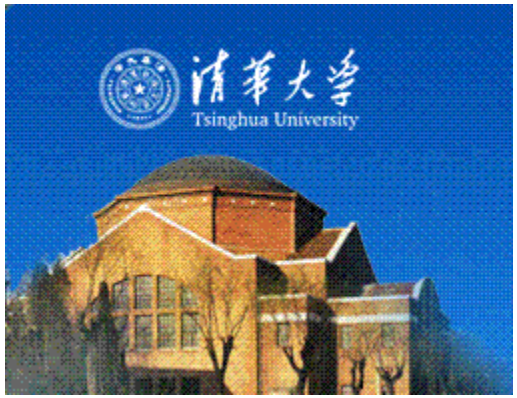
Statistical Methods in Particle Physics
Day 2: Multivariate Methods (I)



清华大学高能物理研究中心

2010年4月12—16日

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

Outline of lectures

Day #1: Introduction
Review of probability and Monte Carlo
Review of statistics: parameter estimation

→ Day #2: Multivariate methods (I)
Event selection as a statistical test
Cut-based, linear discriminant, neural networks

Day #3: Multivariate methods (II)
More multivariate classifiers: BDT, SVM ,...

Day #4: Significance tests for discovery and limits
Including systematics using profile likelihood

Day #5: Bayesian methods
Bayesian parameter estimation and model selection

Day #2: outline

Multivariate methods for HEP

- Event selection as a statistical test
- Neyman-Pearson lemma and likelihood ratio test

Some multivariate classifiers

- Cut-based event selection
- Linear classifiers
- Neural networks
- Probability density estimation methods

Resources on multivariate methods

Books:

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

T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer, 2001

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

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

Materials from some recent meetings:

PHYSTAT conference series (2002, 2003, 2005, 2007,...) see
www.phystat.org

Caltech workshop on multivariate analysis, 11 February, 2008
indico.cern.ch/conferenceDisplay.py?confId=27385

SLAC Lectures on Machine Learning by Ilya Narsky (2006)
www-group.slac.stanford.edu/sluo/Lectures/Stat2006_Lectures.html

Software for multivariate analysis

TMVA, Höcker, Stelzer, Tegenfeldt, Voss, Voss, physics/0703039

From **tmva.sourceforge.net**, also distributed with ROOT
Variety of classifiers
Good manual

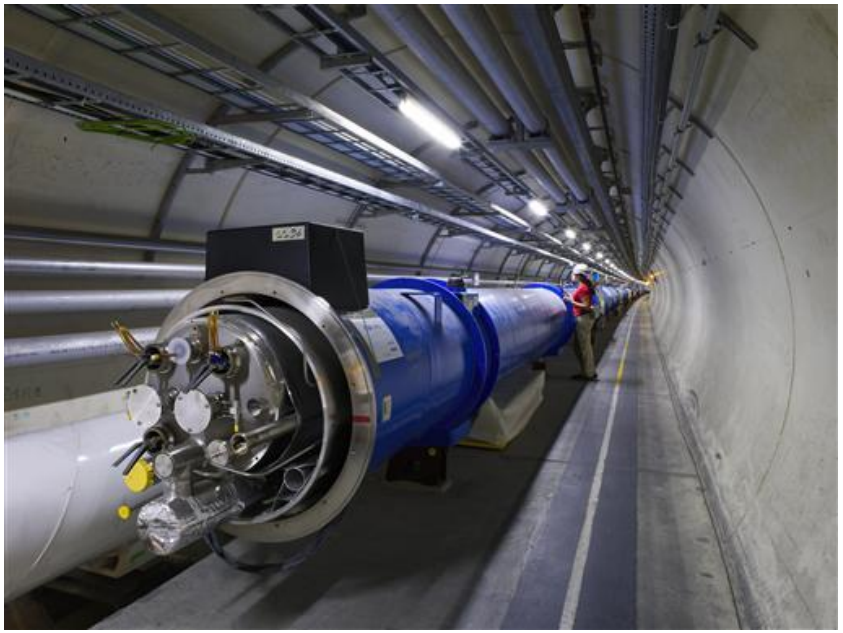
StatPatternRecognition, I. Narsky, physics/0507143

Further info from **www.hep.caltech.edu/~narsky/spr.html**
Also wide variety of methods, many complementary to **TMVA**
Currently appears project no longer to be supported

The Large Hadron Collider

Counter-rotating proton beams
in 27 km circumference ring

pp centre-of-mass energy 14 TeV

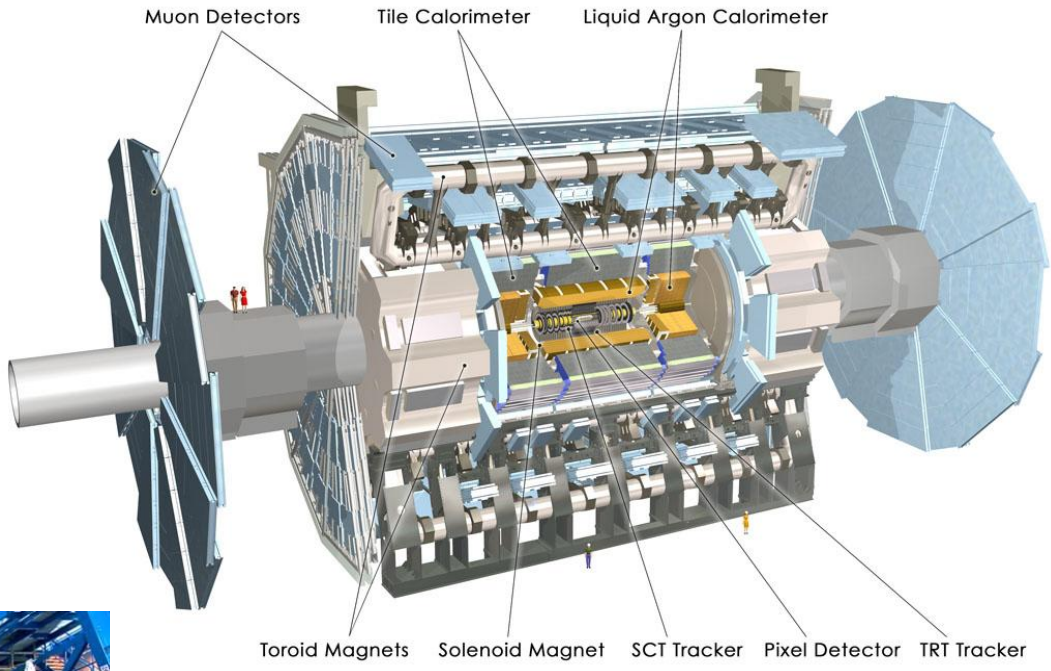


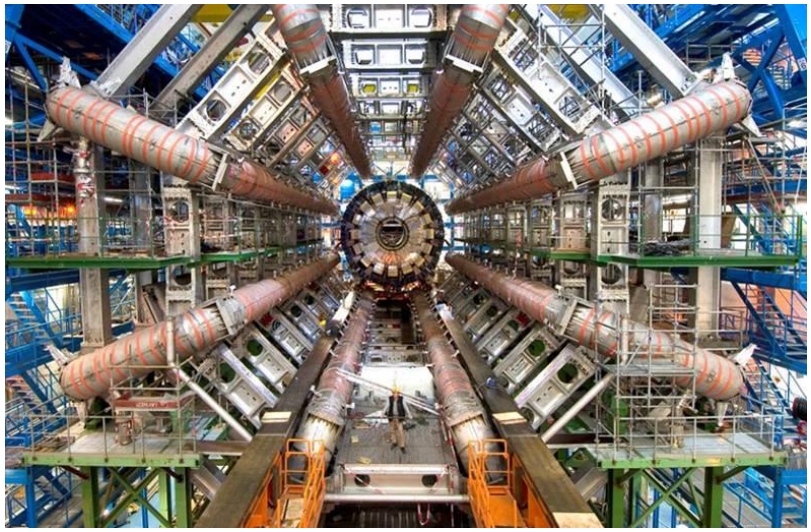
Detectors at 4 pp collision points:

ATLAS
CMS general purpose
LHCb (b physics)
ALICE (heavy ion physics)

The ATLAS detector

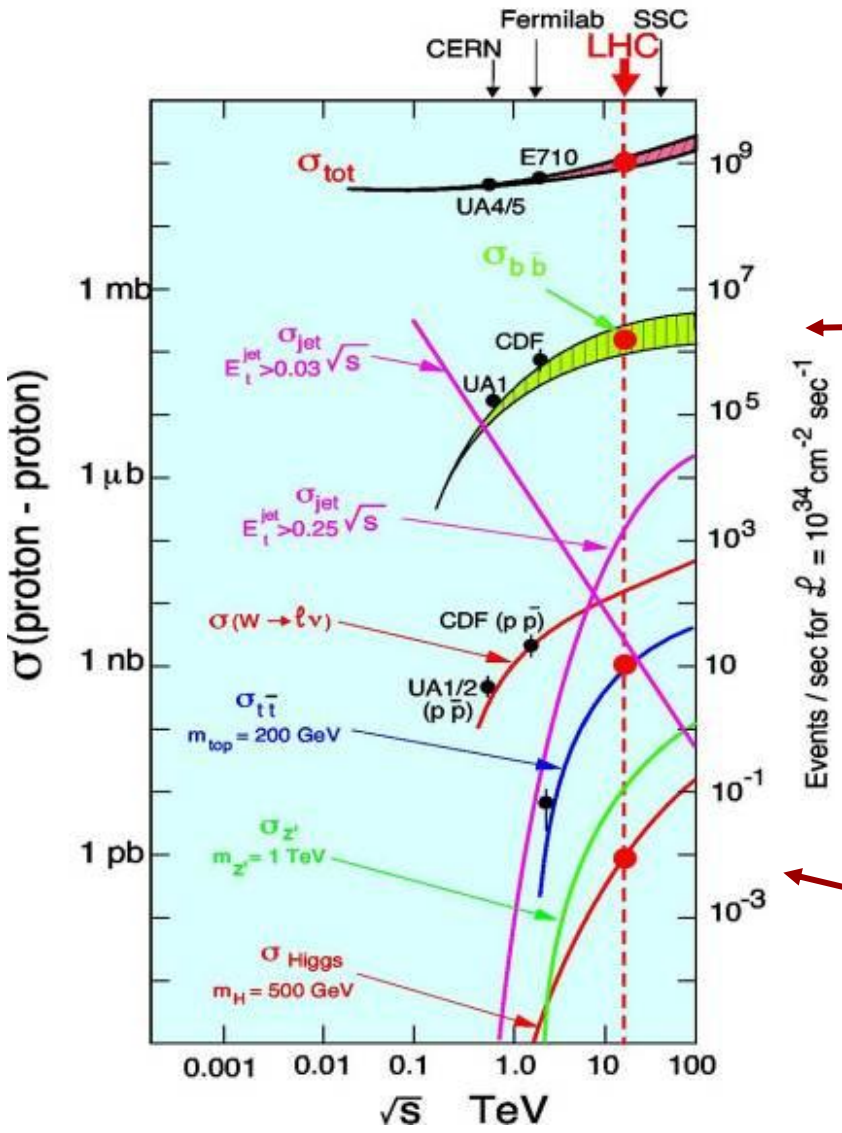
2100 physicists
37 countries
167 universities/labs





25 m diameter
46 m length
7000 tonnes
~10^8 electronic channels

LHC event production rates



most events (boring)

mildly interesting

interesting

very interesting
(~1 out of every 10^{11})

