
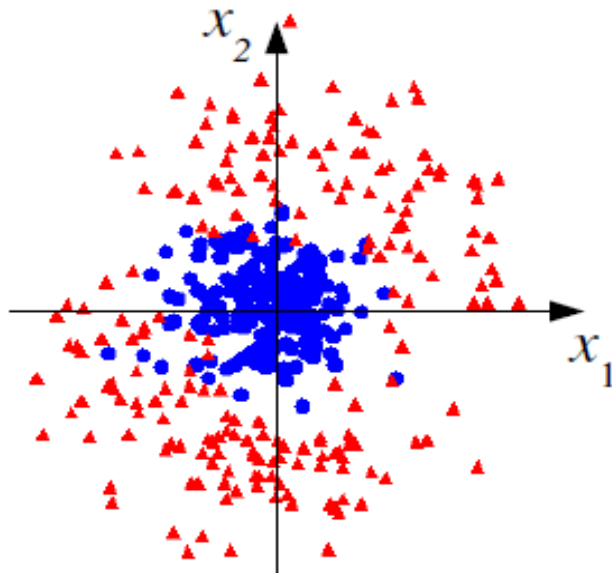
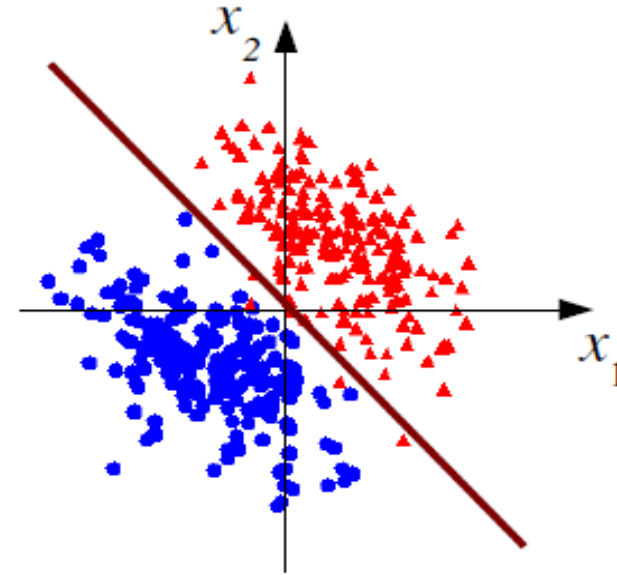


Statistical Data Analysis: Lecture 6

- 1 Probability, Bayes' theorem, random variables, pdfs
- 2 Functions of r.v.s, expectation values, error propagation
- 3 Catalogue of pdfs
- 4 The Monte Carlo method
- 5 Statistical tests: general concepts
-  6 **Test statistics, multivariate methods**
- 7 Goodness-of-fit tests
- 8 Parameter estimation, maximum likelihood
- 9 More maximum likelihood
- 10 Method of least squares
- 11 Interval estimation, setting limits
- 12 Nuisance parameters, systematic uncertainties
- 13 Examples of Bayesian approach
- 14 tba

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.

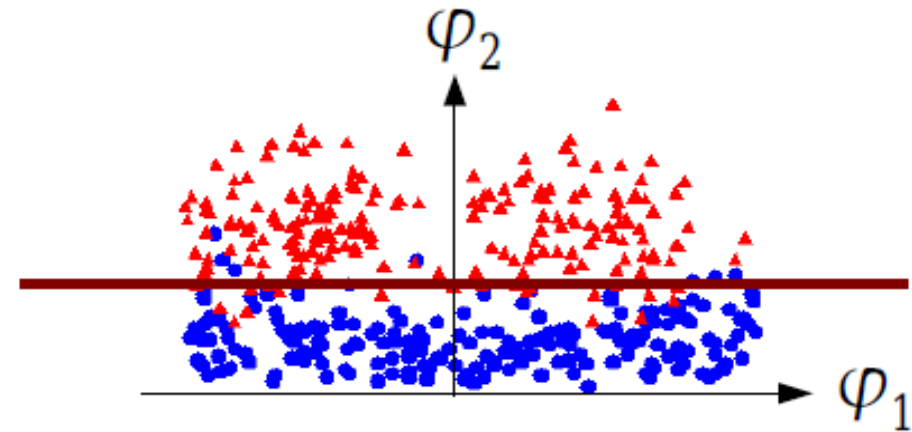
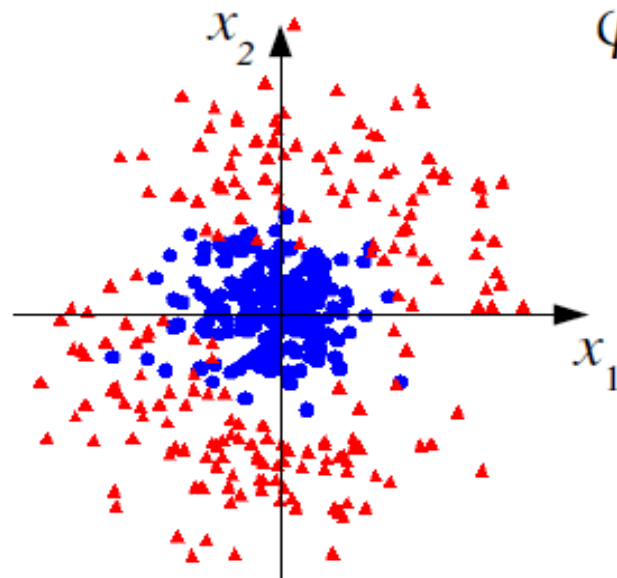
Nonlinear transformation of inputs

