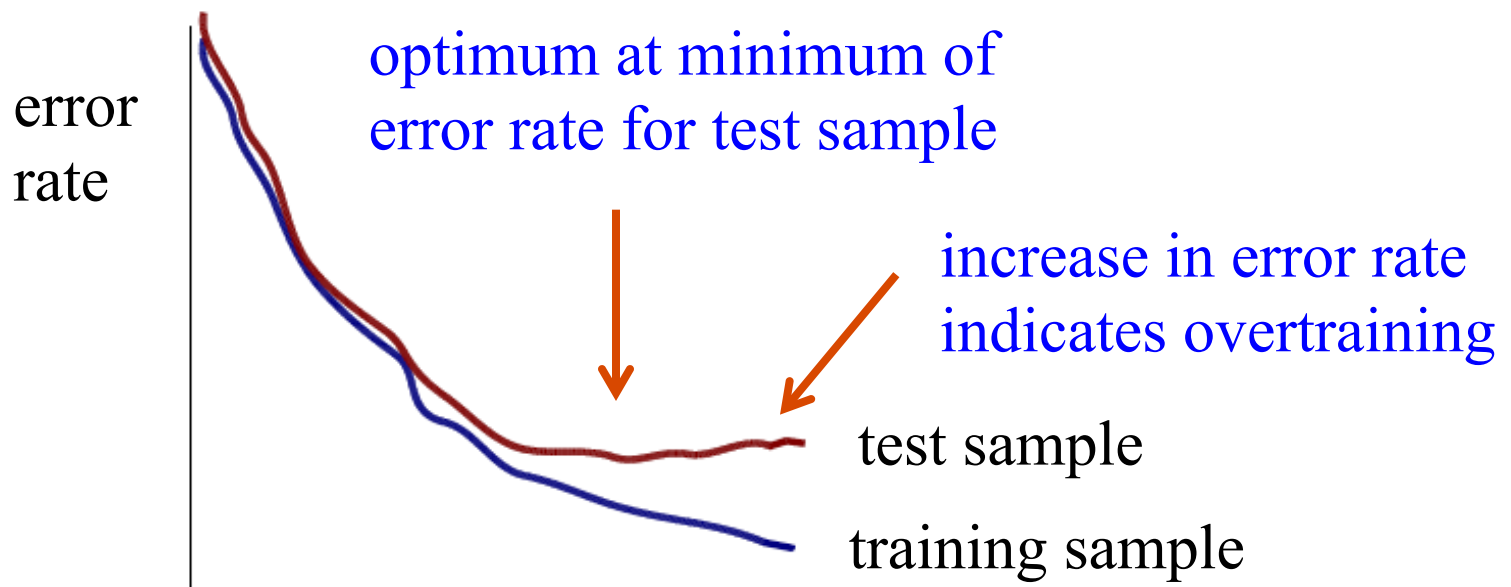
Including more parameters in a classifier makes its decision boundary increasingly flexible, e.g., more nodes/layers for a neural network.

A “flexible” classifier may conform too closely to the training points; the same boundary will not perform well on an independent test data sample (→ “overtraining”).



Monitoring overtraining

If we monitor the fraction of misclassified events (or similar, e.g., error function $E(\boldsymbol{w})$) for test and training samples, it will usually decrease for both as the boundary is made more flexible:

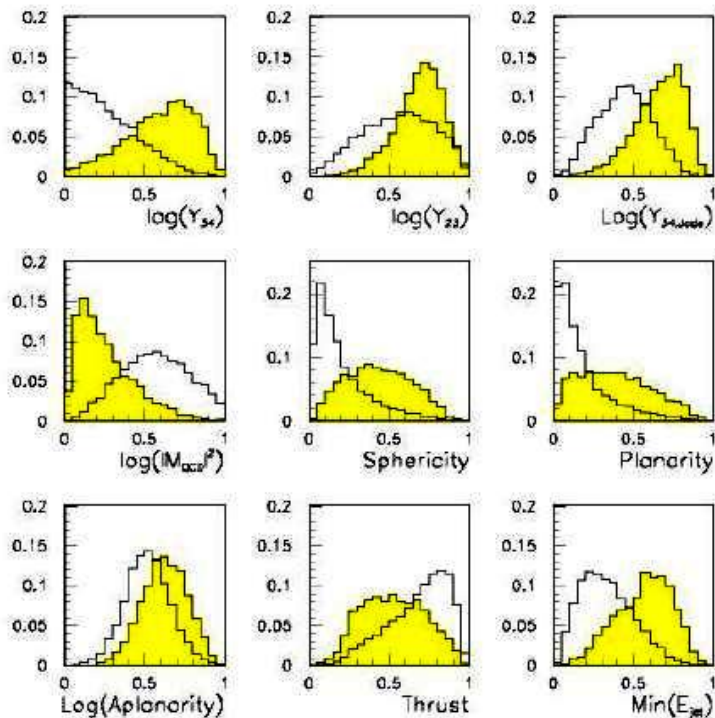


flexibility (e.g., number of nodes/layers in MLP)

