Statistics for Particle Physicists
Lecture 4: Introduction to Machine Learning

Summer Student Lectures
CERN
4 – 7 July 2023

https://indico.cern.ch/event/1254879/timetable/

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

Outline

Lecture 1: Introduction, probability,

Lecture 2: Parameter estimation

Lecture 3: Hypothesis tests

→ Lecture 4: Introduction to Machine Learning

(some exercises on ML here).

What Machine Learning is

The term Machine Learning (ML) refers to algorithms that "learn from data" and make predictions based on what has been learned.

In its simplest sense, "learning" means the algorithm contains adjustable parameters whose values are estimated using data.

Formally, curve fitting can be seen as Machine Learning.
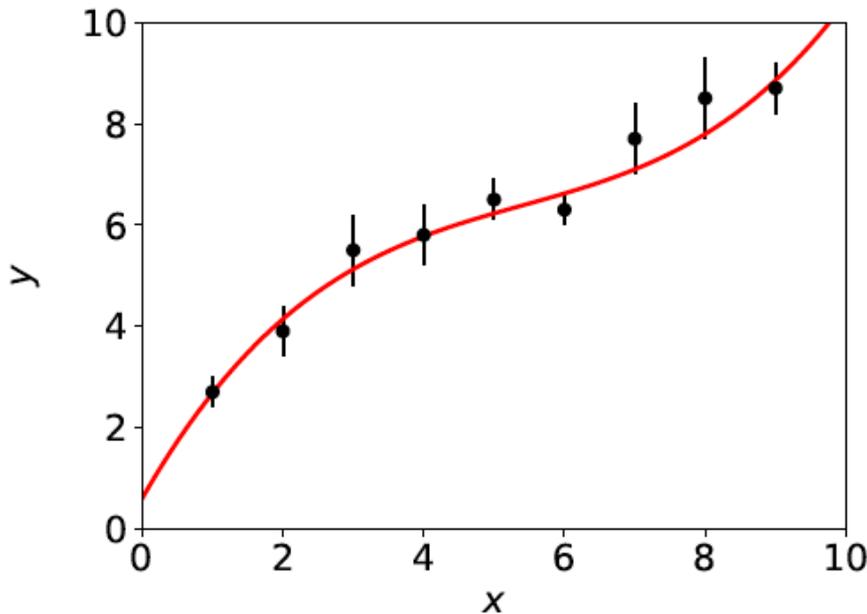
Hypothesized curve:

$$f(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

Values of parameters are "learned" from the data.

Fitted curve can make predictions at x values of future measurements.

More general Machine Learning

But generally ML refers to situations in which:

the hypothesized model is very general (e.g., not just a polynomial) and contains many adjustable parameters;

the quantity we want to predict could depend on many variables, e.g., not just x but a set $\boldsymbol{x} = (x_1,...,x_n)$.

ML can be seen as a part of or related to:

Artificial Intelligence

Pattern Recognition

Statistical Learning

Multivariate Analysis

Development from (mainly) Computer Science, (also) Statistics; sometimes "Data Science" used to refer to all of above.

Curve fitting → regression

The generalisation of curve fitting with a multidimensional control variable $x \rightarrow \boldsymbol{x} = (x_1,..., x_n)$ is called multiple regression.

The data consist of sets of points $(\boldsymbol{x_i}, y_i)$, where now $\boldsymbol{x_i}$ is a multidimensional vector, and usually the y_i does not come with an error bar.

Goal is to predict the expected y value for a new $\boldsymbol{x}$.

https://web.stanford.edu/~hastie/ISLR2/ISLRv2_website.pdf

E.g. in two dimensions, multiple regression means fitting a surface $f(x_1, x_2)$ to data points $(x_{1,i}, x_{2,i}, y_i)$:
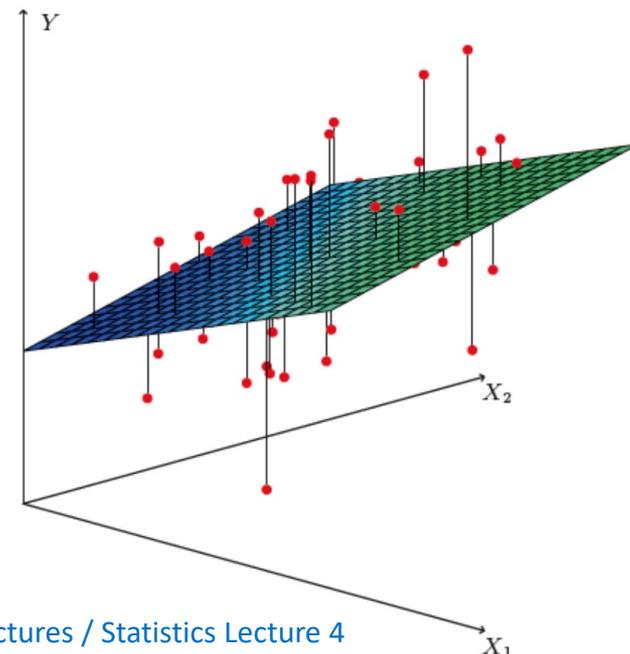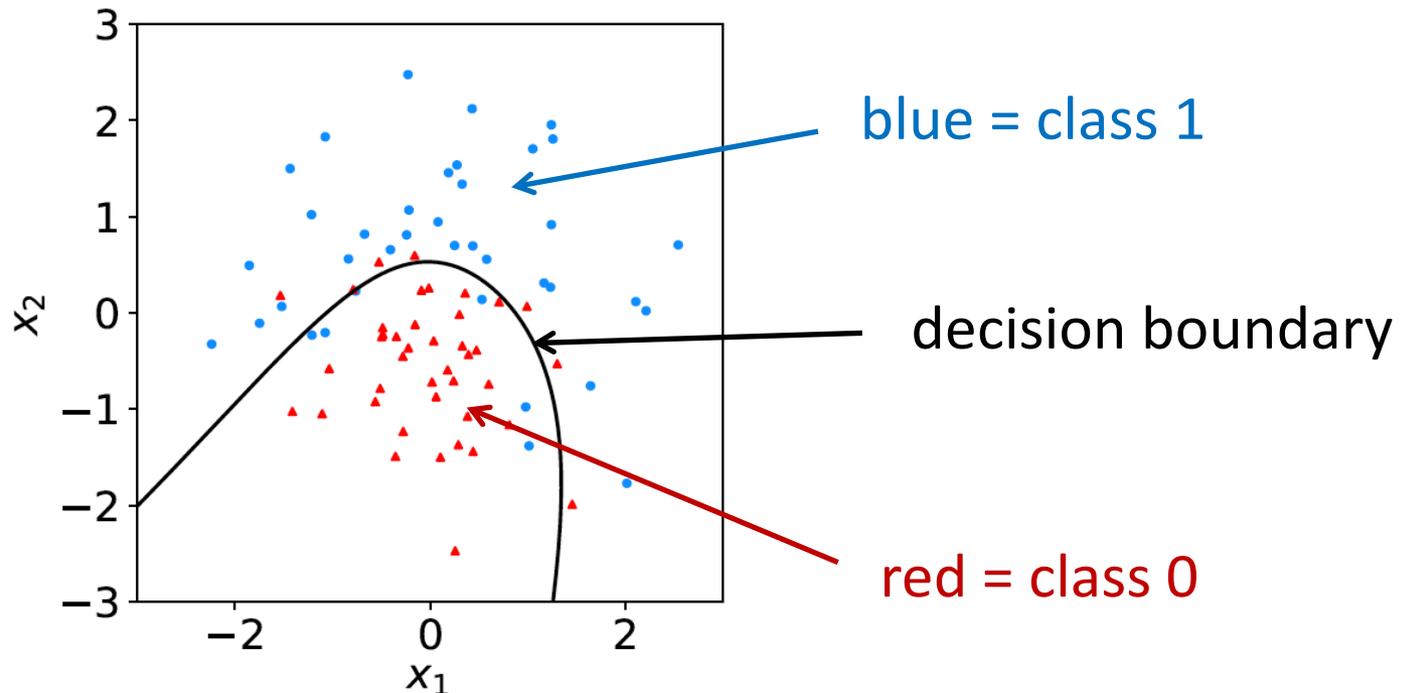
Classification

A related type of ML algorithm is called classification: the data consist of points (x_i, y_i), where now y_i is discrete class label, (e.g., 0 or 1, red or blue, etc.).