LHC data

At LHC, ~10^9 pp collision events per second, mostly uninteresting

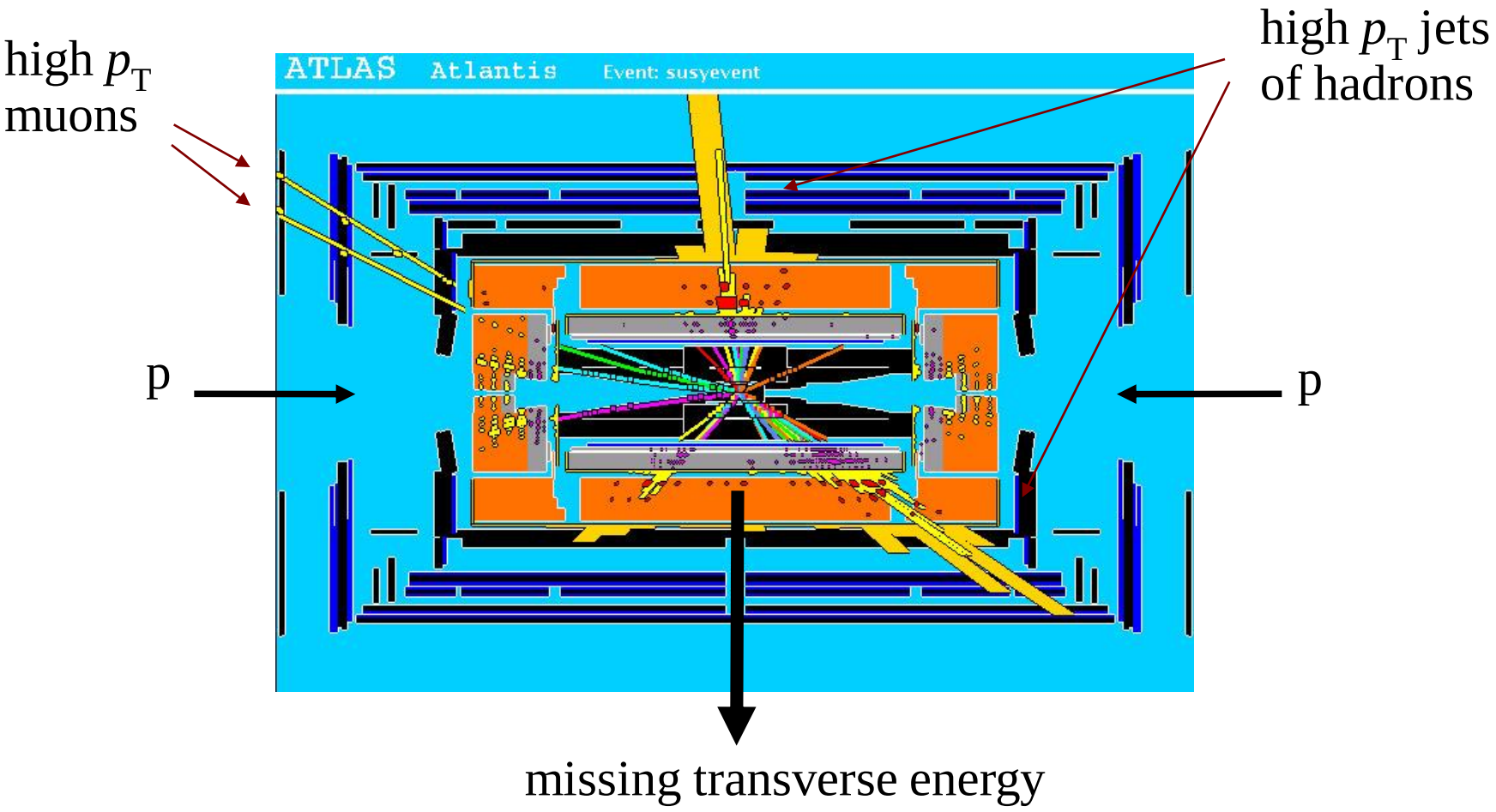
do quick sifting, record ~200 events/sec
single event ~ 1 Mbyte
1 "year" ≈ 10^7 s, 10^{16} pp collisions / year
2×10^9 events recorded / year (~2 Pbyte / year)

For new/rare processes, rates at LHC can be vanishingly small

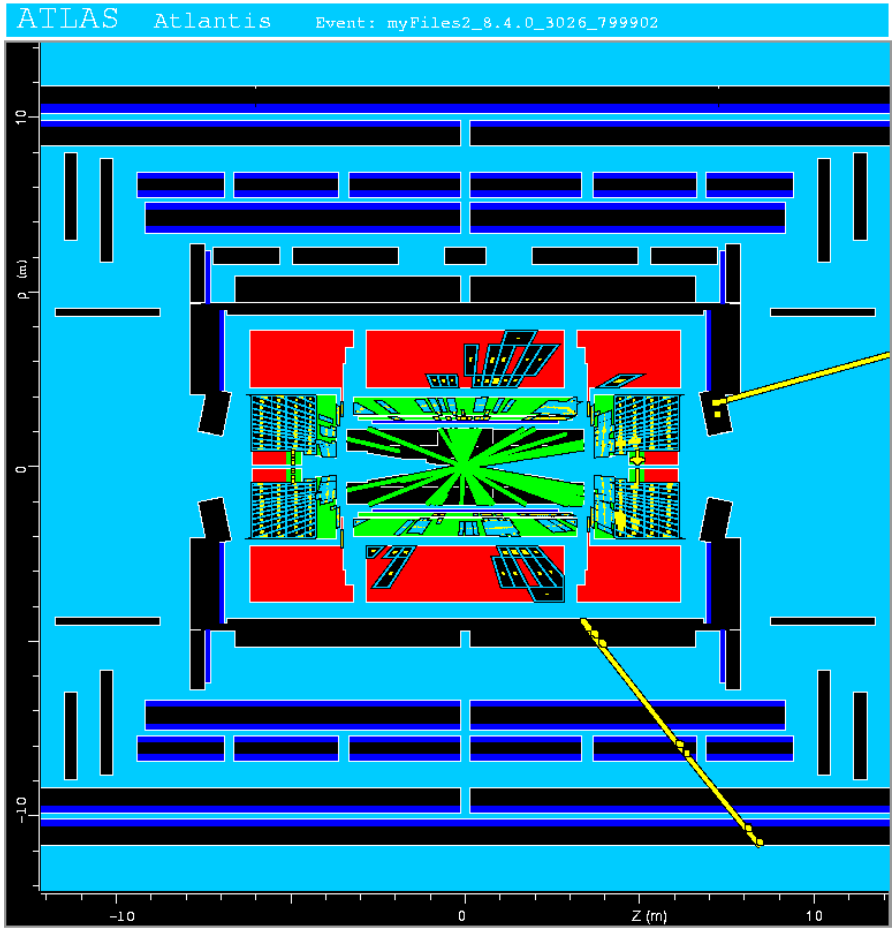
e.g. Higgs bosons detectable per year could be ~10^3
→ 'needle in a haystack'

For Standard Model and (many) non-SM processes we can generate simulated data with Monte Carlo programs (including simulation of the detector).

A simulated SUSY event in ATLAS



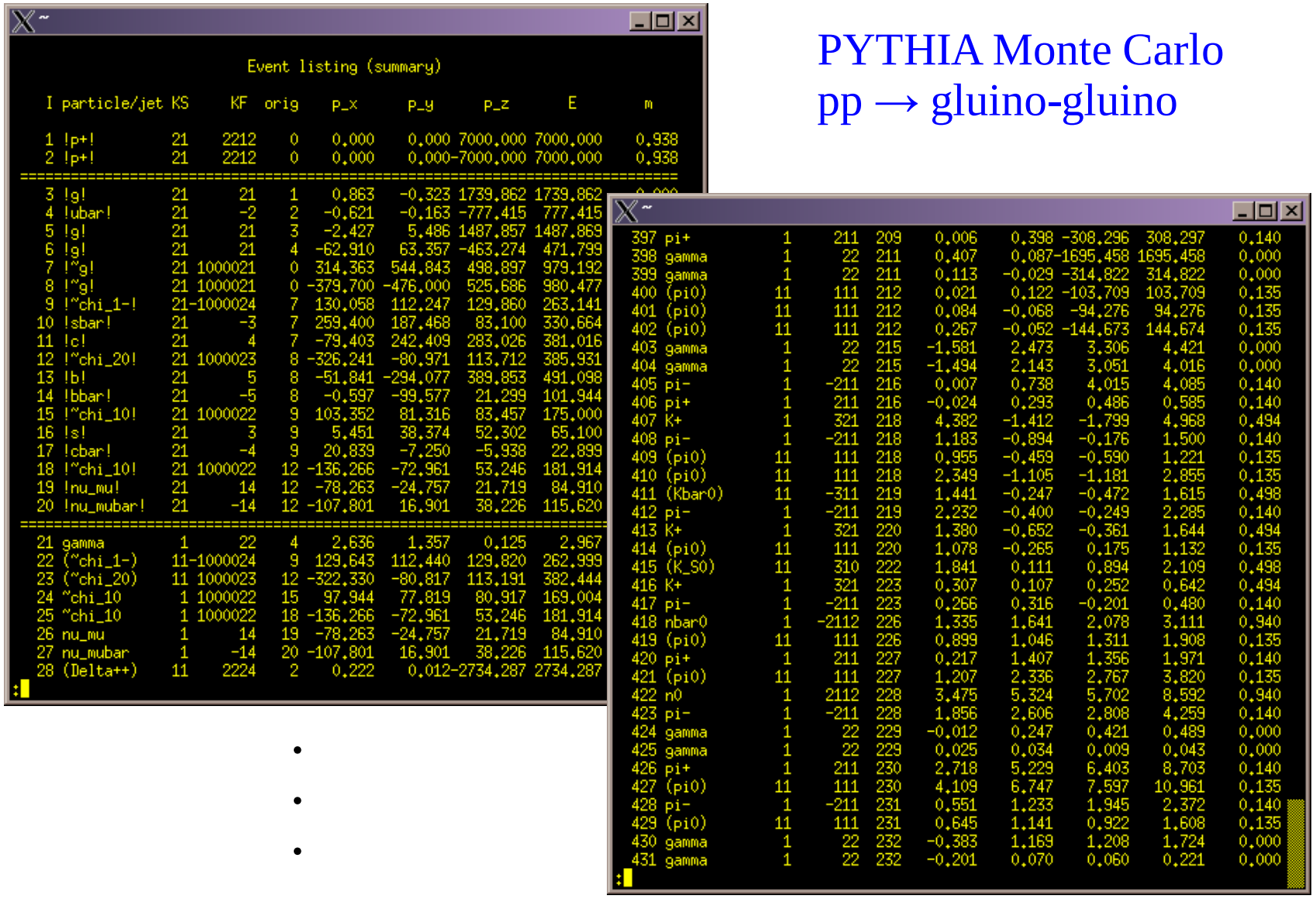
Background events



This event from Standard Model ttbar production also has high p_T jets and muons, and some missing transverse energy.

→ can easily mimic a SUSY event.

A simulated event



Event listing (summary)

I	particle/jet	KS	KF	orig	p_x	p_y	p_z	E	m
1	!p+!	21	2212	0	0.000	0.000	7000.000	7000.000	0.938
2	!p+!	21	2212	0	0.000	0.000	-7000.000	7000.000	0.938
3	!g!	21	21	1	0.863	-0.323	1739.862	1739.862	0.000
4	!ubar!	21	-2	2	-0.621	-0.163	-777.415	777.415	0.000
5	!g!	21	21	3	-2.427	5.486	1487.857	1487.869	0.000
6	!g!	21	21	4	-62.910	63.357	-463.274	471.799	0.000
7	!~g!	21	1000021	0	314.363	544.843	498.897	979.192	0.000
8	!~g!	21	1000021	0	-379.700	-476.000	525.686	980.477	0.000
9	!~chi_1-!	21	-1000024	7	130.058	112.247	129.860	263.141	0.000
10	!sbar!	21	-3	7	259.400	187.468	83.100	330.664	0.000
11	!c!	21	4	7	-79.403	242.409	283.026	381.016	0.000
12	!~chi_20!	21	1000023	8	-326.241	-80.971	113.712	385.931	0.000
13	!b!	21	5	8	-51.841	-294.077	389.853	491.098	0.000
14	!bbar!	21	-5	8	-0.597	-99.577	21.299	101.944	0.000
15	!~chi_10!	21	1000022	9	103.352	81.316	83.457	175.000	0.000
16	!s!	21	3	9	5.451	38.374	52.302	65.100	0.000
17	!cbar!	21	-4	9	20.839	-7.250	-5.938	22.899	0.000
18	!~chi_10!	21	1000022	12	-136.266	-72.961	53.246	181.914	0.000
19	!nu_mu!	21	14	12	-78.263	-24.757	21.719	84.910	0.000
20	!nu_mubar!	21	-14	12	-107.801	16.901	38.226	115.620	0.000
21	gamma	1	22	4	2.636	1.357	0.125	2.967	0.000
22	(~chi_1-)	11	-1000024	9	129.643	112.440	129.820	262.999	0.000
23	(~chi_20)	11	1000023	12	-322.330	-80.817	113.191	382.444	0.000
24	~chi_10	1	1000022	15	97.944	77.819	80.917	169.004	0.000
25	~chi_10	1	1000022	18	-136.266	-72.961	53.246	181.914	0.000
26	nu_mu	1	14	19	-78.263	-24.757	21.719	84.910	0.000
27	nu_mubar	1	-14	20	-107.801	16.901	38.226	115.620	0.000
28	(Delta++)	11	2224	2	0.222	0.012	-2734.287	2734.287	0.000