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)



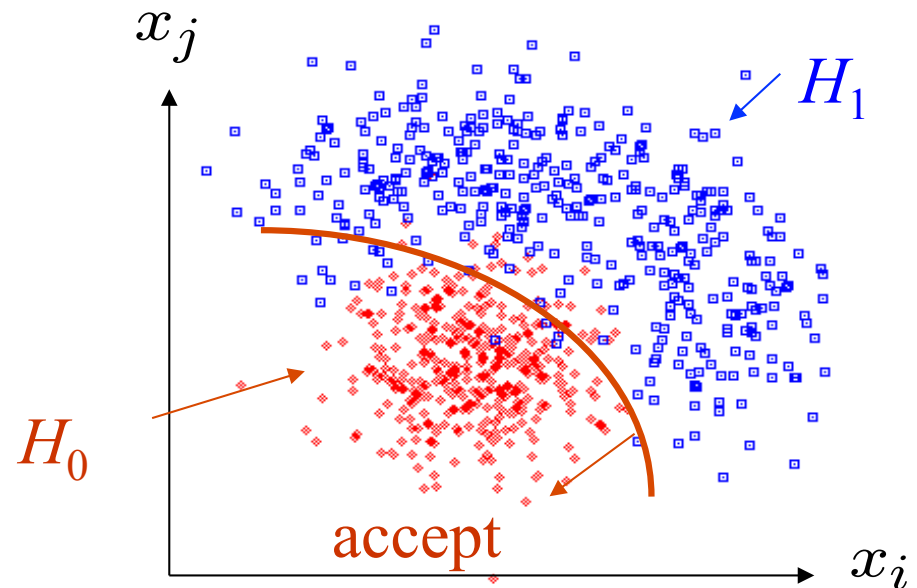
Nonlinear test statistics

The optimal decision boundary may not be a hyperplane,

→ nonlinear test statistic $t(\vec{x})$

Multivariate statistical methods
are a Big Industry:

Neural Networks,
Support Vector Machines,
Kernel density methods,
...



Particle Physics can benefit from progress in **Machine Learning**.

Introduction to neural networks

Used in neurobiology, pattern recognition, financial forecasting, ...
Here, neural nets are just a type of test statistic.