Learning based on events with known y_i = supervised learning.

Goal is to create a decision boundary in x-space so as to predict the class y of a new instance of x.

blue = class 1

decision boundary

red = class 0

Example of classification: Industrial Fishing

You scoop up fish which are of two types:

Sea
Bass

Cod

You examine the fish with automatic sensors and for each one you measure a set of features:

x_1 = length x_4 = area of fins
x_2 = width x_5 = mean spectral reflectance
x_3 = weight x_6 = …

These constitute the "feature vector" $\boldsymbol{x} = (x_1,..., x_n)$.

In addition you hire a fish expert to identify the "true class label" $y = 0$ or 1 (i.e., 0 =sea bass, 1 = cod) for each fish.

Distributions of the features

If we consider only two features $x = (x_1, x_2)$, we can display the results in a scatter plot (red: $y = 0$, blue: $y = 1$).
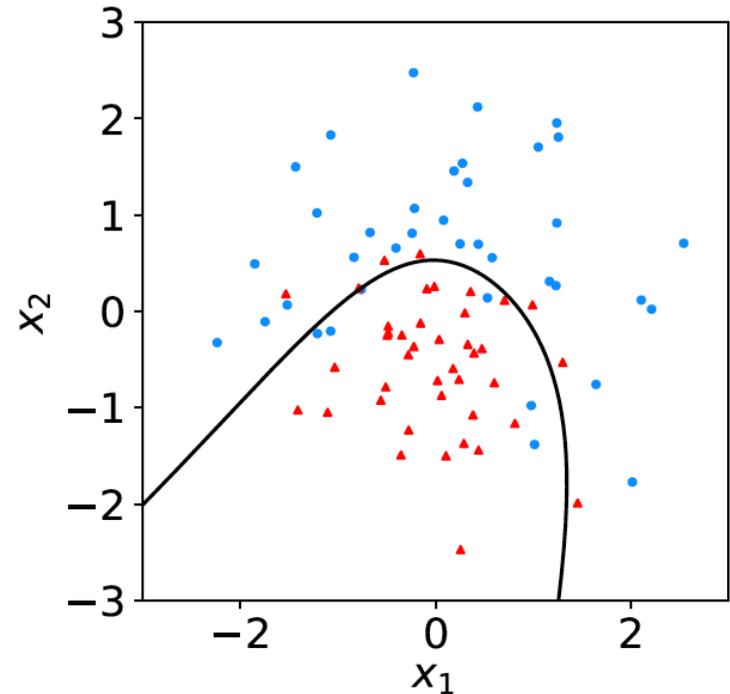
Goal is to determine a decision boundary, so that, without the help of the fish expert, we can classify new fish by seeing where their measured features lie relative to the boundary.

Same idea in multi-dimensional feature space, but cannot represent as 2-D plot. Decision boundary is n-dim. hypersurface.

Example use of Machine Learning: Particle Physics at the LHC

Counter-rotating proton beams in 27 km circumference ring

pp centre-of-mass energy 14 TeV

Detectors at 4 pp collision points:
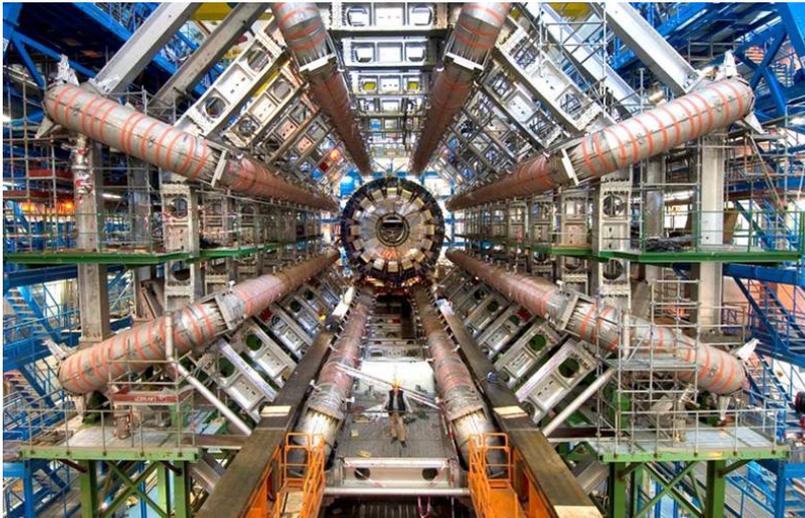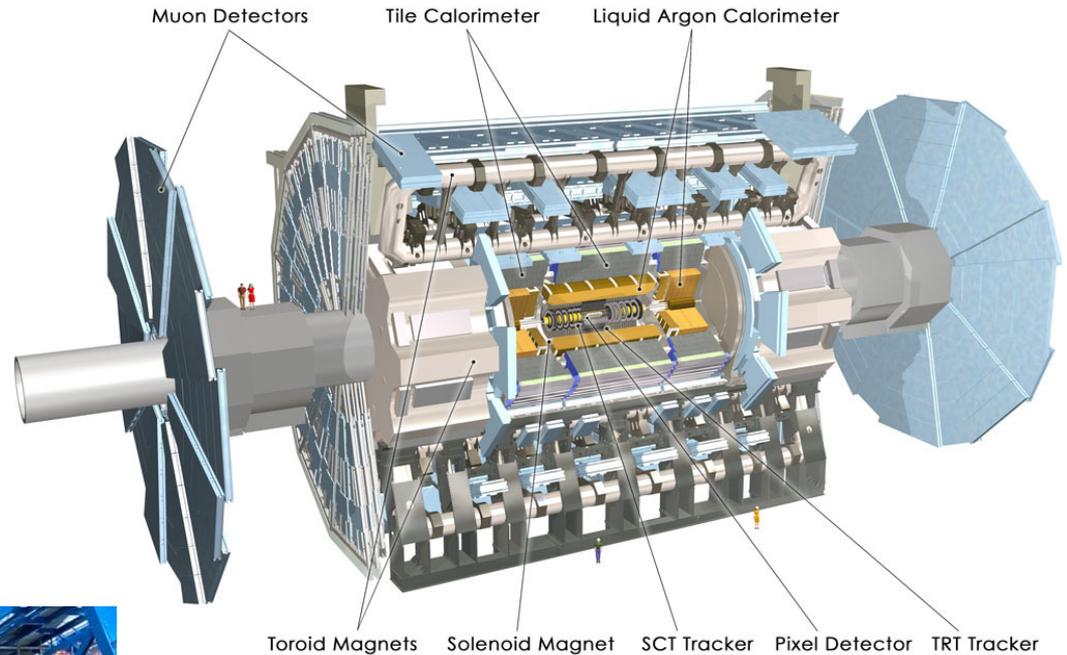   ATLAS
    CMS   general purpose
    LHCb   (b physics)
    ALICE   (heavy ion physics)