Neural network example from LEP II

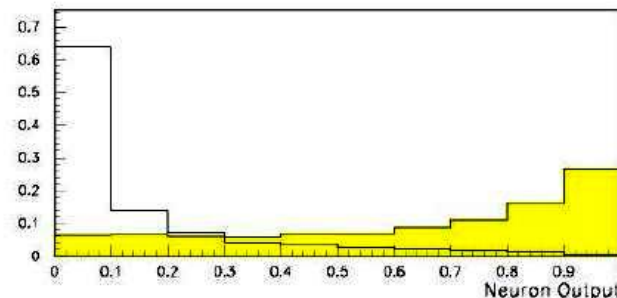
Signal: $e^+e^- \rightarrow W^+W^-$ (often 4 well separated hadron jets)

Background: $e^+e^- \rightarrow q\bar{q}g$ (4 less well separated hadron jets)



← input variables based on jet structure, event shape, ...
none by itself gives much separation.

Neural network output:



(Garrido, Juste and Martinez, ALEPH 96-144)

Summary on multivariate methods

Particle physics has used several multivariate methods for many years:

- linear (Fisher) discriminant
- neural networks
- naive Bayes

and has in recent years started to use a few more:

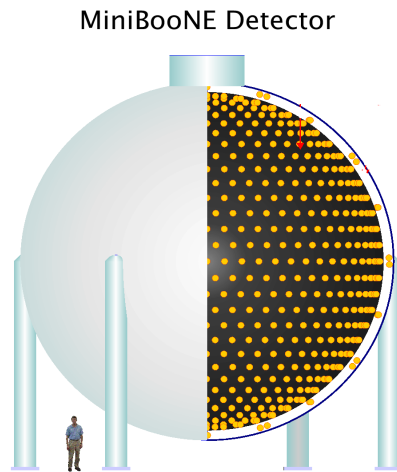
- boosted decision trees
- support vector machines
- kernel density estimation
- k -nearest neighbour

The emphasis is often on controlling systematic uncertainties between the modeled training data and Nature to avoid false discovery.

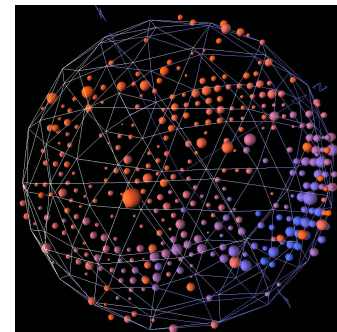
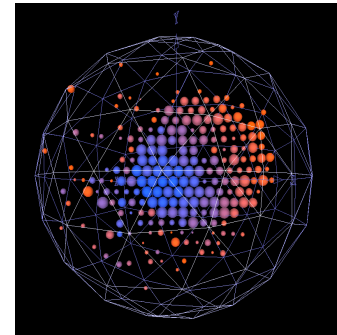
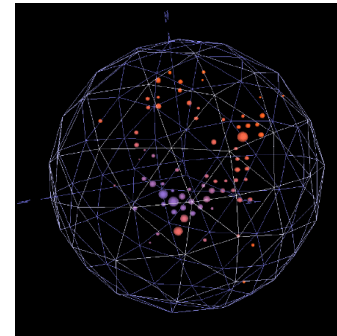
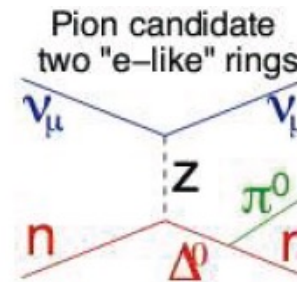
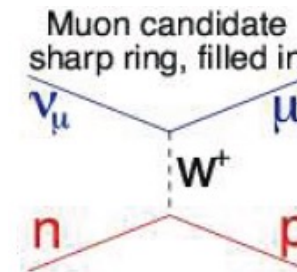
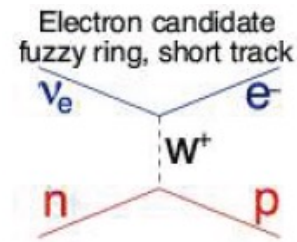
Although many classifier outputs are "black boxes", a discovery at 5σ significance with a sophisticated (opaque) method will win the competition if backed up by, say, 4σ evidence from a cut-based method.

Particle i.d. in MiniBooNE

Detector is a 12-m diameter tank of mineral oil exposed to a beam of neutrinos and viewed by 1520 photomultiplier tubes:



Search for ν_μ to ν_e oscillations required particle i.d. using information from the PMTs.