397	pi+	1	211	209	0.006	0.398	-308.296	308.297	0.140
398	gamma	1	22	211	0.407	0.087	-1695.458	1695.458	0.000
399	gamma	1	22	211	0.113	-0.029	-314.822	314.822	0.000
400	(pi0)	11	111	212	0.021	0.122	-103.709	103.709	0.135
401	(pi0)	11	111	212	0.084	-0.068	-94.276	94.276	0.135
402	(pi0)	11	111	212	0.267	-0.052	-144.673	144.674	0.135
403	gamma	1	22	215	-1.581	2.473	3.306	4.421	0.000
404	gamma	1	22	215	-1.494	2.143	3.051	4.016	0.000
405	pi-	1	-211	216	0.007	0.738	4.015	4.085	0.140
406	pi+	1	211	216	-0.024	0.293	0.486	0.585	0.140
407	K+	1	321	218	4.382	-1.412	-1.799	4.968	0.494
408	pi-	1	-211	218	1.183	-0.894	-0.176	1.500	0.140
409	(pi0)	11	111	218	0.955	-0.459	-0.590	1.221	0.135
410	(pi0)	11	111	218	2.349	-1.105	-1.181	2.855	0.135
411	(Kbar0)	11	-311	219	1.441	-0.247	-0.472	1.615	0.498
412	pi-	1	-211	219	2.232	-0.400	-0.249	2.285	0.140
413	K+	1	321	220	1.380	-0.652	-0.361	1.644	0.494
414	(pi0)	11	111	220	1.078	-0.265	0.175	1.132	0.135
415	(K_S0)	11	310	222	1.841	0.111	0.894	2.109	0.498
416	K+	1	321	223	0.307	0.107	0.252	0.642	0.494
417	pi-	1	-211	223	0.266	0.316	-0.201	0.480	0.140
418	nbar0	1	-2112	226	1.335	1.641	2.078	3.111	0.940
419	(pi0)	11	111	226	0.899	1.046	1.311	1.908	0.135
420	pi+	1	211	227	0.217	1.407	1.356	1.971	0.140
421	(pi0)	11	111	227	1.207	2.336	2.767	3.820	0.135
422	n0	1	2112	228	3.475	5.324	5.702	8.592	0.940
423	pi-	1	-211	228	1.856	2.606	2.808	4.259	0.140
424	gamma	1	22	229	-0.012	0.247	0.421	0.489	0.000
425	gamma	1	22	229	0.025	0.034	0.009	0.043	0.000
426	pi+	1	211	230	2.718	5.229	6.403	8.703	0.140
427	(pi0)	11	111	230	4.109	6.747	7.597	10.961	0.135
428	pi-	1	-211	231	0.551	1.233	1.945	2.372	0.140
429	(pi0)	11	111	231	0.645	1.141	0.922	1.608	0.135
430	gamma	1	22	232	-0.383	1.169	1.208	1.724	0.000
431	gamma	1	22	232	-0.201	0.070	0.060	0.221	0.000

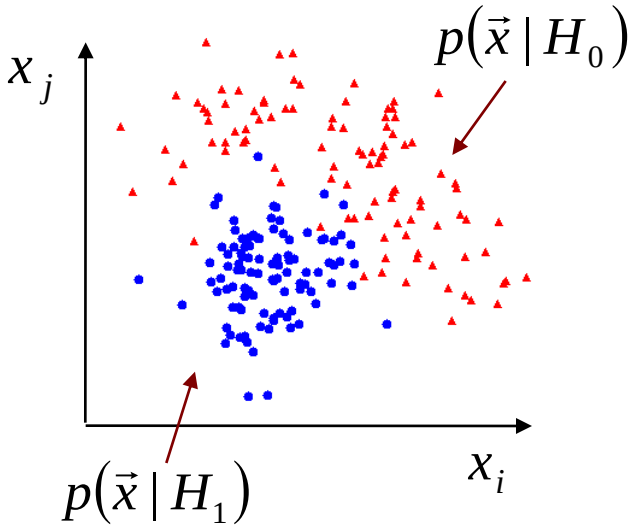
PYTHIA Monte Carlo
pp → gluino-gluino

Event selection as a statistical test

For each event we measure a set of numbers: $\vec{x} = (x_1, \ldots, x_n)$

x_1 = jet p_T
x_2 = missing energy
x_3 = particle i.d. measure, ...

$\vec{x}$ follows some n-dimensional joint probability density, which depends on the type of event produced, i.e., was it $pp \to t\bar{t}$, $pp \to \tilde{g}\tilde{g}, \ldots$

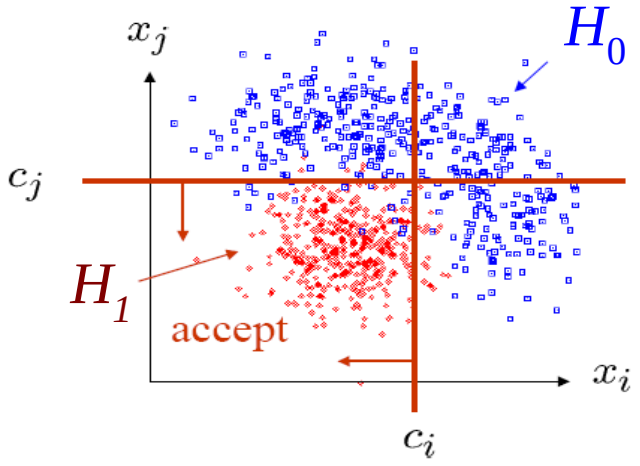


E.g. hypotheses H_0, H_1, ...
Often simply "signal", "background"

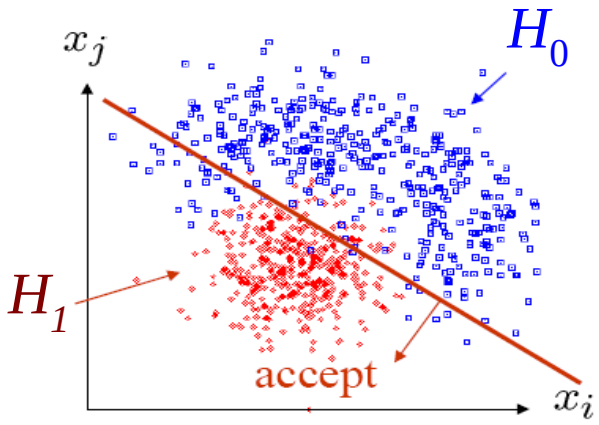
Finding an optimal decision boundary

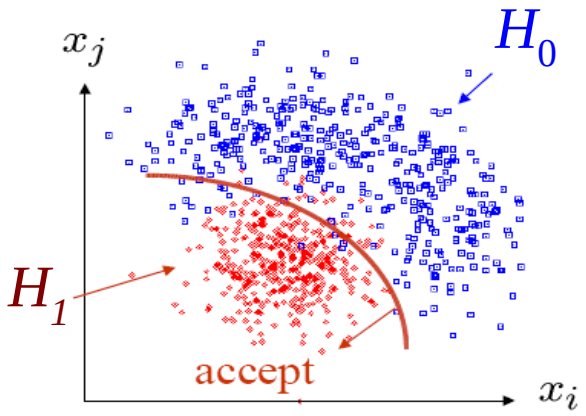
In particle physics usually start by making simple "cuts":

$$x_i < c_i$$
$$x_j < c_j$$



Maybe later try some other type of decision boundary:





Test statistics

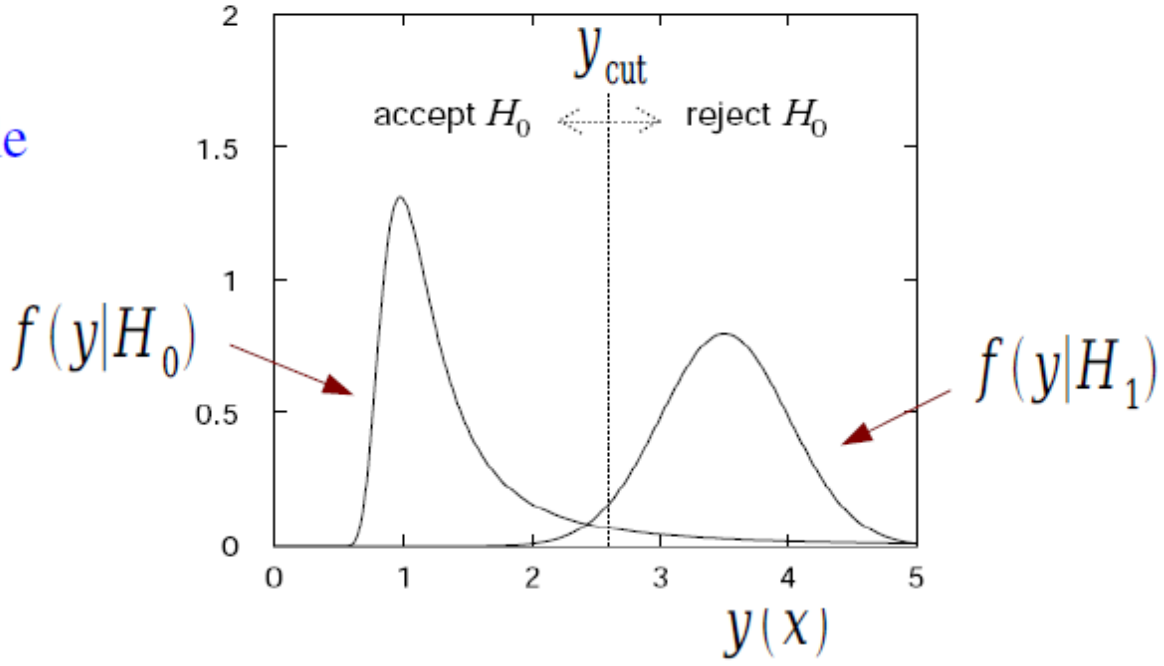
The decision boundary is a surface in the n-dimensional space of input variables, e.g., $y(\vec{x}) = \text{const}$.

We can treat the $y(x)$ as a scalar test statistic or discriminating function, and try to define this function so that its distribution has the maximum possible separation between the event types:

The decision boundary is now effectively a single cut on $y(\boldsymbol{x})$, dividing $\boldsymbol{x}$-space into two regions:

R_0 (accept H_0)

R_1 (reject H_0)



Classification viewed as a statistical test

Probability to reject H_0 if it is true (type I error): $\alpha = \int_{R_1} f(y|H_0)\,dy$

α = significance level, size of test, false discovery rate

Probability to accept H_0 if H_1 is true (type II error): $\beta = \int_{R_0} f(y|H_1)\,dy$

$1 - \beta$ = power of test with respect to H_1

Equivalently if e.g. H_0 = background, H_1 = signal, use efficiencies:

$$\varepsilon_s = \int_{R_1} f(y|H_1)\,dy = 1 - \beta = \text{power} \qquad \varepsilon_b = \int_{R_0} f(y|H_0)\,dy = 1 - \alpha$$

Purity / misclassification rate

Consider the probability that an event assigned to a certain category is classified correctly (i.e., the purity).

Use Bayes' theorem:

Here R_1 is signal region

prior probability

$$P(\mathrm{s}|\mathbf{x} \in R_1) = \frac{P(\mathbf{x} \in R_1|\mathrm{s})P(\mathrm{s})}{P(\mathbf{x} \in R_1|\mathrm{s})P(\mathrm{s}) + P(\mathbf{x} \in R_1|\mathrm{b})P(\mathrm{b})}$$

posterior probability

N.B. purity depends on the prior probabilities for an event to be signal or background (~s, b cross sections).

Constructing a test statistic

The Neyman-Pearson lemma states: to obtain the highest background rejection for a given signal efficiency (highest power for a given significance level), choose the acceptance region for signal such that

$$\frac{p(\vec{x}|\mathrm{s})}{p(\vec{x}|\mathrm{b})} > c$$

where c is a constant that determines the signal efficiency.

Equivalently, the optimal discriminating function is given by the likelihood ratio:

$$y(\vec{x}) = \frac{p(\vec{x}|\mathrm{s})}{p(\vec{x}|\mathrm{b})}$$

N.B. any monotonic function of this is just as good.

Neyman-Pearson doesn't always help

The problem is that we usually don't have explicit formulae for the pdfs $p(\boldsymbol{x}|\mathrm{s})$, $p(\boldsymbol{x}|\mathrm{b})$, so for a given $\boldsymbol{x}$ we can't evaluate the likelihood ratio.

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

"training data"

generate $\vec{x} \sim p(\vec{x}|\mathrm{s})$ $\longrightarrow$ $\vec{x}_1, \ldots, \vec{x}_{N_s}$ events of known type

generate $\vec{x} \sim p(\vec{x}|\mathrm{b})$ $\longrightarrow$ $\vec{x}_1, \ldots, \vec{x}_{N_b}$

Naive try: enter each (s,b) event into an n-dimensional histogram, use e.g. M bins for each of the n dimensions, total of M^n cells.

n is potentially large $\rightarrow$ prohibitively large number of cells to populate, can't generate enough training data.