The ATLAS Detector at the LHC
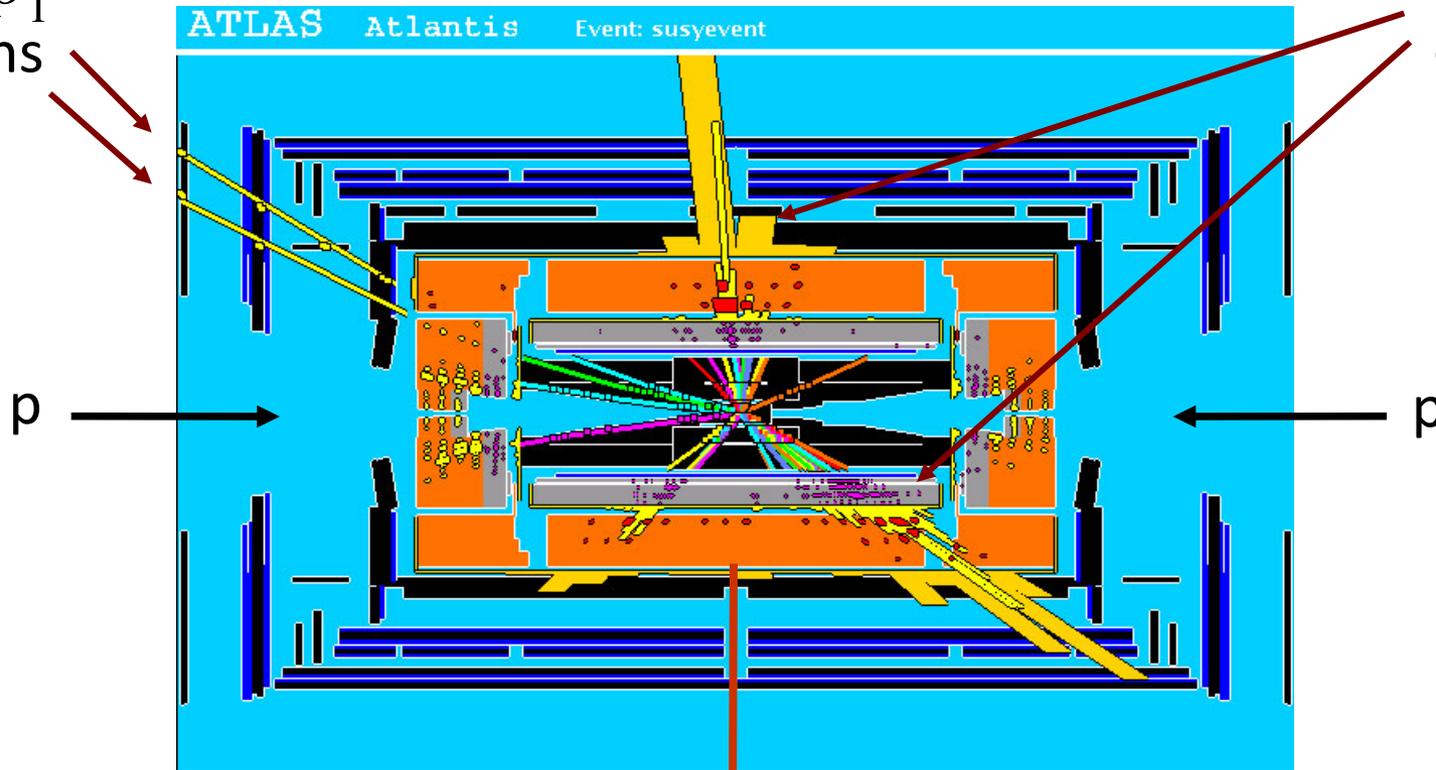

3000 physicists
37 countries
167 universities/labs

Muon Detectors   Tile Calorimeter   Liquid Argon Calorimeter

Toroid Magnets   Solenoid Magnet   SCT Tracker   Pixel Detector   TRT Tracker

25 m diameter
46 m length
7000 tonnes
$\sim 10^8$ electronic channels

A simulated proton-proton collision from supersymmetry ("signal")

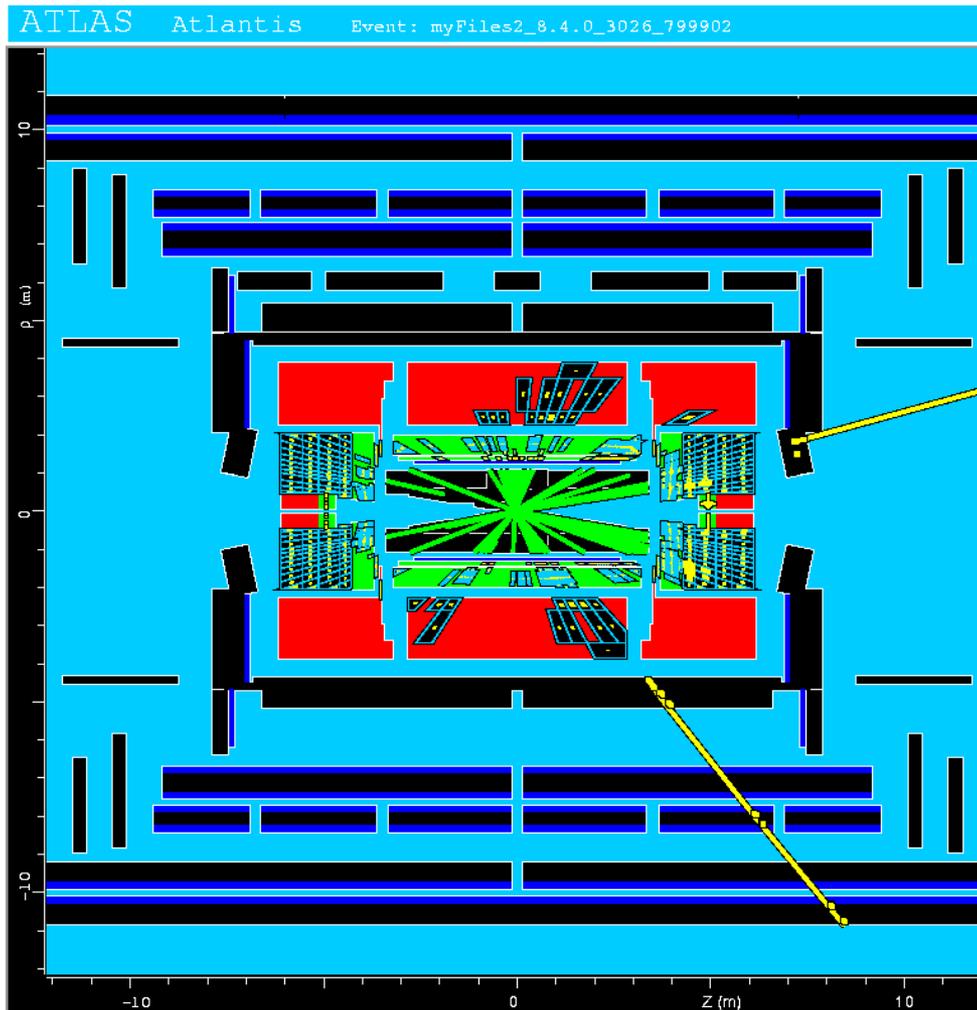high $p_{\rm T}$ muons

high $p_{\rm T}$ jets of hadrons

p

p

missing transverse energy

A simulated proton-proton collision from production of top quarks ("background")

This event has features similar to what we hope to see in supersymmetric events, and thus constitutes a "background" that can mimic the "signal".

Classification of proton-proton collisions

Proton-proton collisions can be considered to come in two classes:

signal (the kind of event we're looking for, $y = 1$)
background (the kind that mimics signal, $y = 0$)

For each collision (event), we measure a collection of features:

x_1 = energy of muon $\qquad$ x_4 = missing transverse energy
x_2 = angle between jets $\qquad$ x_5 = invariant mass of muon pair
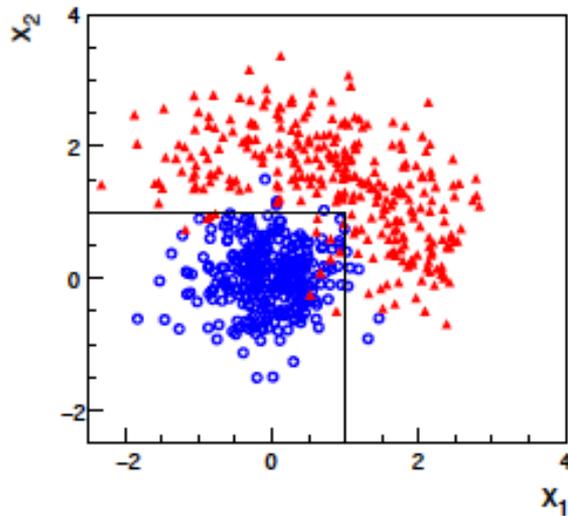x_3 = total jet energy $\qquad$ x_6 = ...