H.J. Yang, MiniBooNE PID, DNP06

Decision trees

Out of all the input variables, find the one for which with a single cut gives best improvement in signal purity:

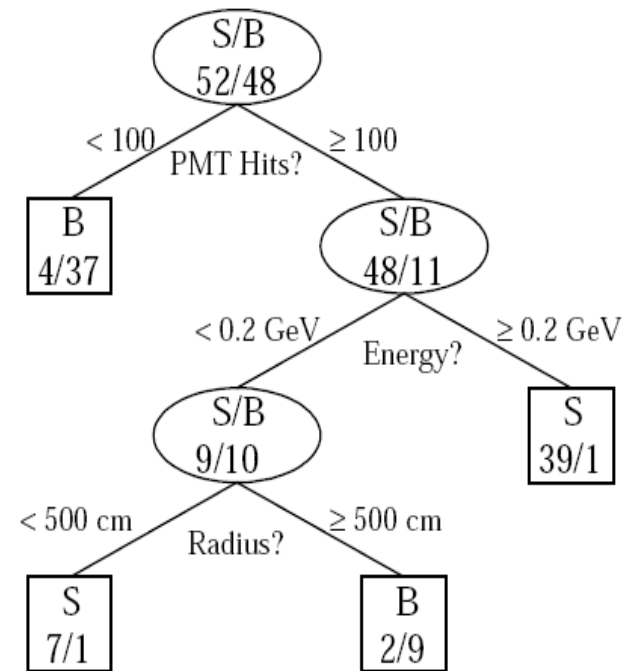
$$P = \frac{\sum_{\text{signal}} w_i}{\sum_{\text{signal}} w_i + \sum_{\text{background}} w_i}$$

where w_i is the weight of the i th event.

Resulting nodes classified as either signal/background.

Iterate until stop criterion reached based on e.g. purity or minimum number of events in a node.

The set of cuts defines the decision boundary.



Example by MiniBooNE experiment, B. Roe et al., NIM 543 (2005) 577

Finding the best single cut

The level of separation within a node can, e.g., be quantified by the *Gini coefficient*, calculated from the (s or b) purity as:

$$G = p(1 - p)$$

For a cut that splits a set of events a into subsets b and c , one can quantify the improvement in separation by the change in weighted Gini coefficients:

$$\Delta = W_a G_a - W_b G_b - W_c G_c \quad \text{where, e.g.,} \quad W_a = \sum_{i \in a} w_i$$

Choose e.g. the cut to the maximize Δ ; a variant of this scheme can use instead of Gini e.g. the misclassification rate:

$$\varepsilon = 1 - \max(p, 1 - p)$$

Decision tree classifier

The terminal nodes (**leaves**) are classified a signal or background depending on majority vote (or e.g. signal fraction greater than a specified threshold).

This classifies every point in input-variable space as either signal or background, a **decision tree classifier**, with discriminant function

$$f(\mathbf{x}) = 1 \text{ if } \mathbf{x} \text{ in signal region, } -1 \text{ otherwise}$$

Decision trees tend to be very sensitive to statistical fluctuations in the training sample.

Methods such as **boosting** can be used to stabilize the tree.

Boosting

Boosting is a general method of creating a set of classifiers which can be combined to achieve a new classifier that is more stable and has a smaller error than any individual one.

Often applied to decision trees but, can be applied to any classifier.

Suppose we have a training sample T consisting of N events with

$\mathbf{x}_1, \dots, \mathbf{x}_N$	event data vectors (each \mathbf{x} multivariate)
y_1, \dots, y_N	true class labels, +1 for signal, -1 for background
w_1, \dots, w_N	event weights

Now define a rule to create from this an ensemble of training samples T_1, T_2, \dots , derive a classifier from each and average them.

Trick is to create modifications in the training sample that give classifiers with smaller error rates than those of the preceding ones.

A successful example is **AdaBoost** (Freund and Schapire, 1997).

AdaBoost

First initialize the training sample T_1 using the original

$\mathbf{x}_1, \dots, \mathbf{x}_N$	event data vectors
$\mathbf{y}_1, \dots, \mathbf{y}_N$	true class labels (+1 or -1)
$\mathbf{w}_1^{(1)}, \dots, \mathbf{w}_N^{(1)}$	event weights

with the weights equal and normalized such that

$$\sum_{i=1}^N w_i^{(1)} = 1$$

Then train the classifier $f_1(\mathbf{x})$ (e.g., a decision tree) with a method that uses the event weights. Recall for an event at point \mathbf{x} ,

$$f_1(\mathbf{x}) = +1 \text{ for } \mathbf{x} \text{ in signal region, } -1 \text{ in background region}$$

We will define an iterative procedure that gives a series of classifiers $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots$

Error rate of the k th classifier

At the k th iteration the classifier $f_k(\mathbf{x})$ has an error rate

$$\varepsilon_k = \sum_{i=1}^N w_i^{(k)} I(y_i f_k(\mathbf{x}_i) \leq 0)$$

where $I(X) = 1$ if X is true and is zero otherwise.

Next assign a score to the k th classifier based on its error rate,

$$\alpha_k = \frac{1}{2} \ln \frac{1 - \varepsilon_k}{\varepsilon_k}$$

Updating the event weights

The classifier at each iterative step is found from an updated training sample, in which the weight of event i is modified from step k to step $k+1$ according to

$$w_i^{(k+1)} = w_i^{(k)} \frac{e^{-\alpha_k f_k(\mathbf{x}_i) y_i}}{Z_k}$$

Here Z_k is a normalization factor defined such that the sum of the weights over all events is equal to one.