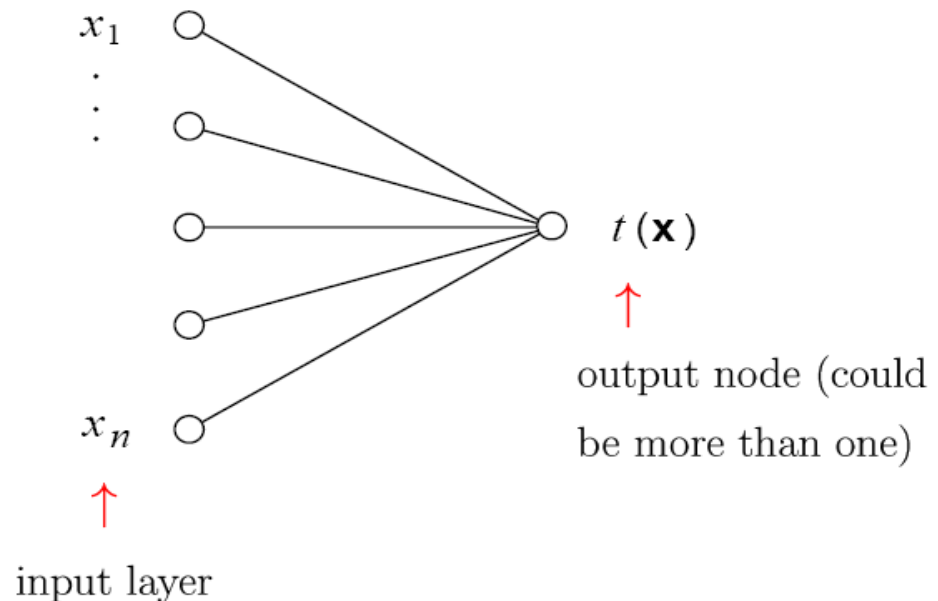
Suppose we take $t(\mathbf{x})$ to have the form

$$t(\vec{x}) = s \left(a_0 + \sum_{i=1}^n a_i x_i \right), \text{ where } s(u) \equiv (1 + e^{-u})^{-1} .$$

logistic
sigmoid

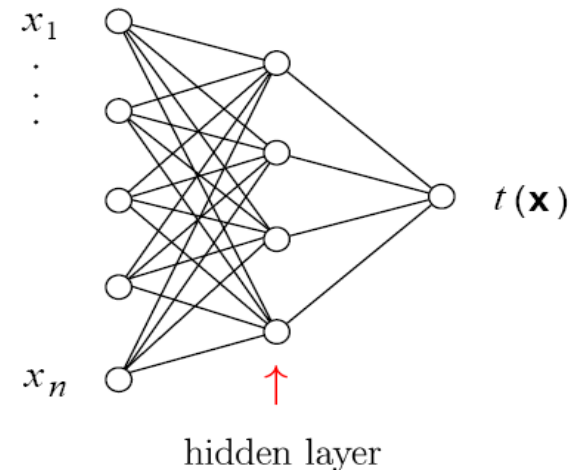
This is called the
single-layer perceptron.

$s(\cdot)$ is monotonic
→ equivalent to linear $t(\mathbf{x})$



The multi-layer perceptron

Generalize from one layer
to the **multilayer perceptron**:



The values of the nodes in the
intermediate (hidden) layer are

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

and the network output is given by

$$t(\vec{x}) = s \left(a_0 + \sum_{i=1}^n a_i h_i(\vec{x}) \right) .$$

$a_i, w_{ij} =$ weights (connection strengths)

Neural network discussion

Easy to generalize to arbitrary number of layers.

Feed-forward net: values of a node depend only on earlier layers, usually only on previous layer (“network architecture”).

More nodes \rightarrow neural net gets closer to optimal $t(\mathbf{x})$, but more parameters need to be determined.

Parameters usually determined by minimizing an error function,

$$\mathcal{E} = E_0[(t - t^{(0)})^2] + E_1[(t - t^{(1)})^2] ,$$

where $t^{(0)}$, $t^{(1)}$ are target values, e.g., 0 and 1 for logistic sigmoid.

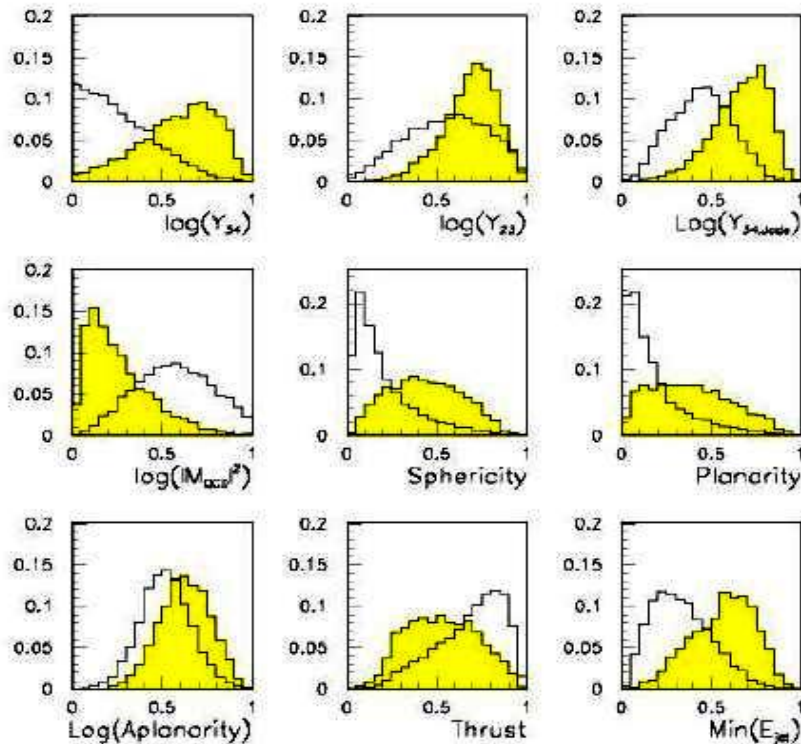
Expectation values replaced by averages of training data (e.g. MC).