The real events don't come with true class labels, but computer-simulated events do. So we can have a set of simulated events that consist of a feature vector x and true class label y (0 for background, 1 for signal):

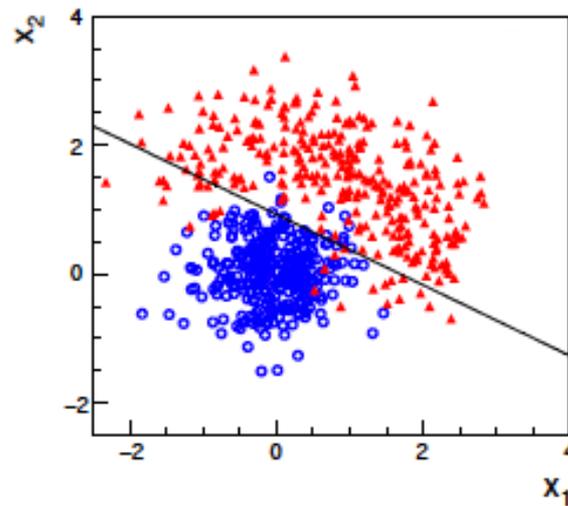$$(x, y)_1, (x, y)_2, ..., (x, y)_N$$

The simulated events are called "training data".
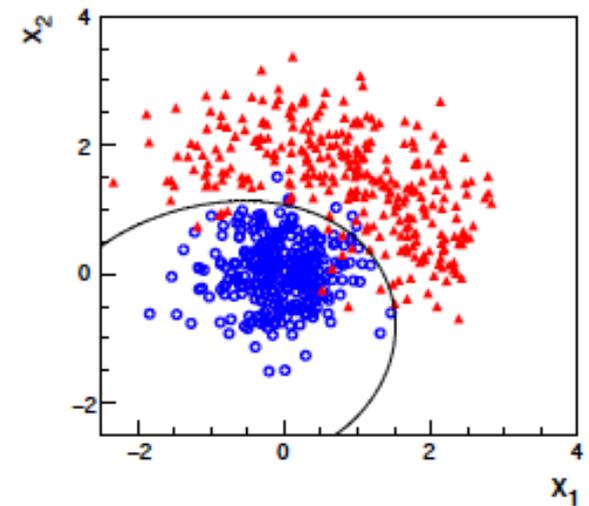
What is the best decision boundary?

"cuts"   linear   nonlinear

What is the best decision function?
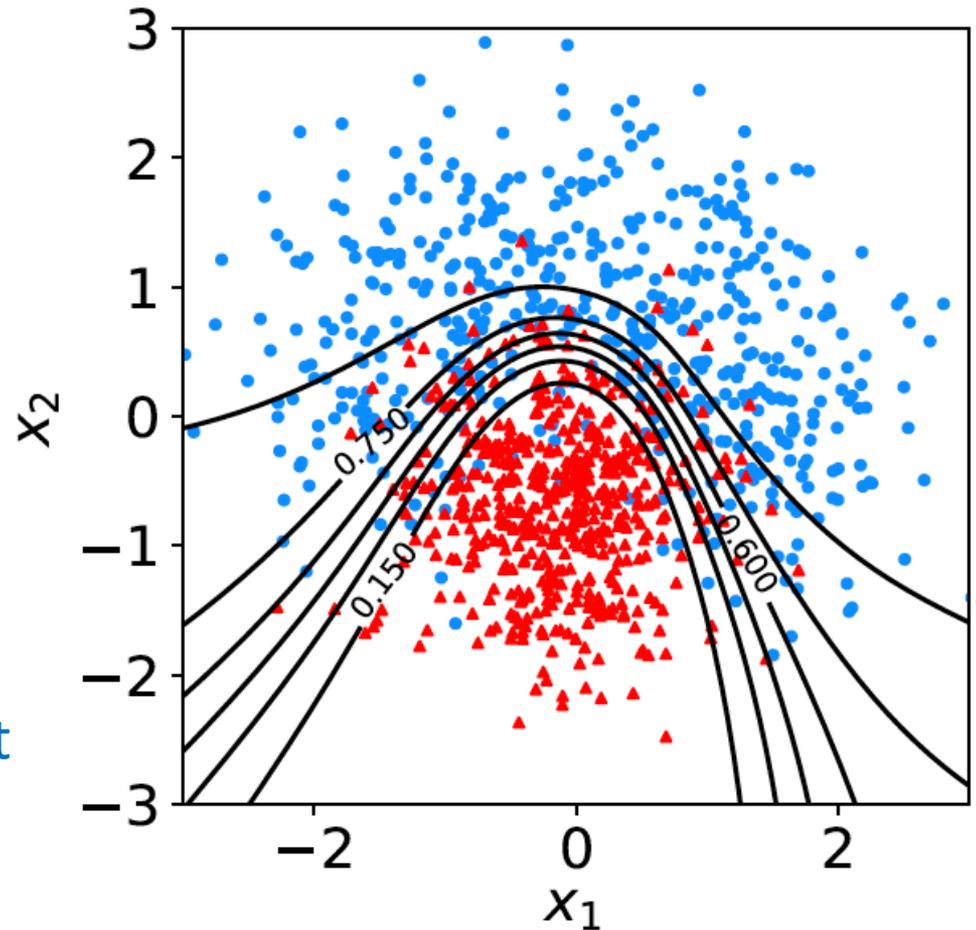
A surface in an n-dimensional space can be described by

$$t(x_1, \ldots, x_n) = t_{\mathrm{c}}$$

scalar function

constant

Different values of the constant t_{c} result in a family of surfaces.

Problem is reduced to finding the best decision function $t(\boldsymbol{x})$.

Linear decision boundary

A simple *Ansatz* is to try a decision function of the form

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

where the coefficients $w_1, ..., w_n$ are constants (or "weights") we need to determine using the training data.

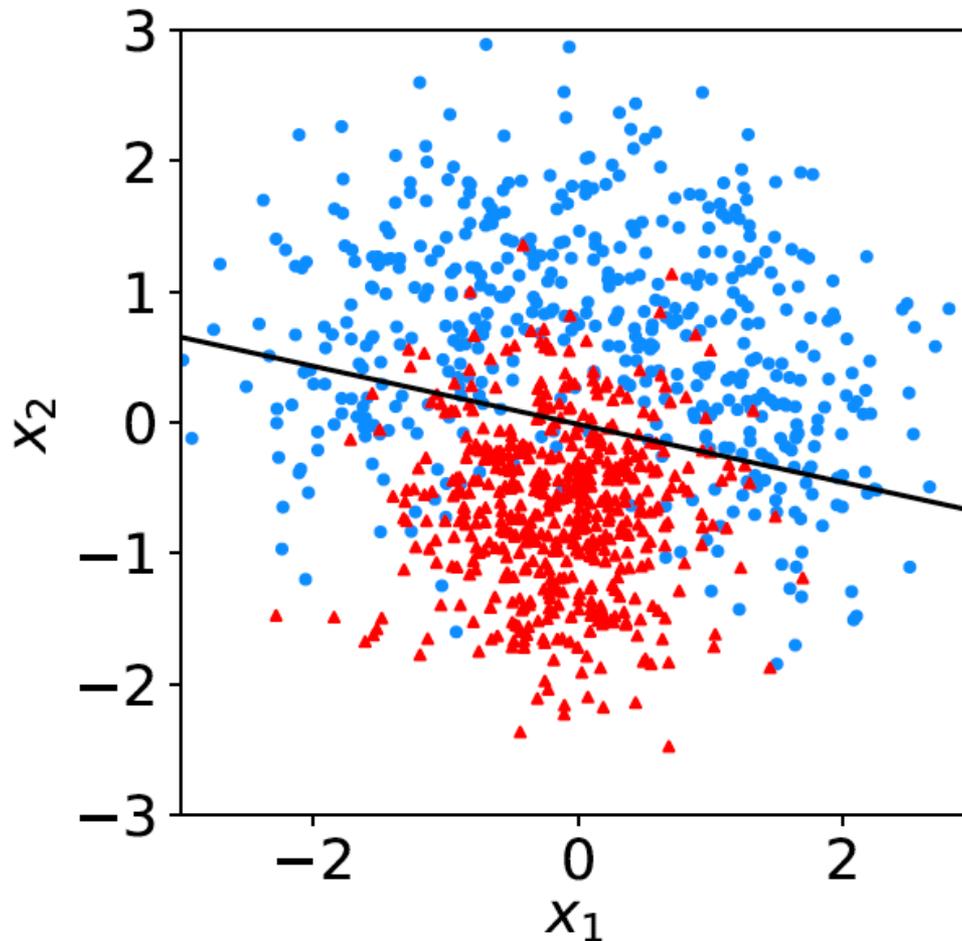A given choice of the weights fixes the function $t(\mathbf{x})$.

Look at the training events from the two classes, for each evaluate $t(\mathbf{x})$ and enter into a histogram.