That is, the weight for event i is increased in the $k+1$ training sample if it was classified incorrectly in step k .

Idea is that next time around the classifier should pay more attention to this event and try to get it right.

Defining the classifier

After K boosting iterations, the final classifier is defined as a weighted linear combination of the $f_k(\mathbf{x})$,

$$t(\mathbf{x}) = \sum_{k=1}^K \alpha_k f_k(\mathbf{x})$$

One can show that the error rate on the training data of the final classifier satisfies the bound

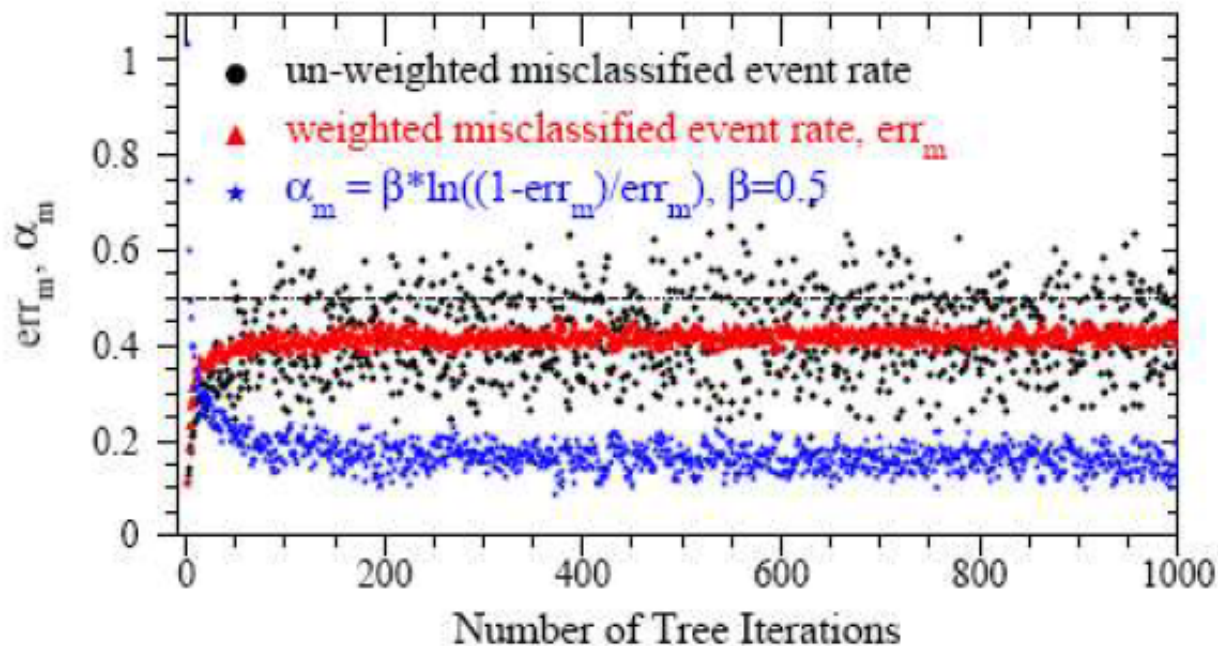
$$\varepsilon \leq \prod_{k=1}^K 2\sqrt{\varepsilon_k(1 - \varepsilon_k)}$$

i.e. as long as the $\varepsilon_k < \frac{1}{2}$ (better than random guessing), with enough boosting iterations every event in the training sample will be classified correctly.

BDT example from MiniBooNE

~200 input variables for each event (ν interaction producing e , μ or π).

Each individual tree is relatively weak, with a misclassification error rate $\sim 0.4 - 0.45$