In general training can be difficult; standard software available.

Neural network example from LEP II

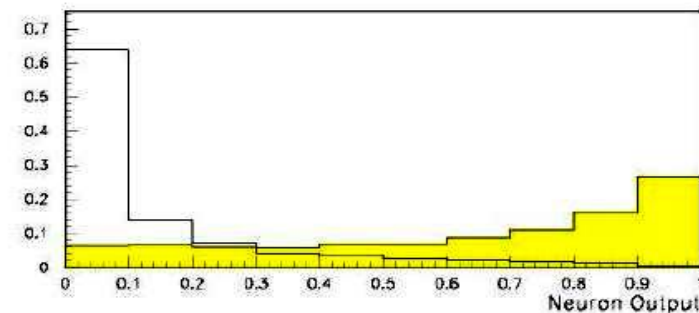
Signal: $e^+e^- \rightarrow W^+W^-$ (often 4 well separated hadron jets)

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 does better...



(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.

Usually 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.

Probability Density Estimation (PDE) techniques

Construct non-parametric estimators of the pdfs $f(\vec{x}|H_0)$, $f(\vec{x}|H_1)$:
and use these to construct the likelihood ratio

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

(n -dimensional histogram is a brute force example of this.)

More clever estimation techniques can get this to work for
(somewhat) higher dimension.

See e.g. K. Cranmer, *Kernel Estimation in High Energy Physics*, CPC **136** (2001) 198; hep-ex/0011057; T. Carli and B. Koblitz, *A multi-variate discrimination technique based on range-searching*, NIM A **501** (2003) 576; hep-ex/0211019

Kernel-based PDE (KDE, Parzen window)

Consider d dimensions, N training events, $\mathbf{x}_1, \dots, \mathbf{x}_N$,
estimate $f(\mathbf{x})$ with

$$\hat{f}(\vec{x}) = \frac{1}{Nh^d} \sum_{i=1}^N K\left(\frac{\vec{x} - \vec{x}_i}{h}\right)$$

kernel

bandwidth
(smoothing parameter)

Use e.g. Gaussian kernel:

$$K(\vec{x}) = \frac{1}{(2\pi)^{d/2}} e^{-|\vec{x}|^2/2}$$

Need to sum N terms to evaluate function (slow);
faster algorithms only count events in vicinity of \mathbf{x}
(k -nearest neighbor, range search).