Two distinct event selection problems

In some cases, the event types in question are both known to exist.

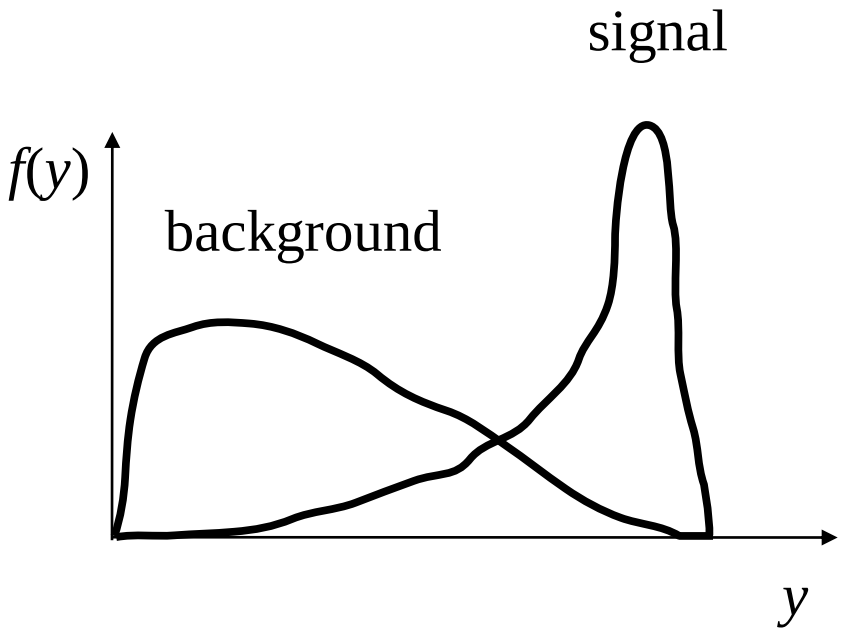
Example: separation of different particle types (electron vs muon)
Use the selected sample for further study.

In other cases, the null hypothesis H_0 means "Standard Model" events, and the alternative H_1 means "events of a type whose existence is not yet established" (to do so is the goal of the analysis).

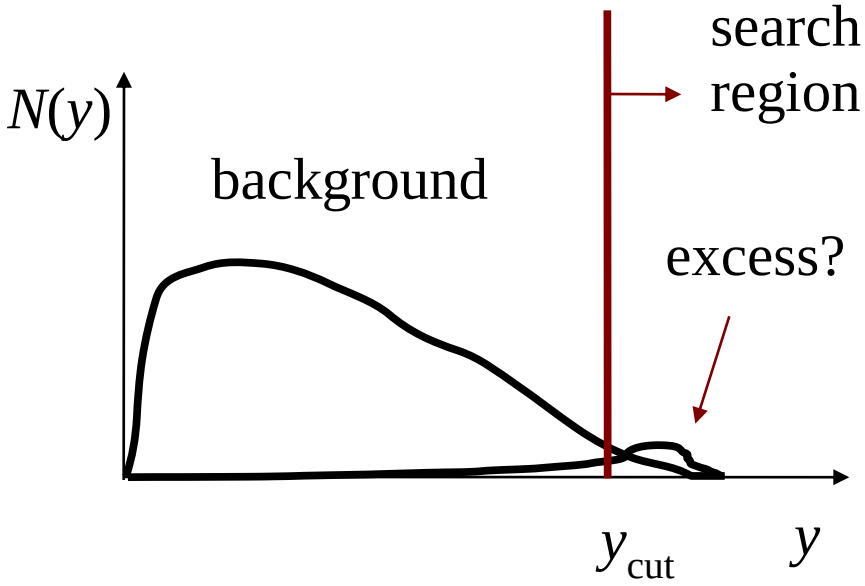
Many subtle issues here, mainly related to the heavy burden of proof required to establish presence of a new phenomenon.

Typically require p-value of background-only hypothesis below ~ 10^{-7} (a 5 sigma effect) to claim discovery of "New Physics".

Using classifier output for discovery



Normalized to unity



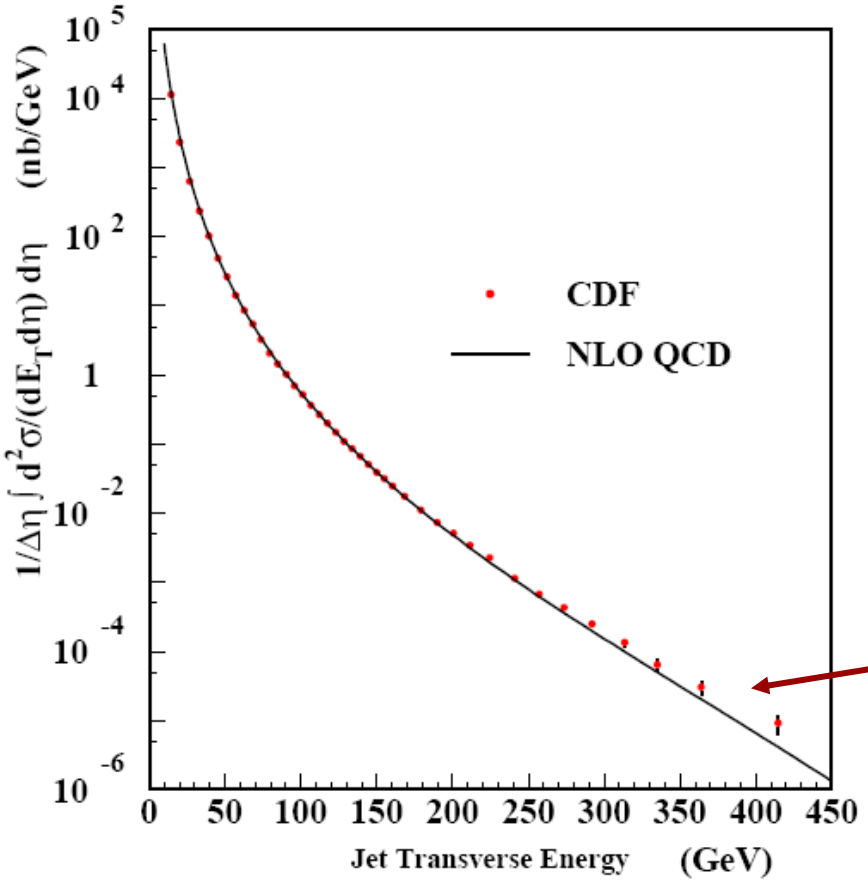
Normalized to expected
number of events

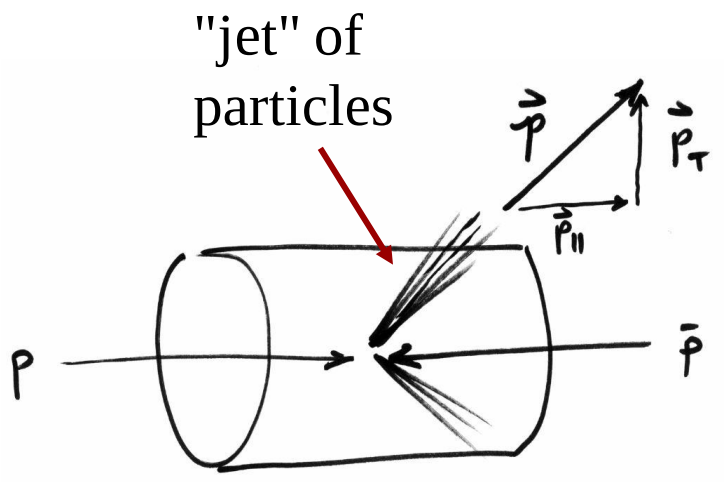
Discovery = number of events found in search region incompatible with background-only hypothesis.

p-value of background-only hypothesis can depend crucially distribution $f(y|\mathrm{b})$ in the "search region".

Example of a "cut-based" study

In the 1990s, the CDF experiment at Fermilab (Chicago) measured the number of hadron jets produced in proton-antiproton collisions as a function of their momentum perpendicular to the beam direction:

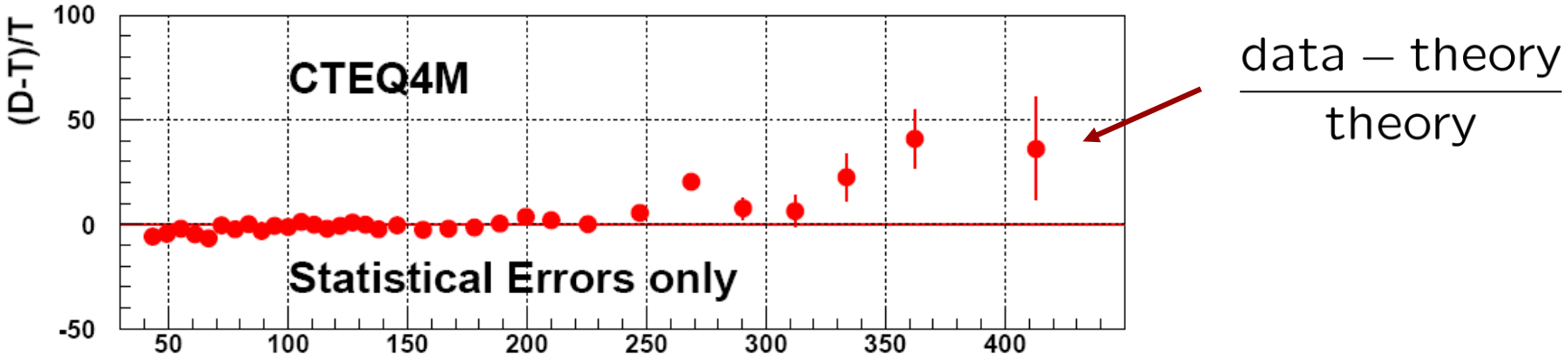




Prediction low relative to data for very high transverse momentum.

High p_T jets = quark substructure?

Although the data agree remarkably well with the Standard Model (QCD) prediction overall, the excess at high p_T appears significant:



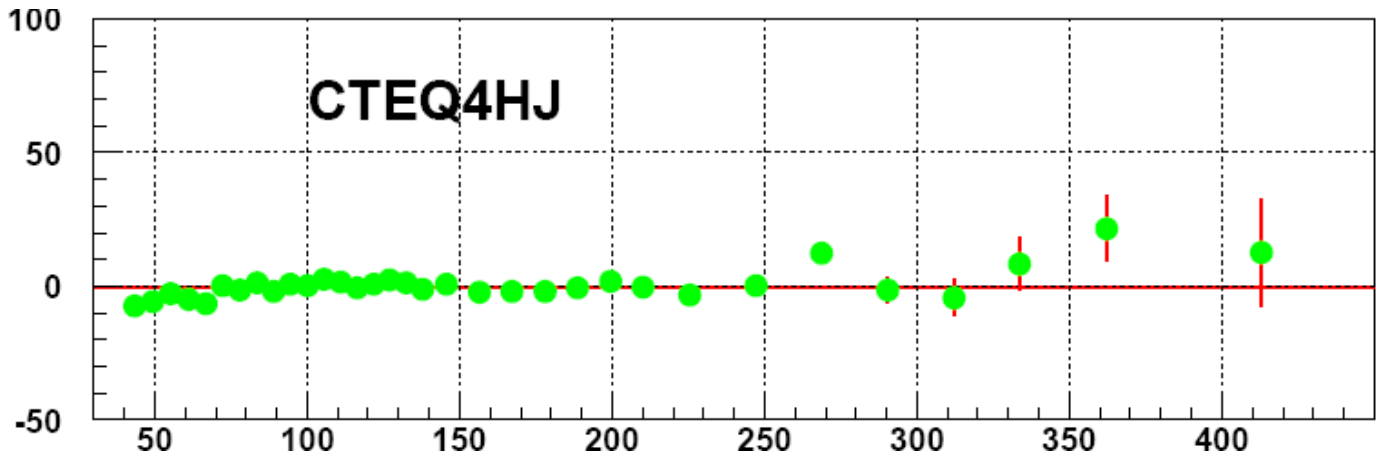
The fact that the variable is "understandable" leads directly to a plausible explanation for the discrepancy, namely, that quarks could possess an internal substructure.

Would not have been the case if the variable plotted was a complicated combination of many inputs.

High p_T jets from parton model uncertainty

Furthermore the physical understanding of the variable led one to a more plausible explanation, namely, an uncertain modeling of the quark (and gluon) momentum distributions inside the proton.