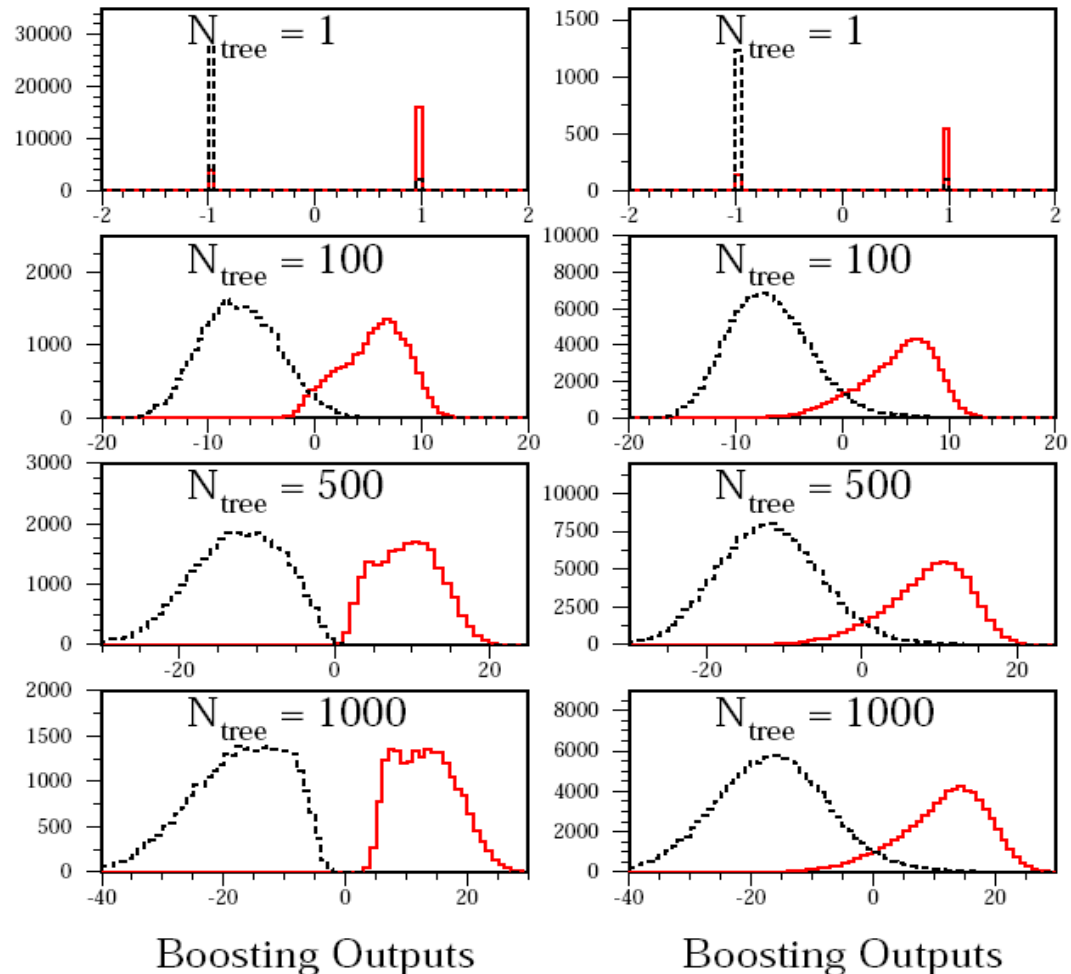
B. Roe et al., NIM 543 (2005) 577

Monitoring overtraining

From MiniBooNE
example:

Performance stable
after a few hundred
trees.

Training MC Samples .VS. Testing MC Samples



A simple example (2D)

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).

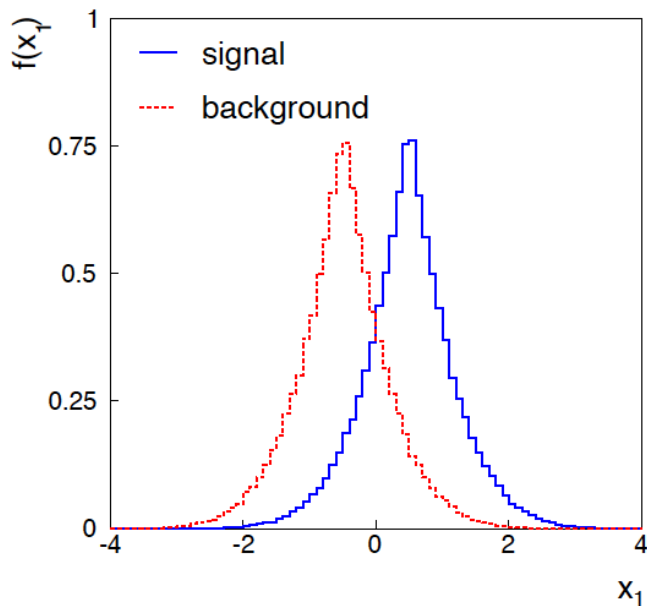
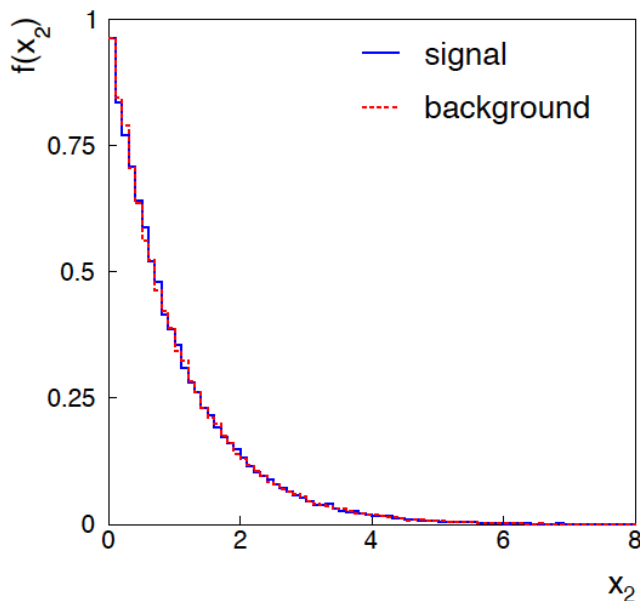
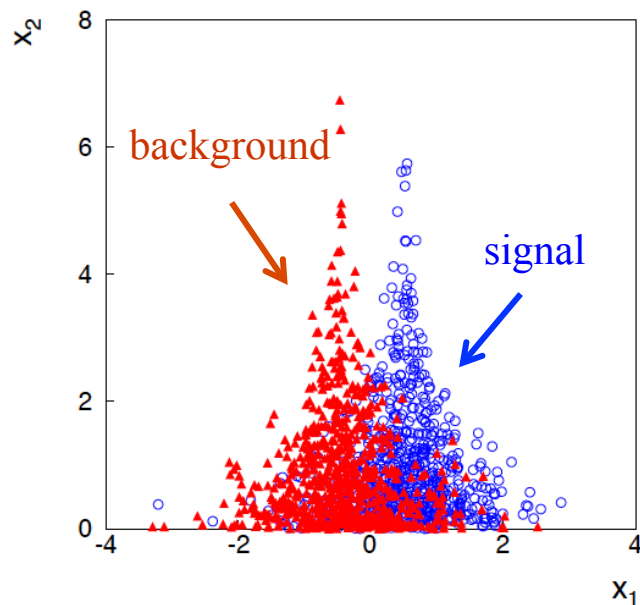
$f(x_1|x_2) \sim$ Gaussian, different means for s/b,
Gaussians have same σ , which depends on x_2 ,
 $f(x_2) \sim$ exponential, same for both s and b,
 $f(x_1, x_2) = f(x_1|x_2)f(x_2)$:

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

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

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

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?

Likelihood ratio for 2D example

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

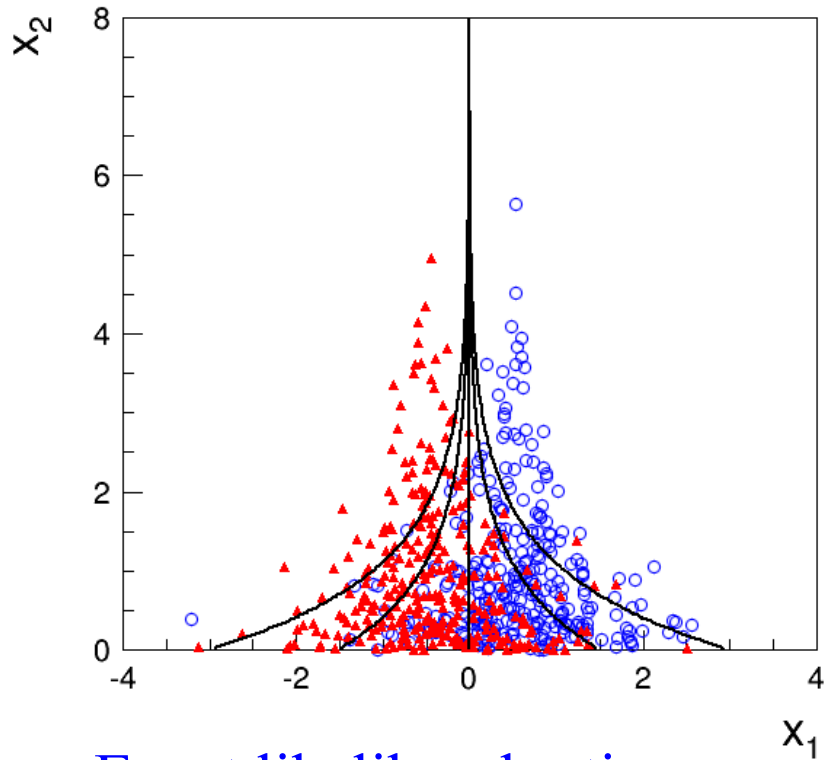
$$t(x_1, x_2) = \frac{f(x_1, x_2 | \mathbf{s})}{f(x_1, x_2 | \mathbf{b})}$$