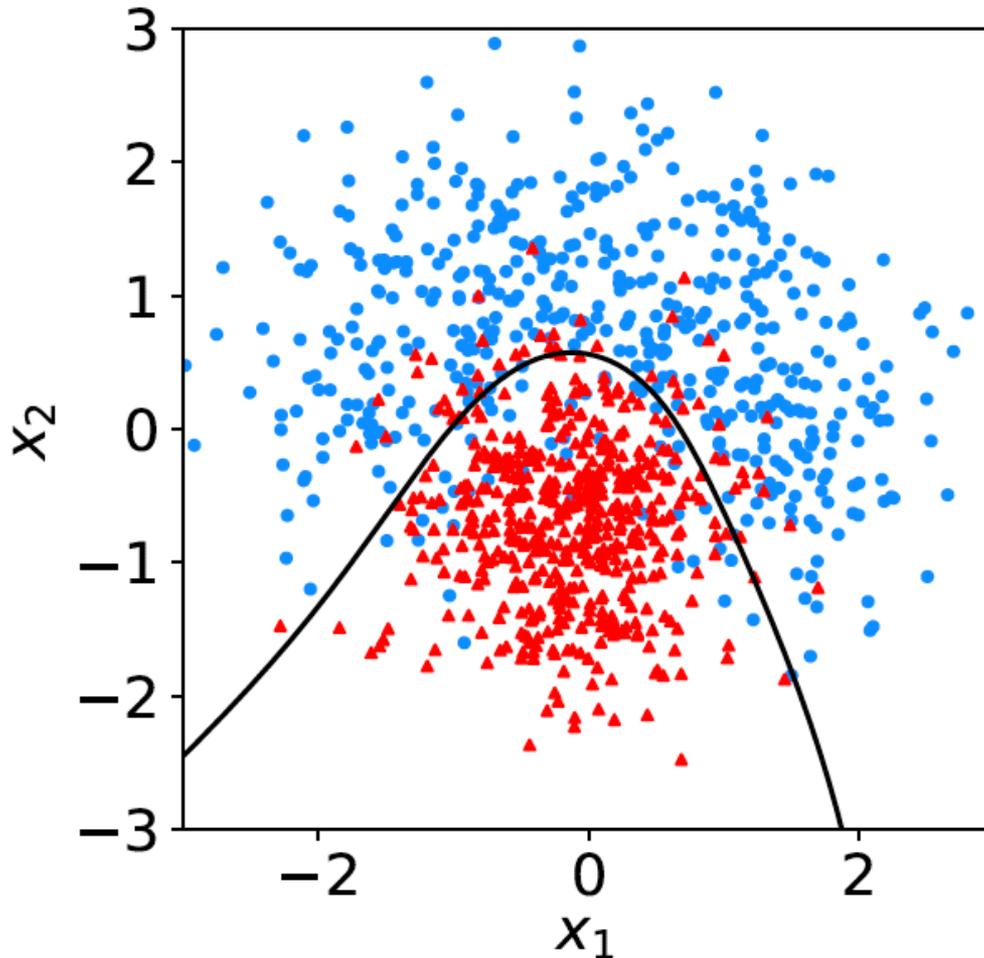Goal is to maximize the "separation" between the two distributions.
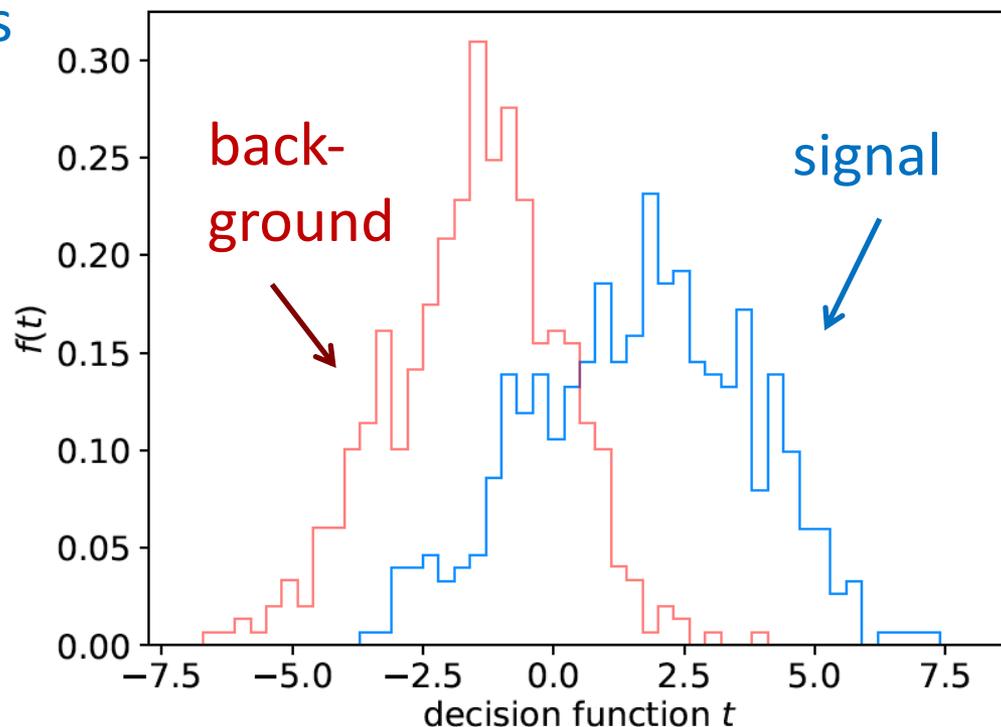
Fisher discriminant

Using a particular definition of what constitutes "best separation" due to R. Fisher one obtains a *Fisher discriminant*:

Nonlinear decision boundaries

From the scatter plot below it's clear that some nonlinear boundary would be better than a linear one:

And to have a nonlinear boundary, the decision function $t(\boldsymbol{x})$ must be nonlinear in $\boldsymbol{x}$.
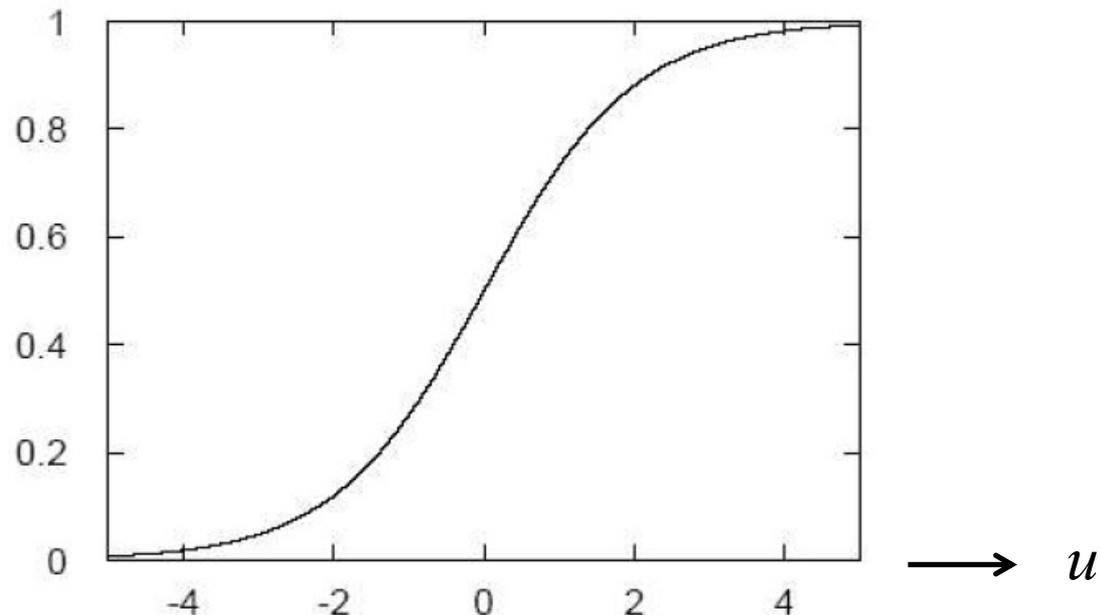
Neural Networks