When model adjusted, discrepancy largely disappears:



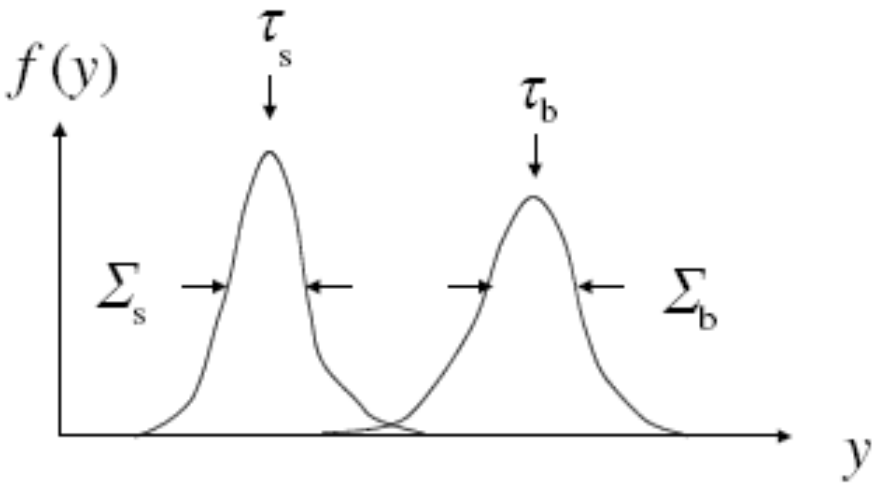
Can be regarded as a "success" of the cut-based approach. Physical understanding of output variable led to solution of apparent discrepancy.

Linear test statistic

Ansatz: $y(\vec{x}) = \sum_{i=1}^{n} w_i x_i = \vec{w}^T \vec{x}$

Choose the parameters w_1, ..., w_n so that the pdfs $f(y|\mathrm{s})$, $f(y|\mathrm{b})$ have maximum 'separation'. We want:

large distance between mean values, small widths



→ Fisher: maximize $J(\vec{w}) = \dfrac{(\tau_s - \tau_b)^2}{\Sigma_s^2 + \Sigma_b^2}$

Coefficients for maximum separation

We have $(\mu_k)_i = \int x_i \, p(\vec{x}|H_k) \, d\vec{x}$ ← mean, covariance of $\boldsymbol{x}$

$$(V_k)_{ij} = \int (x-\mu_k)_i (x-\mu_k)_j \, p(\vec{x}|H_k) \, d\vec{x}$$

where $k = 0, 1$ (hypothesis)

and $i, j = 1, \ldots, n$ (component of $\boldsymbol{x}$)

For the mean and variance of $y(\vec{x})$ we find

$$\tau_k = \int y(\vec{x}) \, p(\vec{x}|H_k) \, d\vec{x} = \vec{w}^T \vec{\mu}_k$$

$$\Sigma_k^2 = \int (y(\vec{x}) - \tau_k)^s \, p(\vec{x}|H_k) \, d\vec{x} = \vec{w}^T V_k \vec{w}$$

Determining the coefficients $\boldsymbol{w}$

The numerator of $J(\boldsymbol{w})$ is

$$(\tau_0-\tau_1)^2=\sum_{i,j=1}^{n} w_i w_j (\mu_0-\mu_1)_i (\mu_0-\mu_1)_j$$

'between' classes

$$=\sum_{i,j=1}^{n} w_i w_j B_{ij}=\vec{w}^T B\,\vec{w}$$

and the denominator is

'within' classes

$$\Sigma_0^2+\Sigma_1^2=\sum_{i,j=1}^{n} w_i w_j (V_0+V_1)_{ij}=\vec{w}^T W\,\vec{w}$$

→ maximize

$$J(\vec{w})=\frac{\vec{w}^T B\,\vec{w}}{\vec{w}^T W\,\vec{w}}=\frac{\text{separation between classes}}{\text{separation within classes}}$$

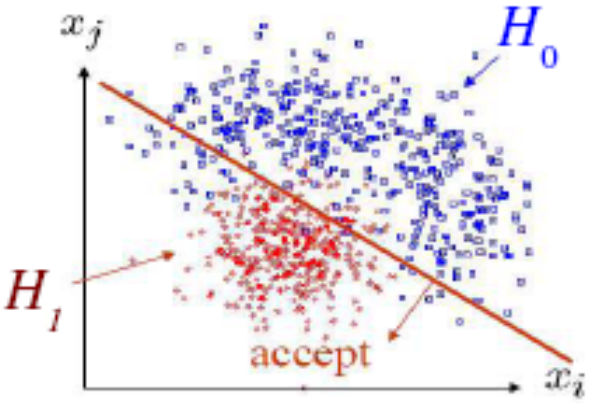
Fisher discriminant function

Setting $\dfrac{\partial J}{\partial w_i} = 0$ gives Fisher's linear discriminant function:

$$y(\vec{x}) = \vec{w}^T \vec{x} \qquad \text{with } \vec{w} \propto W^{-1}(\vec{\mu}_0 - \vec{\mu}_1)$$

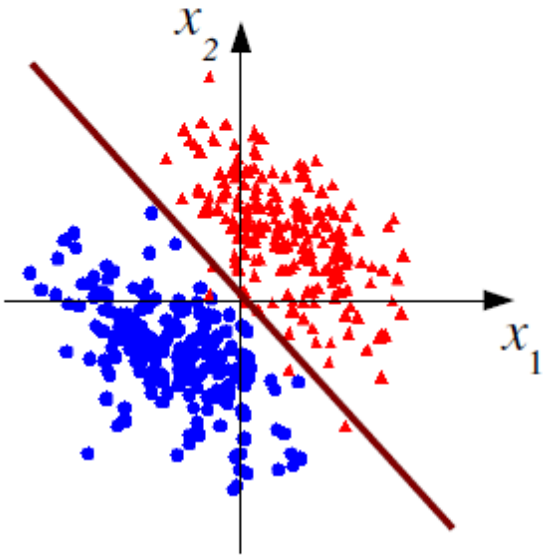
Gives linear decision boundary.

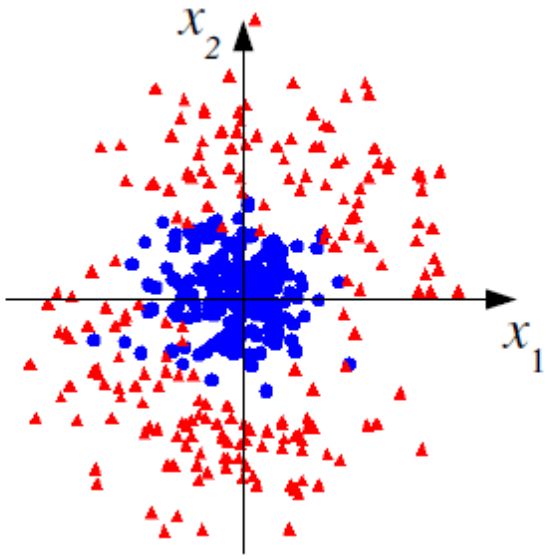
Projection of points in direction of decision boundary gives maximum separation.



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.

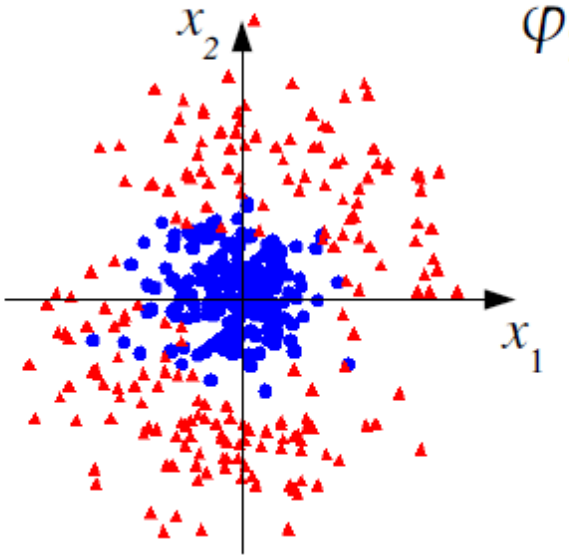
Nonlinear transformation of inputs

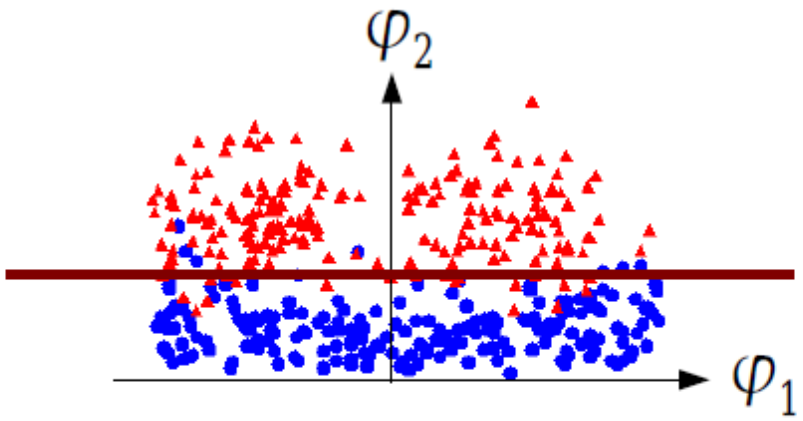
We can try to find a transformation, $x_1, \ldots, x_n \to \varphi_1(\vec{x}), \ldots, \varphi_m(\vec{x})$ so that the transformed "feature space" variables can be separated better by a linear boundary:

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

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

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.

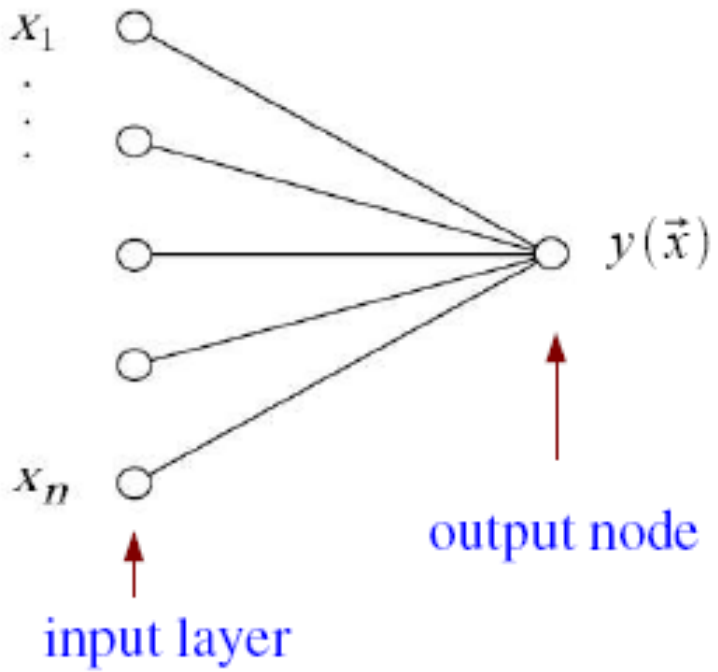
The training data are then used to adjust the parameters so that the resulting discriminant function has the best performance.

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 $y(\boldsymbol{x})$ is equivalent to the original linear discriminant. This is an example of a "generalized linear model" called the single layer perceptron.



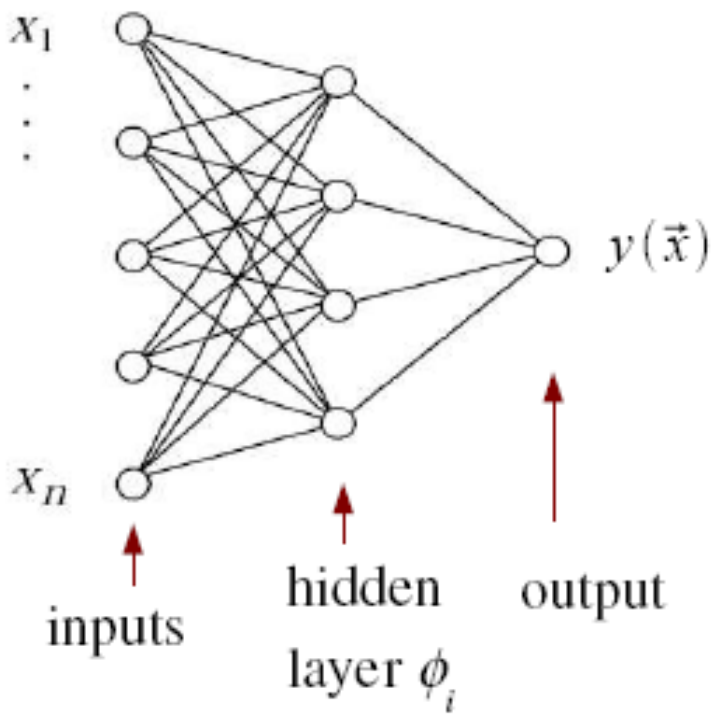
The multilayer perceptron

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

Superscript for weights indicates layer number

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

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



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

Network training

The type of each training event is known, i.e., for event *a* we have:

$\vec{x}_a = (x_1, \ldots, x_n)$ the input variables, and

$t_a = 0, 1$ a numerical label for event type ("target value")

Let $\boldsymbol{w}$ denote the set of all of the weights of the network. We can determine their optimal values by minimizing a sum-of-squares "error function"

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

Contribution to error function from each event

Numerical minimization of $E(\boldsymbol{w})$

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

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

learning rate ($\eta>0$)

If we do this with the full error function $E(\boldsymbol{w})$, gradient descent does surprisingly poorly; better to use "conjugate gradients".