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

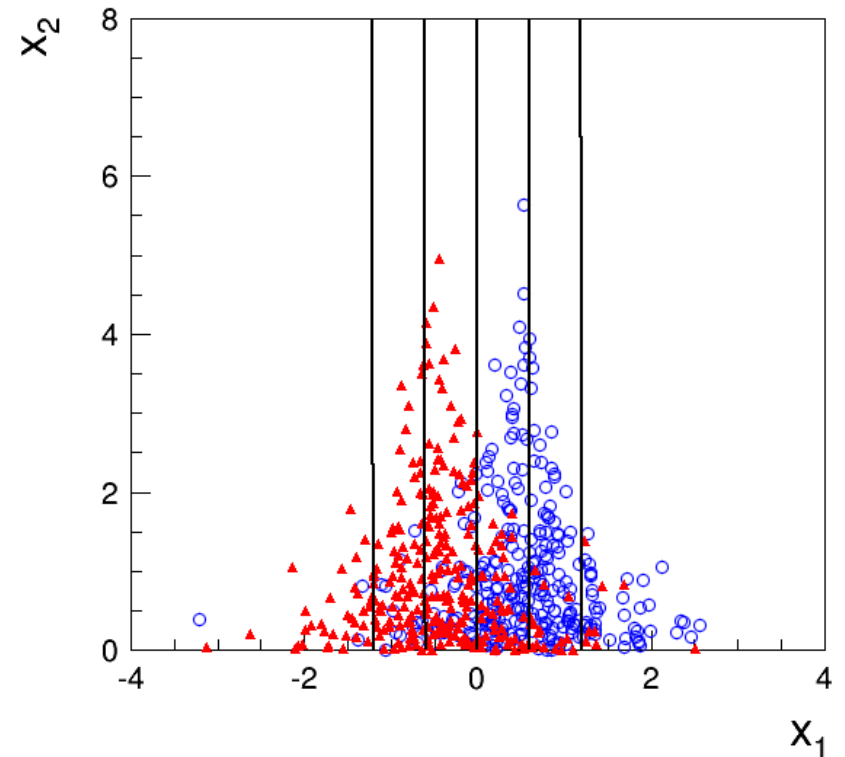
$$\ln t = \frac{\frac{1}{2}(\mu_{\mathbf{b}}^2 - \mu_{\mathbf{s}}^2) + (\mu_{\mathbf{s}} - \mu_{\mathbf{b}})x_1}{\sigma_0^2 e^{-2x_2/\xi}}$$