A simple nonlinear decision function can be constructed as

$$t(\mathbf{x}) = h\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$
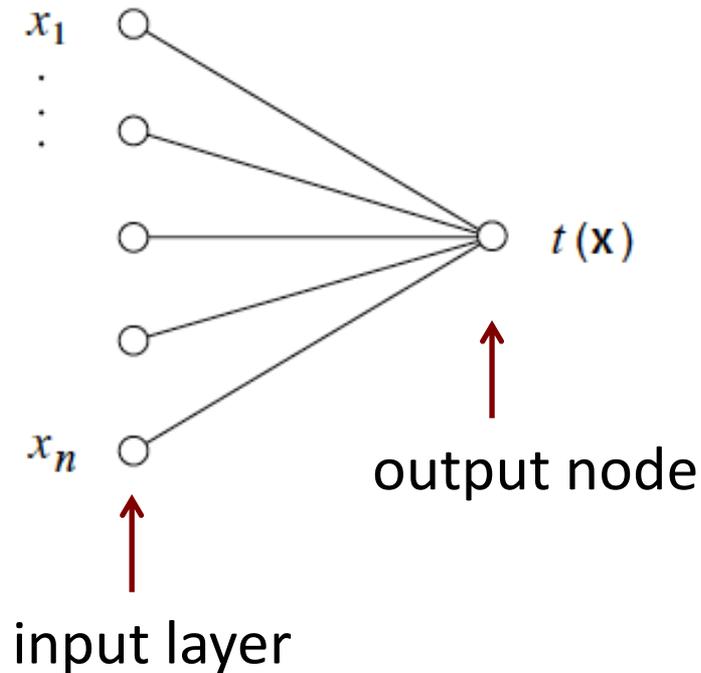
where h is called the "activation function". For this one can use, e.g., a logistic sigmoid function,

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

Single Layer Perceptron

In this form, the decision function is called a Single Layer Perceptron – the simplest example of a Neural Network.

x_1

x_n

$t(\mathbf{x})$

output node

input layer

But the surface described by $t(\boldsymbol{x}) = t_\mathrm{c}$ is the same as by

$$h^{-1}(t(\mathbf{x})) = w_0 + \sum_{i=1}^{n} w_i x_i = h^{-1}(t_\mathrm{c})$$

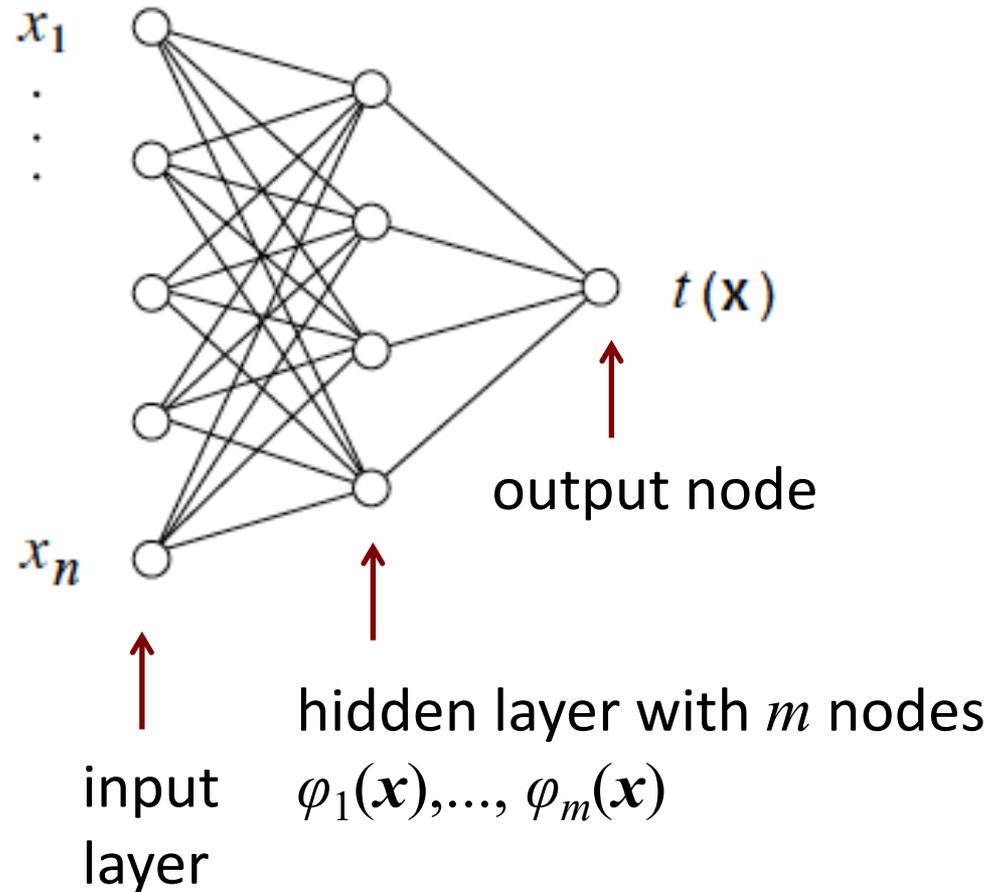So here we still have a linear decision boundary.

Multilayer Perceptron

The Single Layer Perceptron can be generalized by defining first a set of functions $\varphi_i(x)$, with $i = 1,...,m$:

$$\varphi_i(\mathbf{x}) = h\left(w_{i0}^{(1)} + \sum_{j=1}^{n} w_{ij}^{(1)} x_j\right) \qquad i = 1,\ldots,m$$

The $\varphi_i(x)$ are then treated as if they were the input variables, in a perceptron, i.e., the decision function (output node) is

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

Multilayer Perceptron (2)

x_1

x_n

$t(\mathbf{x})$

output node

hidden layer with m nodes
$\varphi_1(\boldsymbol{x}), \ldots, \varphi_m(\boldsymbol{x})$

input
layer

Each line in the graph represents one of the weights $w_{ij}^{(k)}$, which must be adjusted using the training data.

Training a Neural Network

To train the network (i.e., determine the best values for the weights), define a loss function, e.g.,

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} |t(\mathbf{x}_i) - y_i|^2$$

where w represents the set of all weights, the sum is over the set of training events, and y_i is the (numeric) true class label of each event (0 or 1).

The optimal values of the weights are found by minimizing $E(w)$ with respect to the weights (non-trivial algorithms: backpropagation, stochastic gradient descent,...).
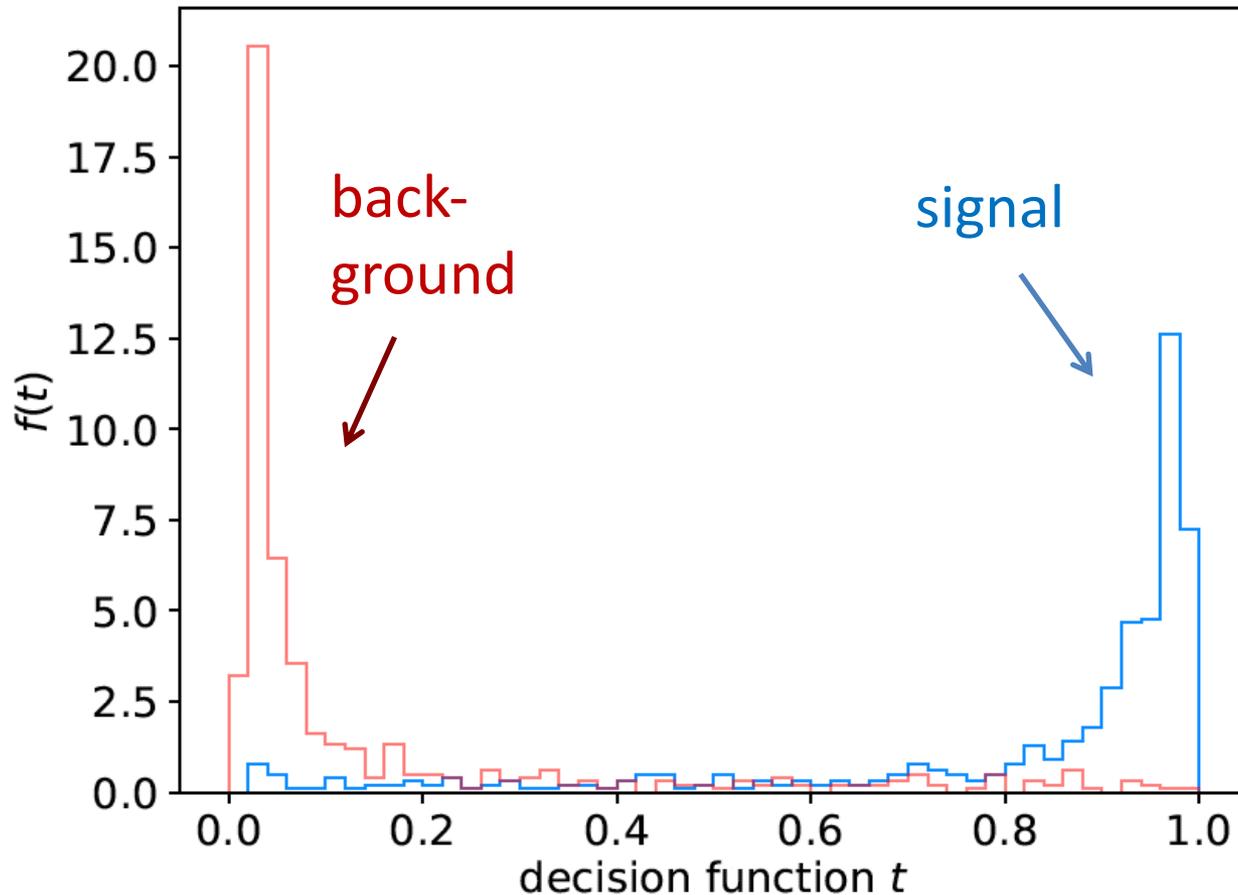
The desired result for an event with feature vector x is:

if the event is of type 0, want $t(x) \sim 0$,
if the event is of type 1, want $t(x) \sim 1$.

Distribution of neural net output

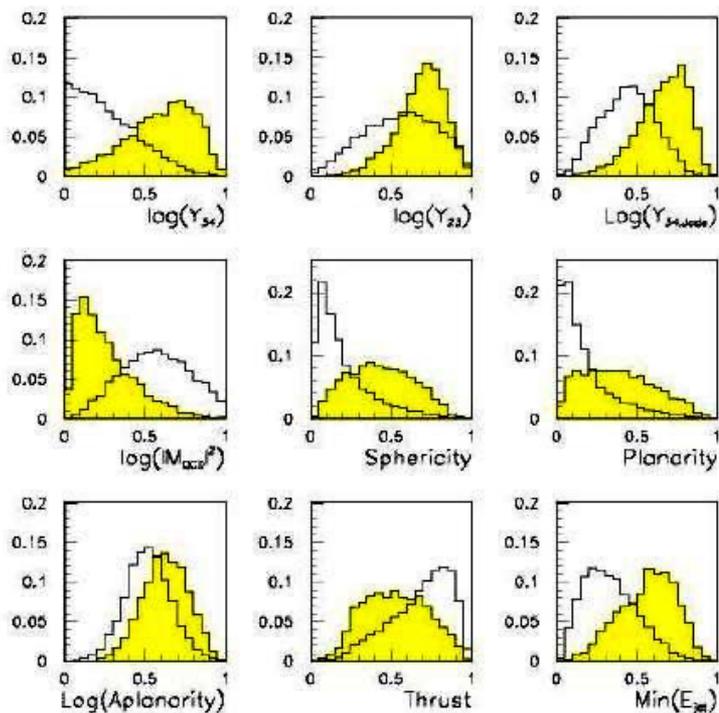Degree of separation between classes now much better than with linear decision function:
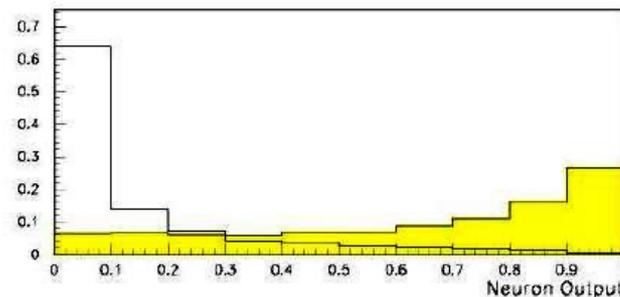

back-ground

signal

Neural network example from LEP II

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

Background: $e^+e^- \rightarrow qqgg$ (4 less well separated hadron jets)

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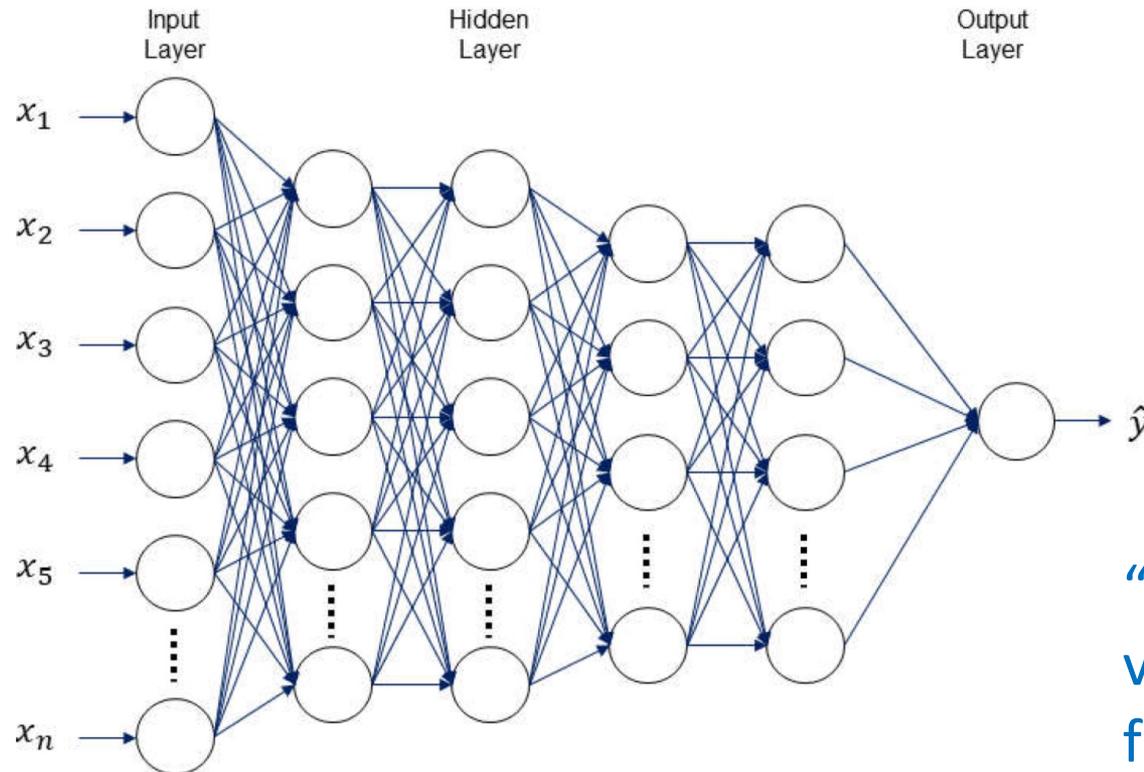none by itself gives much separation.

Neural network output:

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

Deep Neural Networks

The multilayer perceptron can be generalized to have an arbitrary number of hidden layers, with an arbitrary number of nodes in each (= "network architecture").

A "deep" network has several (or many) hidden layers:

"Deep Learning" is a very recent and active field of research.

Other types of classifiers

We have seen only two types of classifiers:

Linear (Fisher discriminant)

Neural Network

There are many others:

Support Vector Machine

Boosted Decision Tree

K-Nearest Neighbour

...

The field is rapidly developing with advances, e.g., that allow one to use feature vectors of very high dimension, such as the pixels of an image.

$\rightarrow$ face/handwriting recognition, driverless cars...

Finally

Four lectures only enough for a brief introduction to:

Probability, frequentist & Bayesian approaches

Parameter estimation, maximum likelihood

Hypothesis tests, p-values, limits

Intro to Machine Learning

Many other important areas:

Treatment of systematic uncertainties (nuisance parameters)

Profile likelihood ratio tests, asymptotics,...