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

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

Error backpropagation

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

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

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

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

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

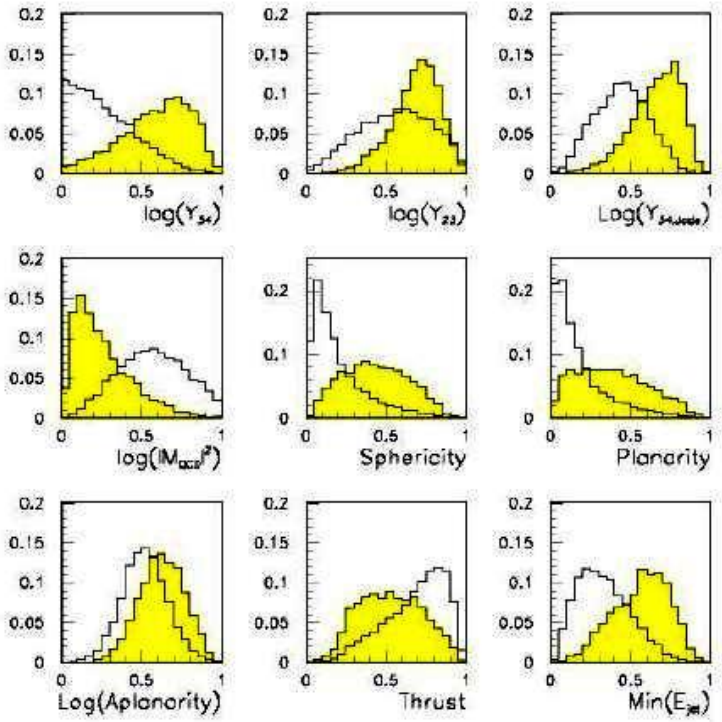
derivative of activation function

Chain rule gives all the needed derivatives.

Neural network example from LEP II

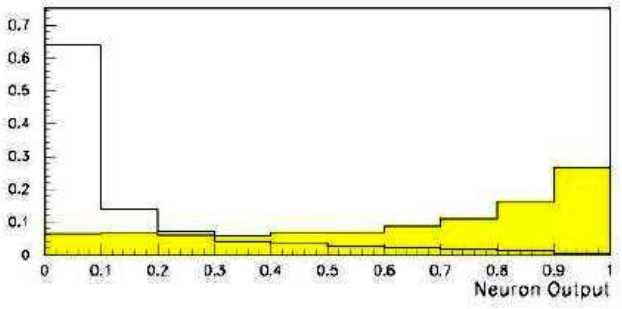
Signal: $e^+e^- \to W^+W^-$ (often 4 well separated hadron jets)

Background: $e^+e^- \to qqgg$ (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)

Some issues with neural networks

In the example with WW events, goal was to select these events so as to study properties of the W boson.

> Needed to avoid using input variables correlated to the properties we eventually wanted to study (not trivial).

In principle a single hidden layer with an sufficiently large number of nodes can approximate arbitrarily well the optimal test variable (likelihood ratio).

> Usually start with relatively small number of nodes and increase until misclassification rate on validation data sample ceases to decrease.

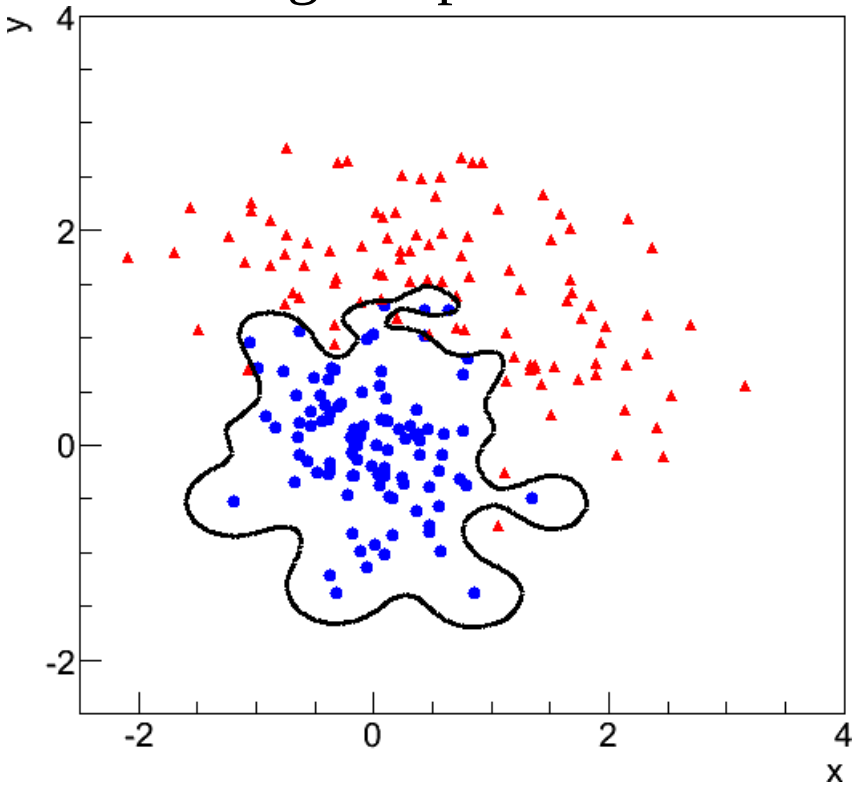
Often MC training data is cheap -- problems with getting stuck in local minima, overtraining, etc., less important than concerns of systematic differences between the training data and Nature, and concerns about the ease of interpretation of the output.

Overtraining

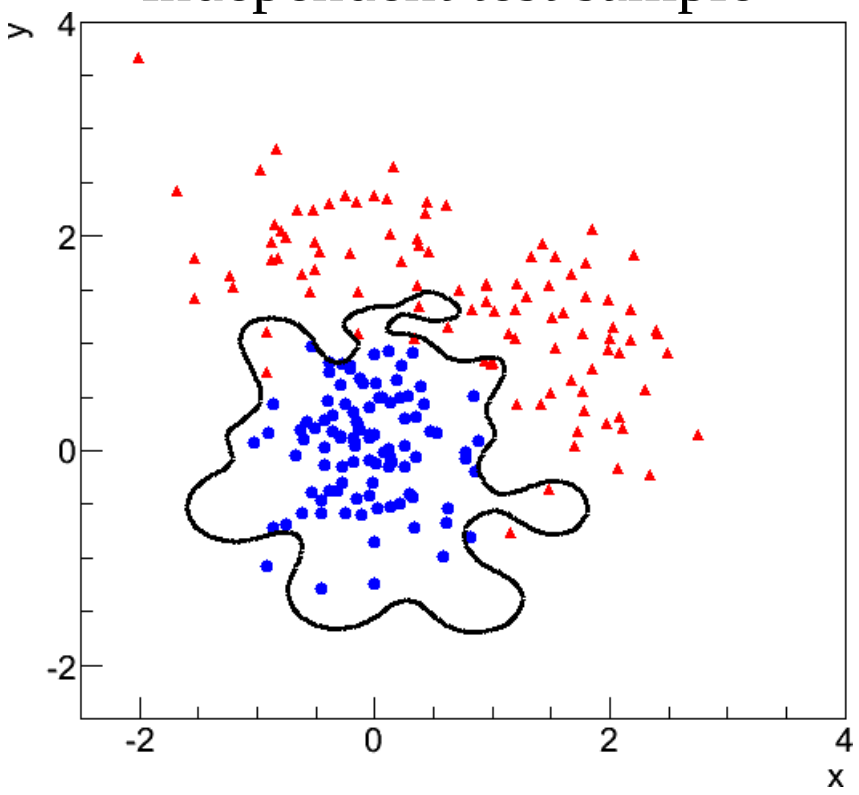
If decision boundary is too flexible it will conform too closely to the training points → overtraining.

Monitor by applying classifier to independent test sample.

training sample

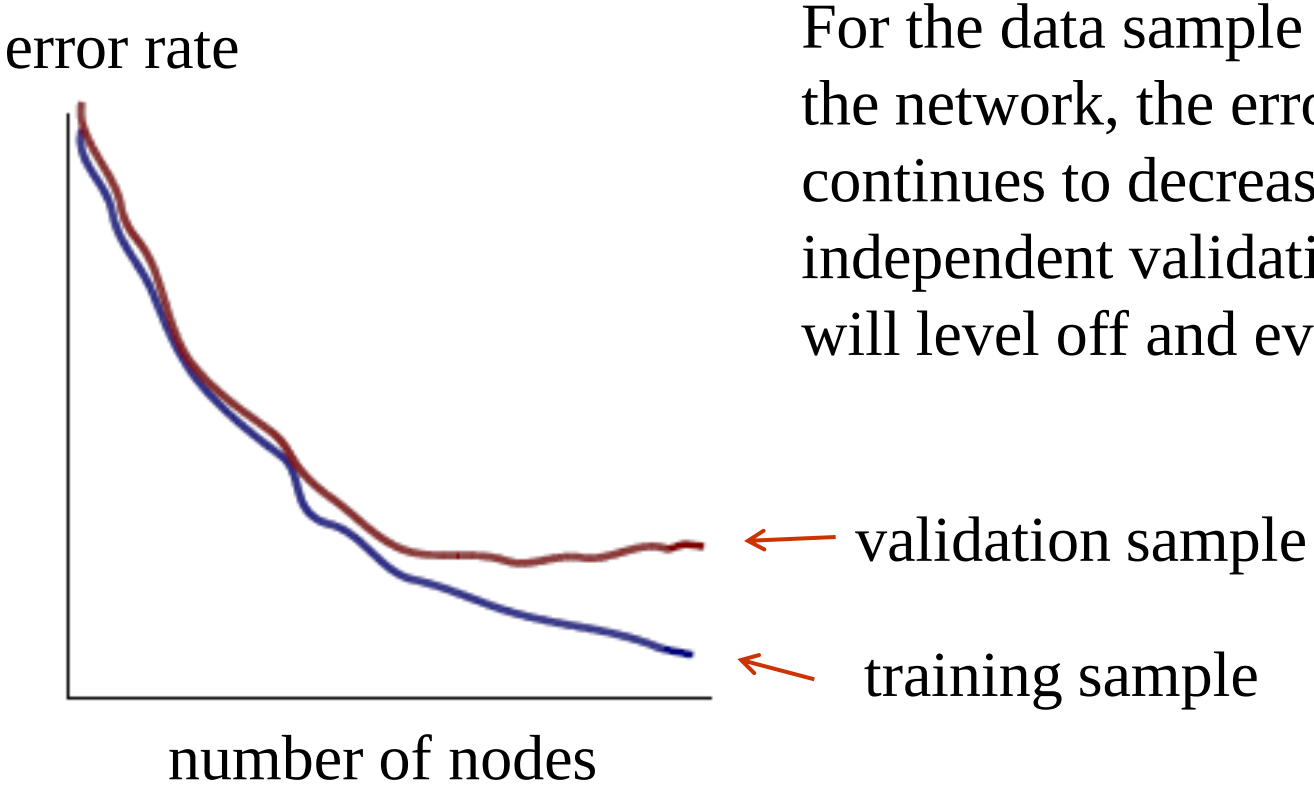


independent test sample



Monitoring overtraining

We can monitor the misclassification rate (or value of the error function) as a function of some parameter related to the level of flexibility of the decision boundary, such as the number of nodes in the hidden layer.



For the data sample used to train the network, the error rate continues to decrease, but for an independent validation sample, it will level off and even increase.

Validation and testing

The validation sample can be used to make various choices about the network architecture, e.g., adjust the number of hidden nodes so as to obtain good "generalization performance" (ability to correctly classify unseen data).

If the validation stage is iterated may times, the estimated error rate based on the validation sample has a bias, so strictly speaking one should finally estimate the error rate with an independent test sample.

	train : validate : test
Rule of thumb if data not too expensive (Narsky):	50 : 25 : 25

But this depends on the type of classifier. Often the bias in the error rate from the validation sample is small and one can omit the test step.