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

Contours of constant MVA output

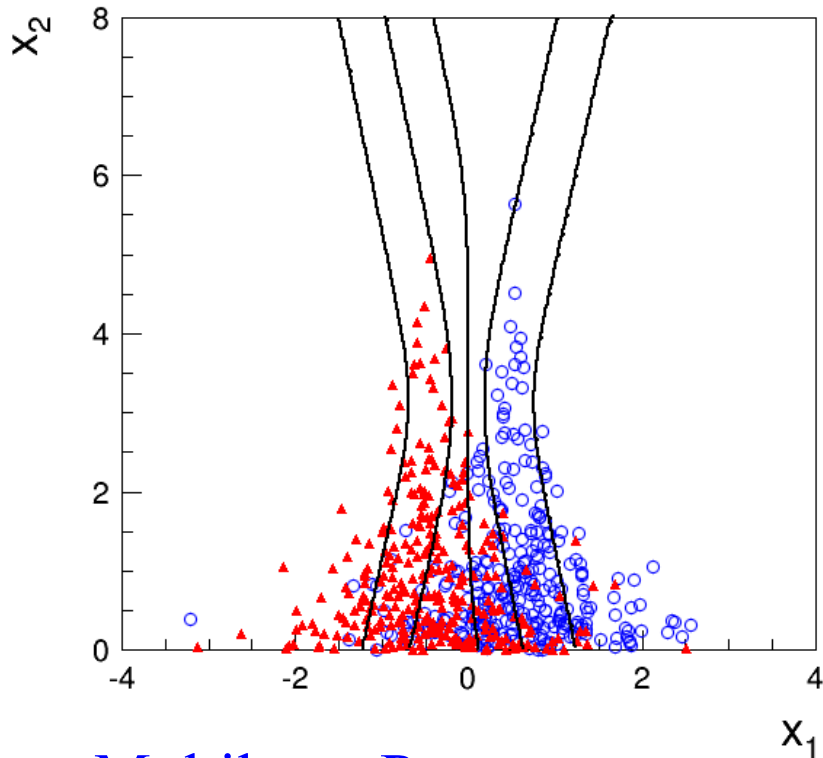


Exact likelihood ratio

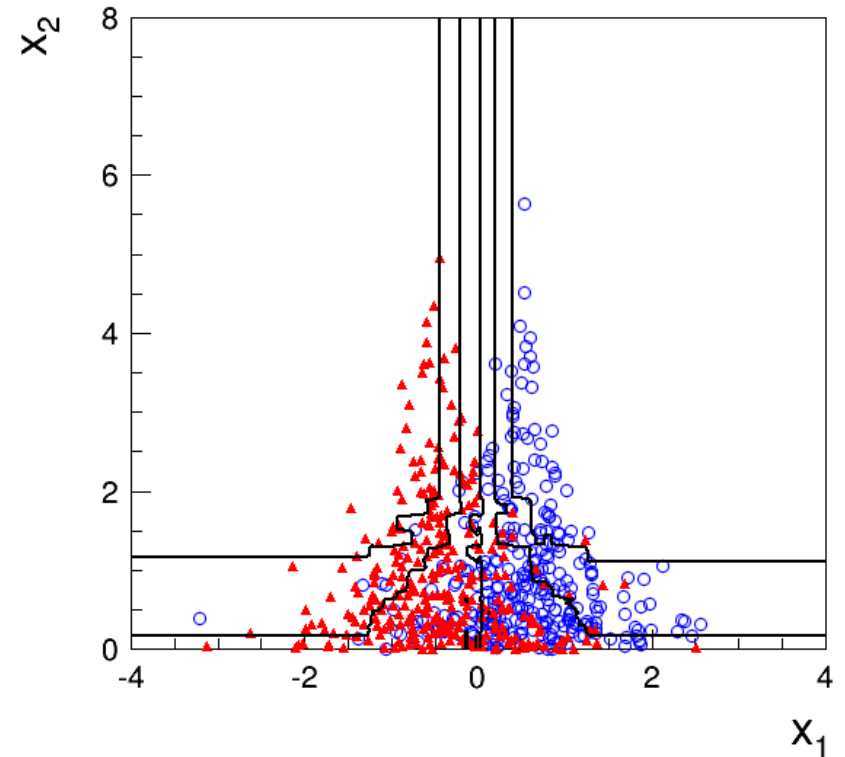


Fisher discriminant

Contours of constant MVA output



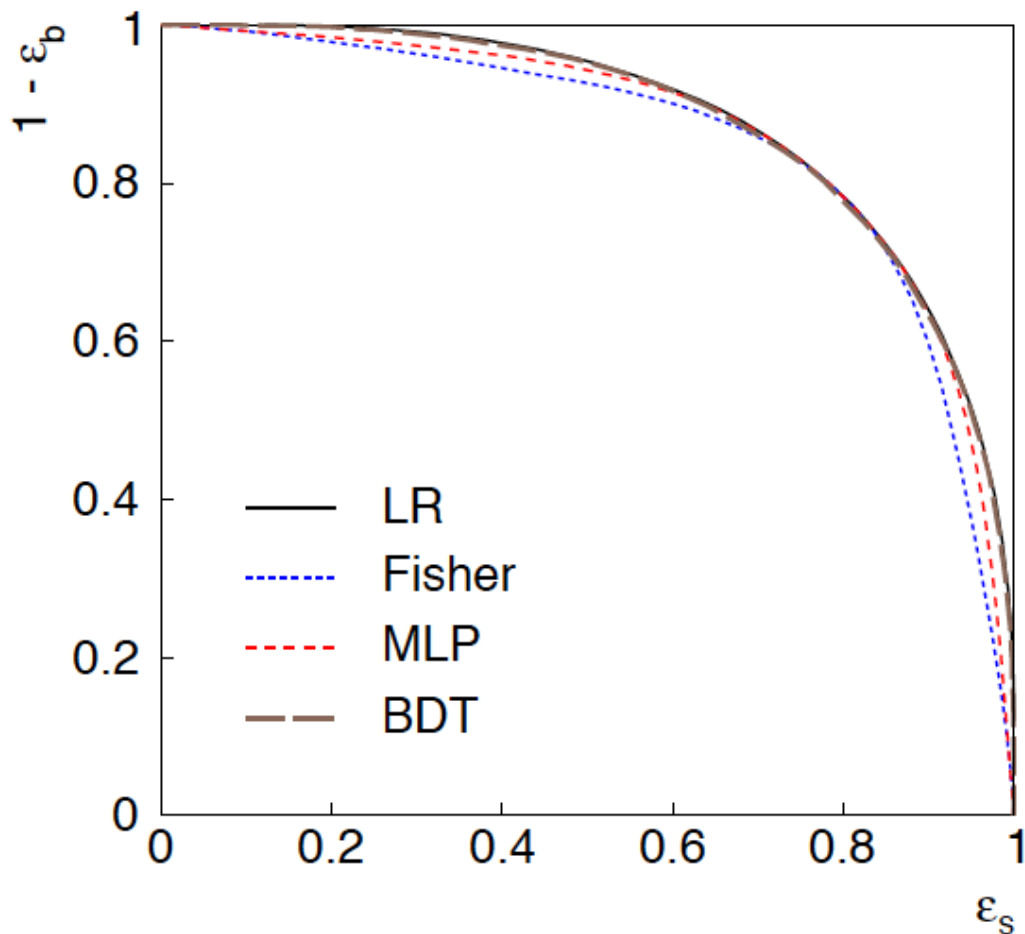
Multilayer Perceptron
1 hidden layer with 2 nodes



Boosted Decision Tree
200 iterations (AdaBoost)

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

ROC curve



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

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

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

2D Example: discussion

Even though the distribution of x_2 is same for signal and background, x_1 and x_2 are not independent, so using x_2 as an input variable helps.

Here we can understand why: high values of x_2 correspond to a smaller σ for the Gaussian of x_1 . So high x_2 means that the value of x_1 was well measured.

If we don't consider x_2 , then all of the x_1 measurements are lumped together. Those with large σ (low x_2) “pollute” the well measured events with low σ (high x_2).

Often in HEP there may be variables that are characteristic of how well measured an event is (region of detector, number of pile-up vertices,...). Including these variables in a multivariate analysis preserves the information carried by the well-measured events, leading to improved performance.