I have put links to some exercises that contain simple python programs on the web pages for lectures 2 (parameter estimation), 3 (hypothesis tests) and 4 (machine learning) – enjoy!
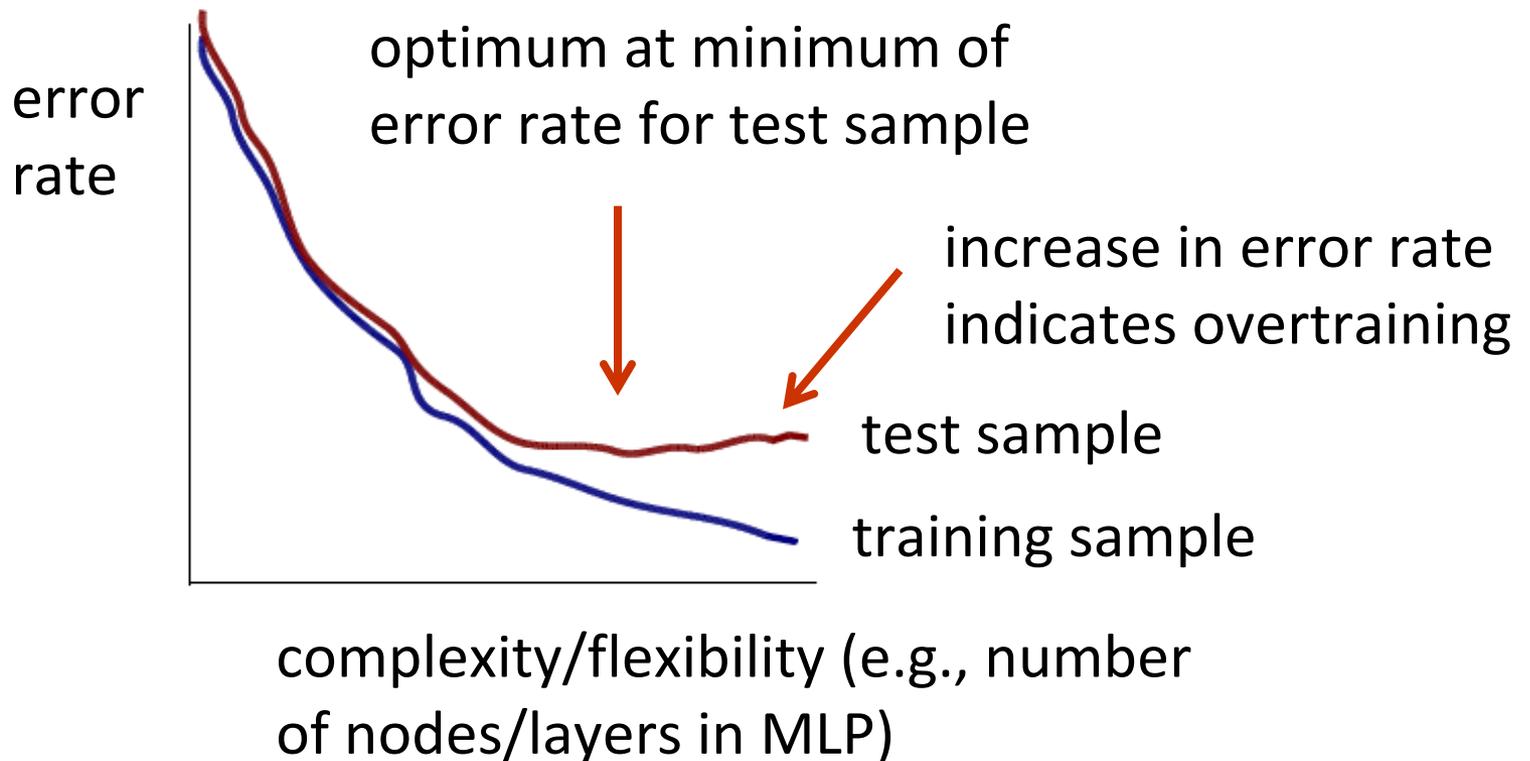
Extra slides

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 ($\rightarrow$ "overtraining").
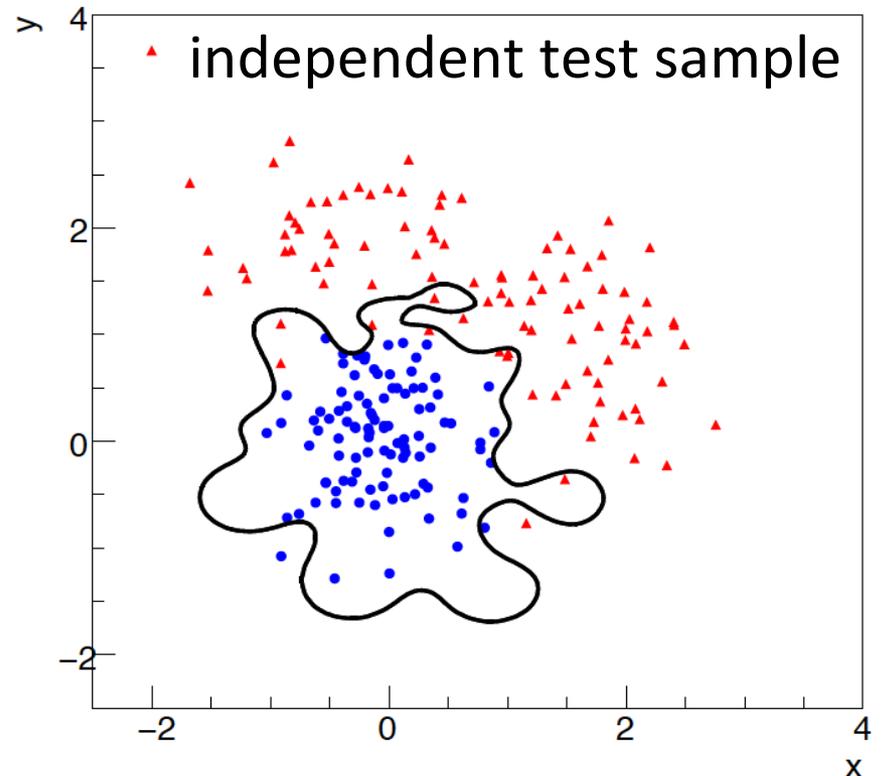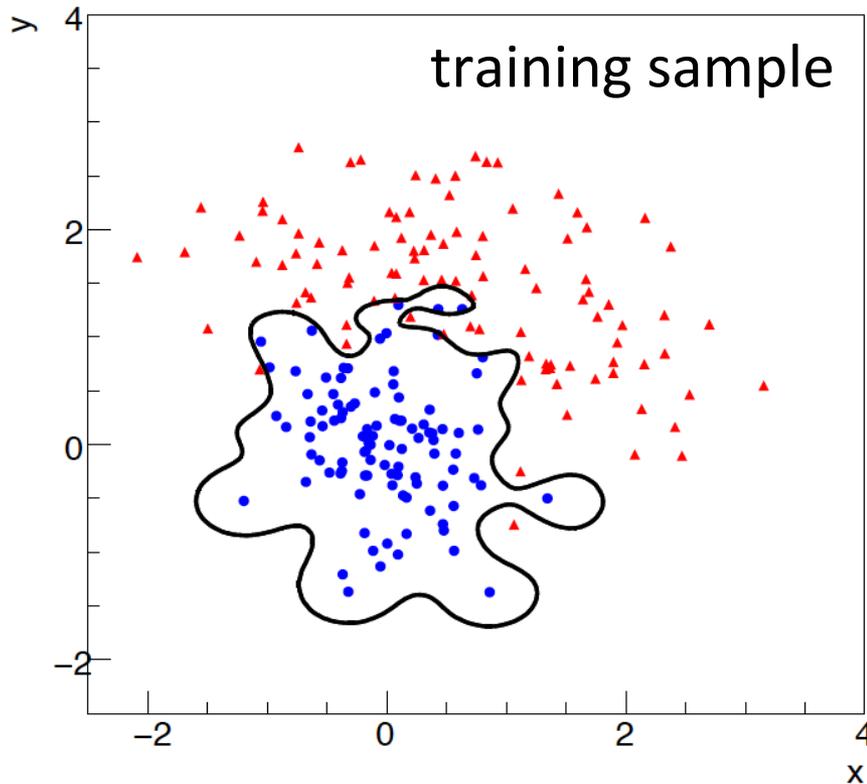
Monitoring overtraining

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

error rate

optimum at minimum of
error rate for test sample

increase in error rate
indicates overtraining

test sample

training sample

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

Machine Learning for handwriting recognition

Initial feature vector = set of pixel values of an image

Scene parsing/labeling with Convolutional Neural Nets

[Farabet et al. ICML 2012, PAMI 2013]

See *Deep Learning and the Future of AI*, seminar at CERN by Yann LeCun: **https://indico.cern.ch/event/510372/**