Regularized neural networks

Often one uses the test sample to optimize the number of hidden nodes.

Alternatively one may use a relatively large number of hidden nodes but include in the error function a regularization term that penalizes overfitting, e.g.,

regularization parameter

$$\tilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

Increasing λ gives a smoother boundary (higher bias, lower variance)

Known as "weight decay", since the weights are driven to zero unless supported by the data (an example of "parameter shrinkage").

Probability density estimation methods

If we could estimate the pdfs $p(\boldsymbol{x}|H_0)$, $p(\boldsymbol{x}|H_1)$ for the classes of events we want to separate, then we could form the optimal discriminating function from their ratio:

$$y(\vec{x}) = \frac{p(\vec{x}|H_0)}{p(\vec{x}|H_1)}$$

So the problem reduces to estimating the joint pdfs $p(\boldsymbol{x})$. We may choose different methods for numerator and denominator.

Methods for estimating pdfs can be

parametric, i.e., we have a function $p(\vec{x}\,;\theta_1,\ldots,\theta_m)$

non-parametric, i.e., model independent (e.g. histogram, ...); also contain parameters but they are "local"; not rigidly tied to any model.

Correlation vs. independence

In a general a multivariate distribution $p(\boldsymbol{x})$ does not factorize into a product of the marginal distributions for the individual variables:

$$p(\vec{x}) = \prod_{i=1}^{n} p_i(x_i)$$

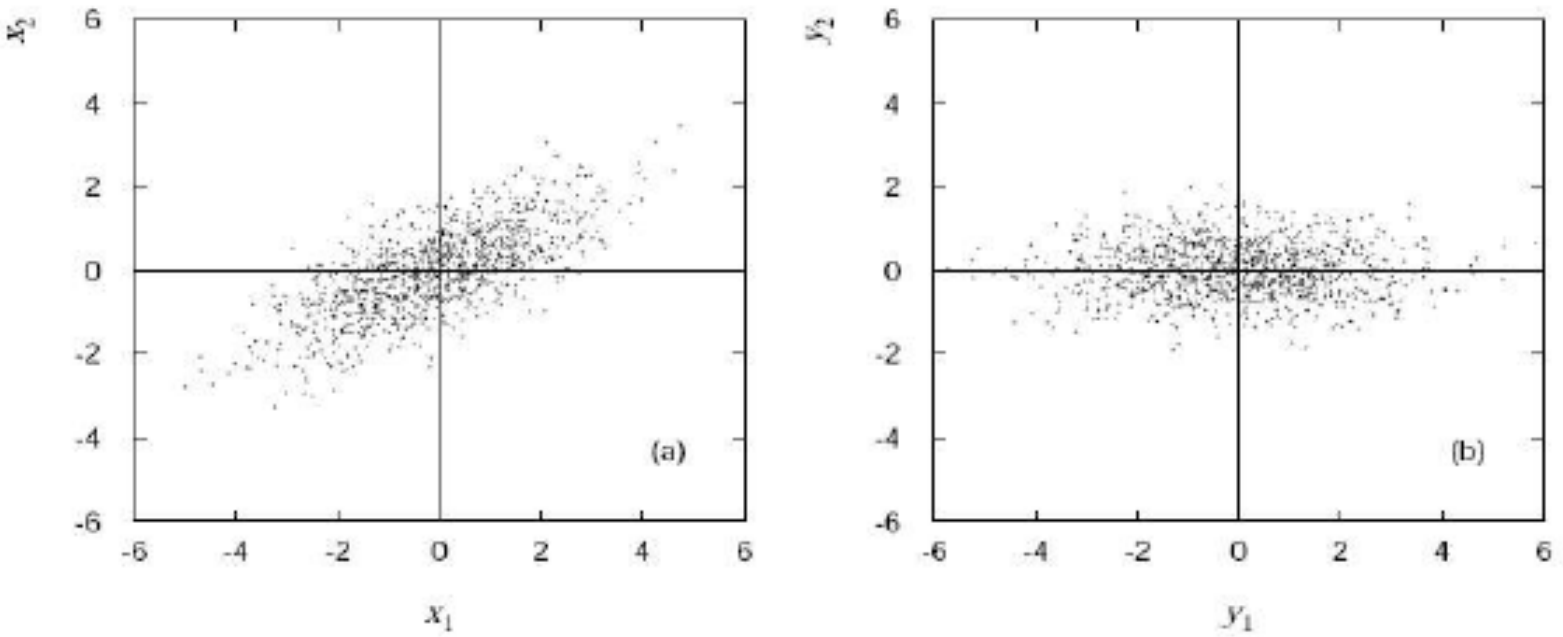
holds only if the components of $\boldsymbol{x}$ are independent

Most importantly, the components of $\boldsymbol{x}$ will generally have nonzero covariances (i.e. they are correlated):

$$V_{ij} = \mathrm{cov}[x_i, x_j] = E[x_i x_j] - E[x_i]E[x_j] \neq 0$$

Decorrelation of input variables

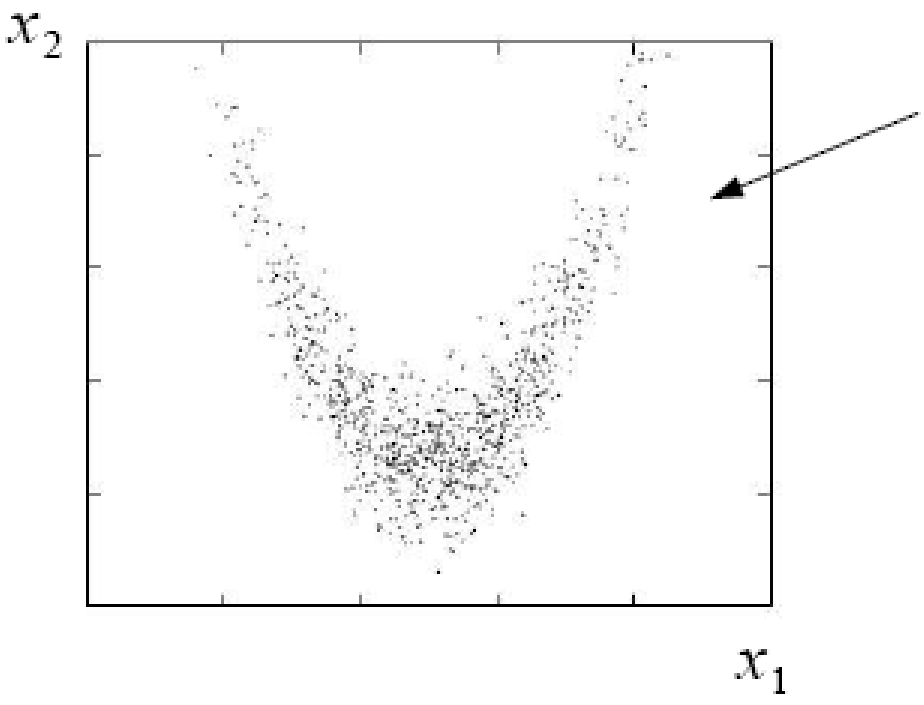
But we can define a set of uncorrelated input variables by a linear transformation, i.e., find the matrix A such that for $\vec{y} = A\vec{x}$ the covariances $\mathrm{cov}[y_i, y_j] = 0$:



For the following suppose that the variables are "decorrelated" in this way for each of $p(\mathbf{x}|H_0)$ and $p(\mathbf{x}|H_1)$ separately (since in general their correlations are different).

Decorrelation is not enough

But even with zero correlation, a multivariate pdf $p(x)$ will in general have nonlinearities and thus the decorrelated variables are still not independent.



pdf with zero covariance but components still not independent, since clearly

$$p(x_2|x_1) \equiv \frac{p(x_1, x_2)}{p_1(x_1)} \neq p_2(x_2)$$

and therefore

$$p(x_1, x_2) \neq p_1(x_1)\, p_2(x_2)$$

Naive Bayes

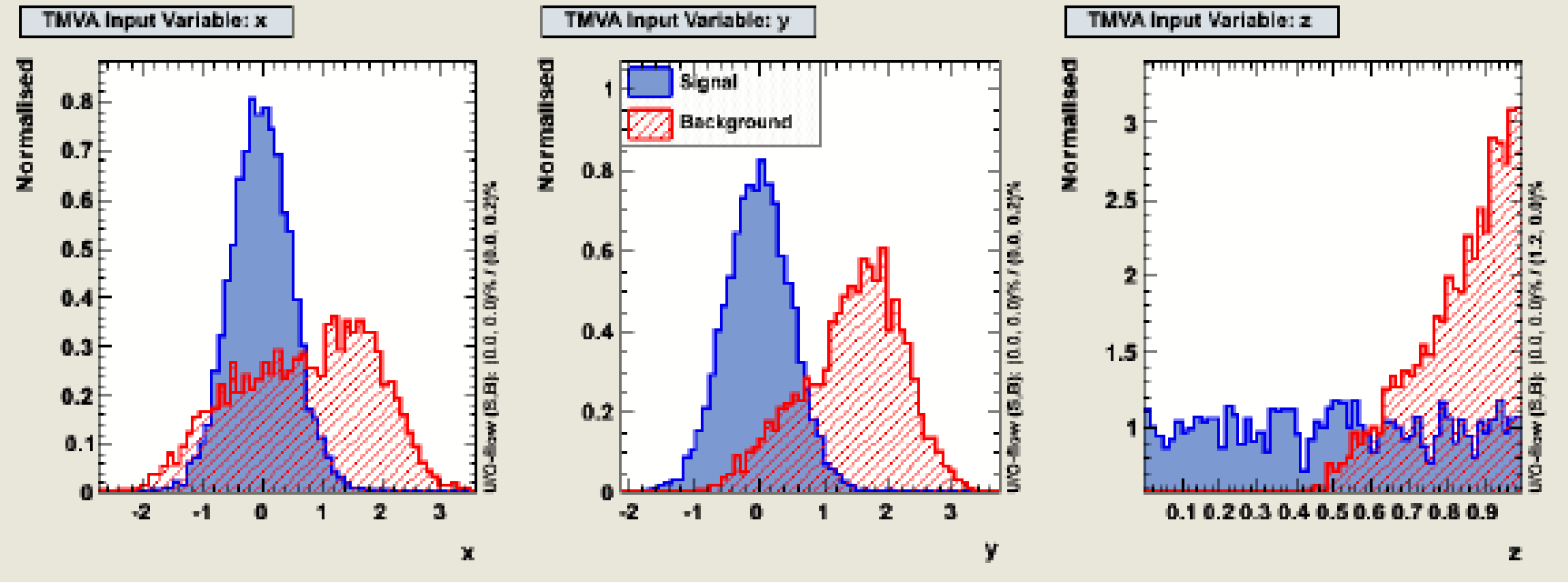
But if the nonlinearities are not too great, it is reasonable to first decorrelate the inputs and take as our estimator for each pdf

$$\hat{p}(\vec{x}) = \prod_{i=1}^{n} \hat{p}_i(x_i)$$

So this at least reduces the problem to one of finding estimates of one-dimensional pdfs.

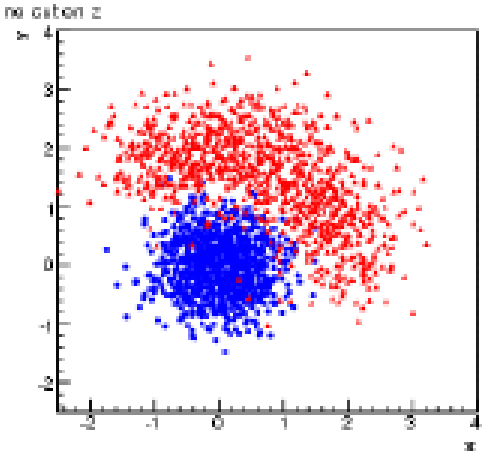
The resulting estimated likelihood ratio gives the Naive Bayes classifier (in HEP sometimes called the "likelihood method").

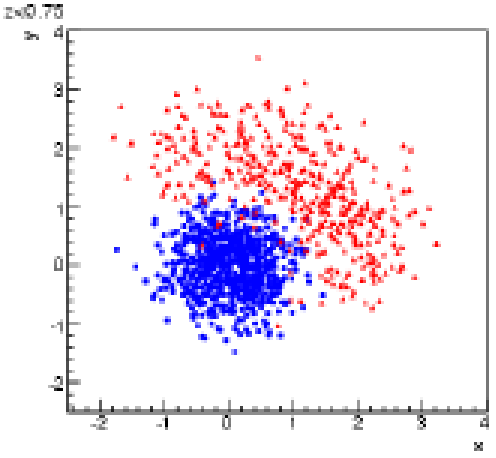
Test example with TMVA



Test example, x vs. y with cuts on z

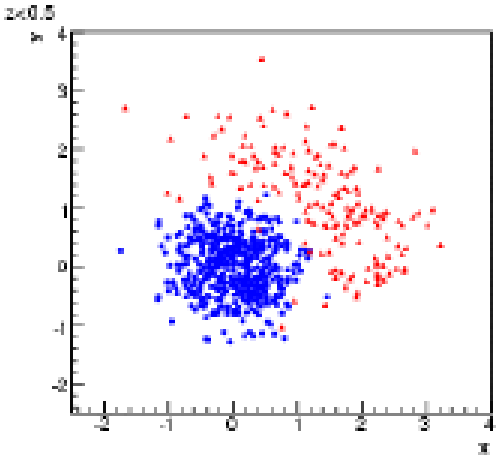
no cut on z

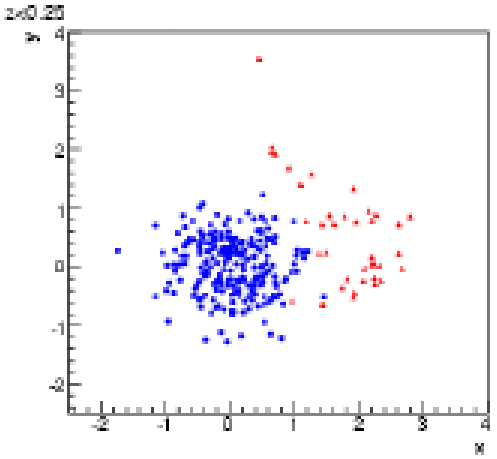




$z < 0.75$

$z < 0.5$

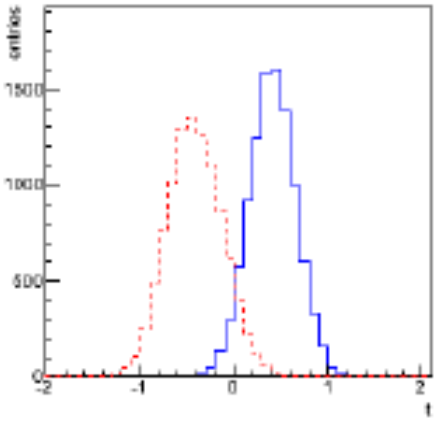


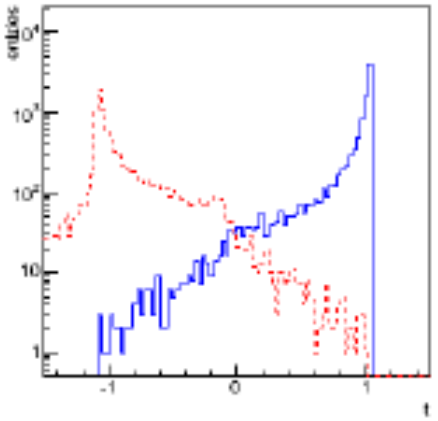


$z < 0.25$

Test example results

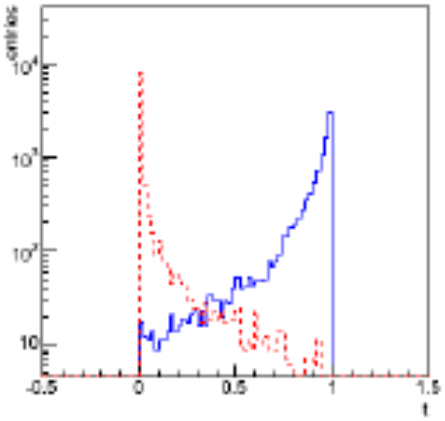
Fisher discriminant

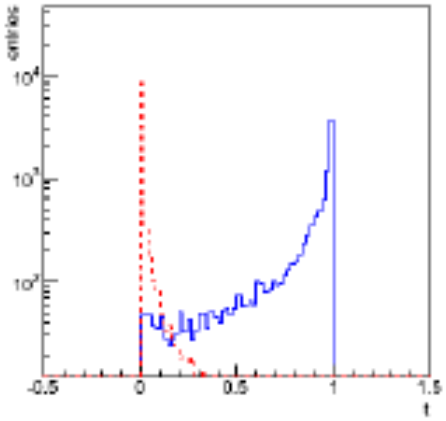




Multilayer perceptron

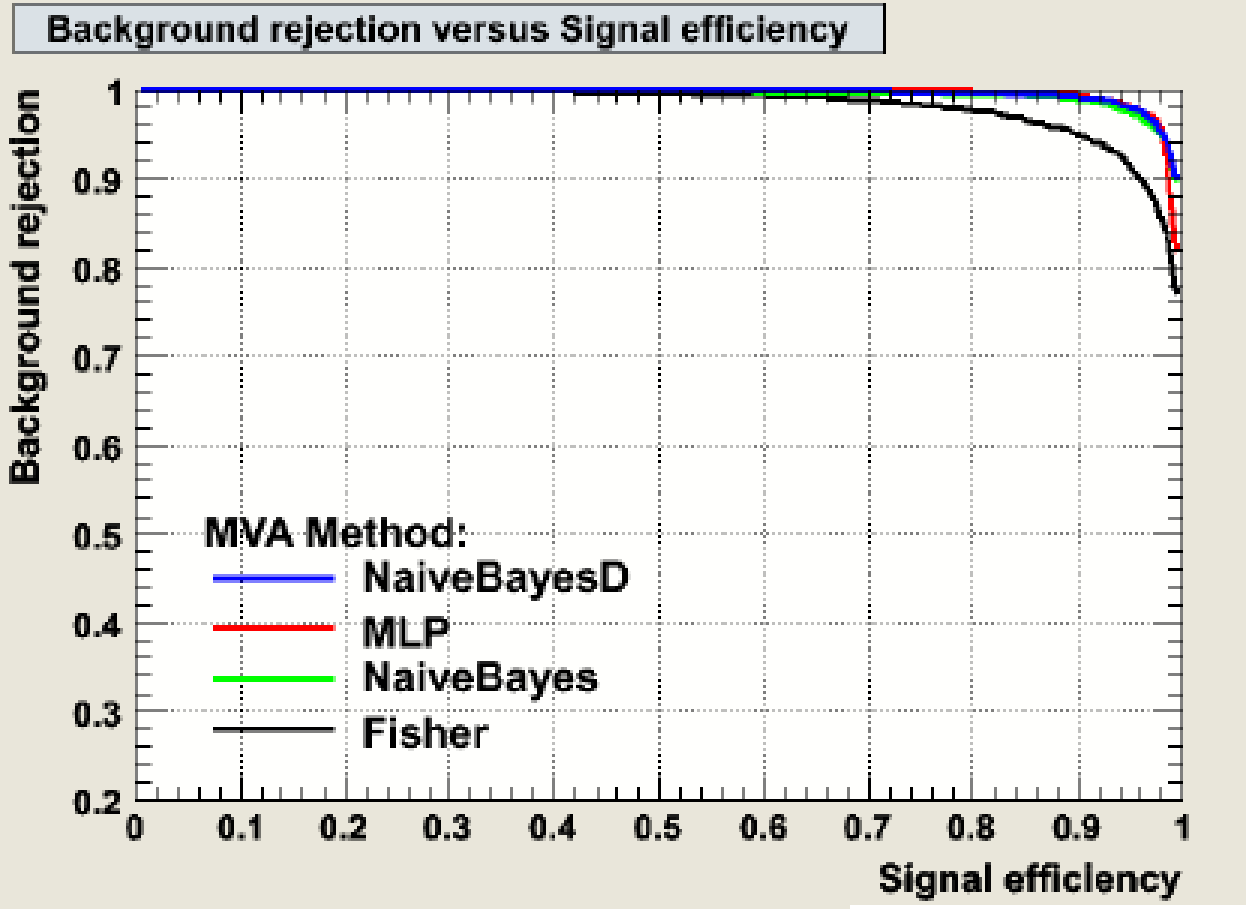
Naive Bayes, no decorrelation





Naive Bayes with decorrelation

Test example ROC curves

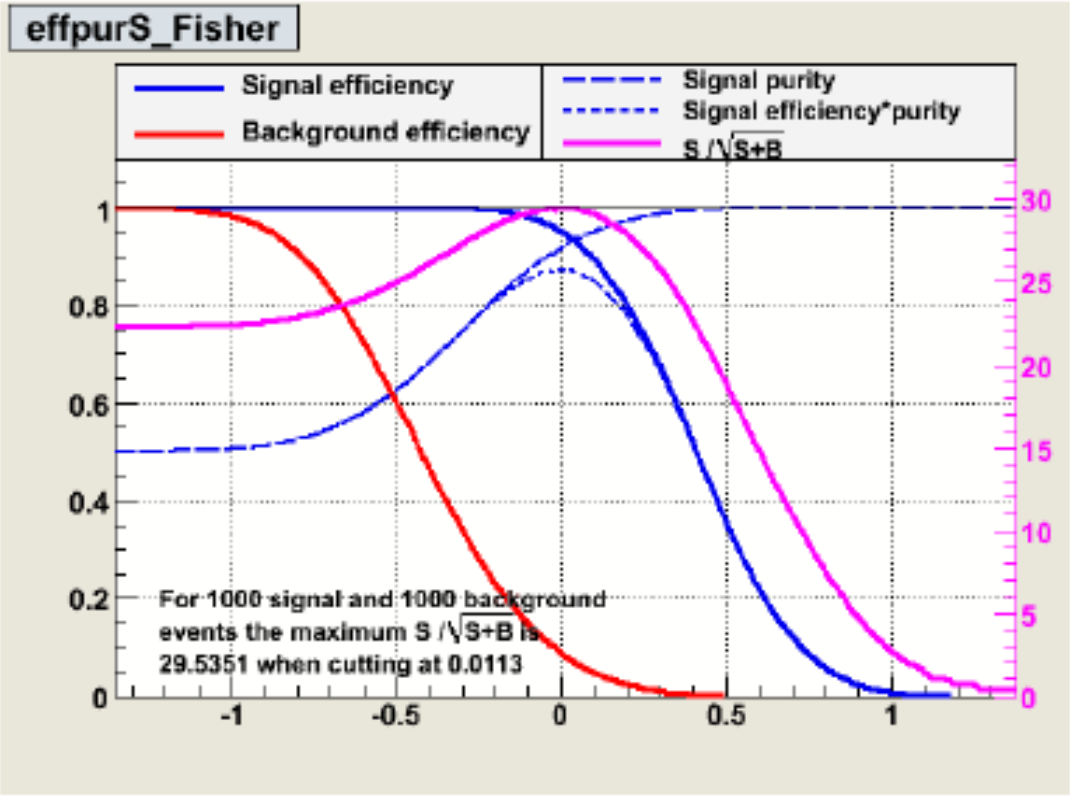


TMVA macro `efficiencies.C`

Efficiencies versus cut value

Select signal by cutting on output: $y > y_{\text{cut}}$

Fisher discriminant



TMVA macro `mvaeffs.C`

Beyond Naive Bayes

Recall that in the naive Bayes approach we approximated the n-dimensional joint pdf as the product of the marginal densities:

$$p(\vec{x}) \approx \prod_{i=1}^{n} p_i(x_i)$$

So the problem is reduced to estimating the one-dimensional marginal pdfs $p_i(x_i)$, usually straightforward.

But this does not capture the higher order nonlinearities of $p(\boldsymbol{x})$.

Lancaster models

Lancaster models approximate an n-dimensional joint pdf $p(\boldsymbol{x}) = p(x_1,\ldots,x_n)$ in terms of the marginal distributions for up to a certain number s of the n components.

For $s = 1$ this was Naive Bayes:. For e.g. $s = 2$ we approximate $p(\boldsymbol{x})$ in terms of one- and two-dimensional marginal densities $p_i(x_i)$ and $p_{ij}(x_i,x_j)$ as

$$p(\vec{x}) \approx \left[\sum_{i,j,i<j} \frac{p_{ij}(x_i, x_j)}{p_i(x_i)\, p_j(x_j)} - \left[\binom{n}{2} - 1 \right] \right] \prod_{i=1}^{n} p_i(x_i)$$

This will not capture the full nonlinear structure of $p(\boldsymbol{x})$ but goes further in that direction than assuming independence. (cf. Webb Ch. 3)

Summary

Information from many variables can be used to distinguish between event types.

Try to exploit as much information as possible.
Try to keep method as simple as possible.
Often start with: cuts, linear classifiers
And then try less simple methods: neural networks

Tomorrow we will see some more multivariate classifiers:

Probability density estimation methods
Boosted Decision Trees
Support Vector Machines