Correlation vs. independence

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

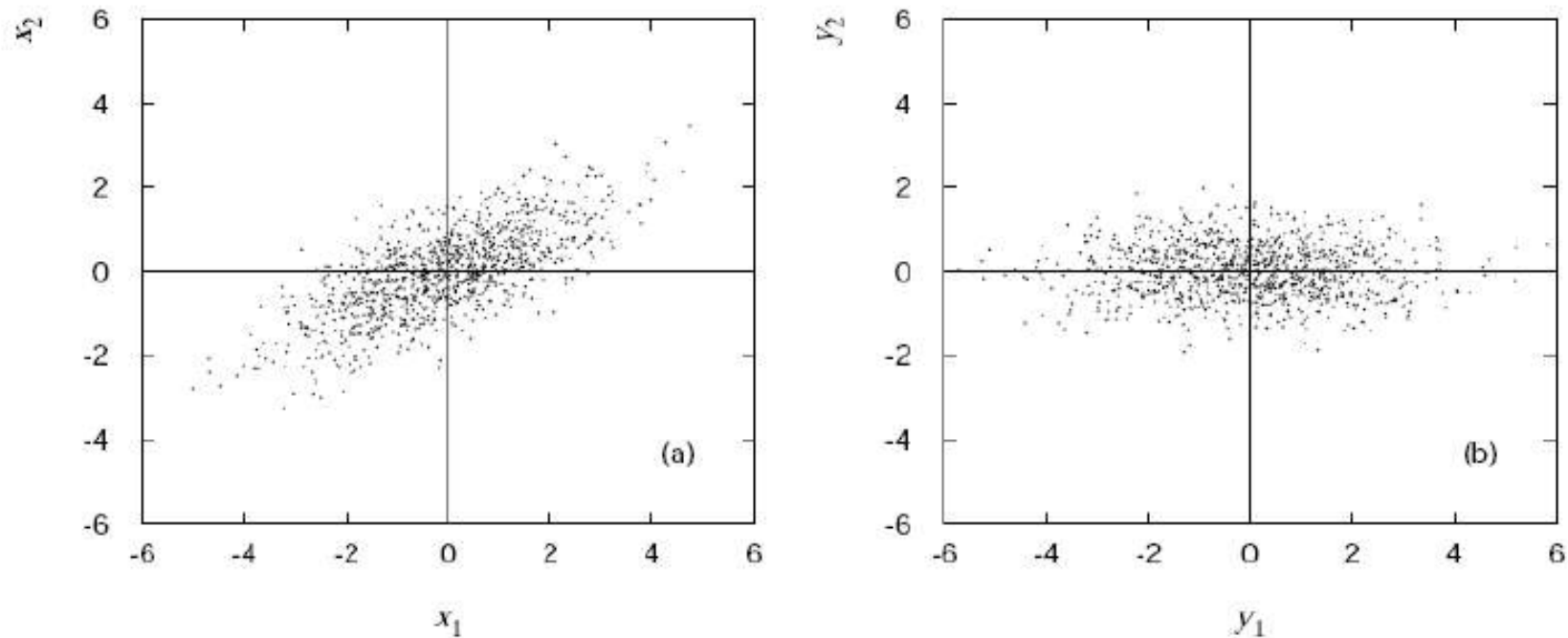
$$p(\vec{x}) = \prod_{i=1}^n p_i(x_i) \quad \leftarrow \text{holds only if the components of } \mathbf{x} \text{ are independent}$$

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

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

Decorrelation of input variables

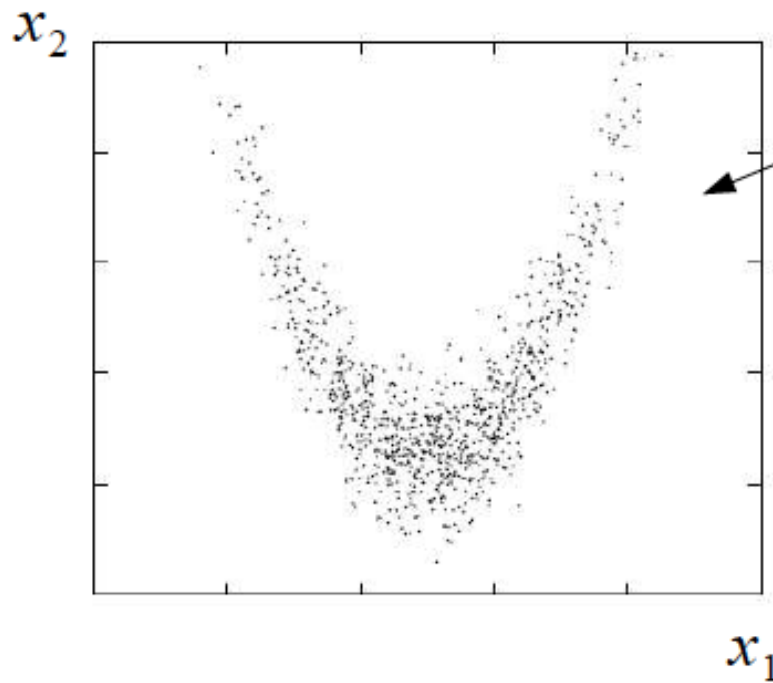
But we can define a set of uncorrelated input variables by a linear transformation, i.e., find the matrix A such that for $\vec{y} = A \vec{x}$ the covariances $\text{cov}[y_i, y_j] = 0$:



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 $p(\mathbf{x})$ 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”).

Decision trees

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

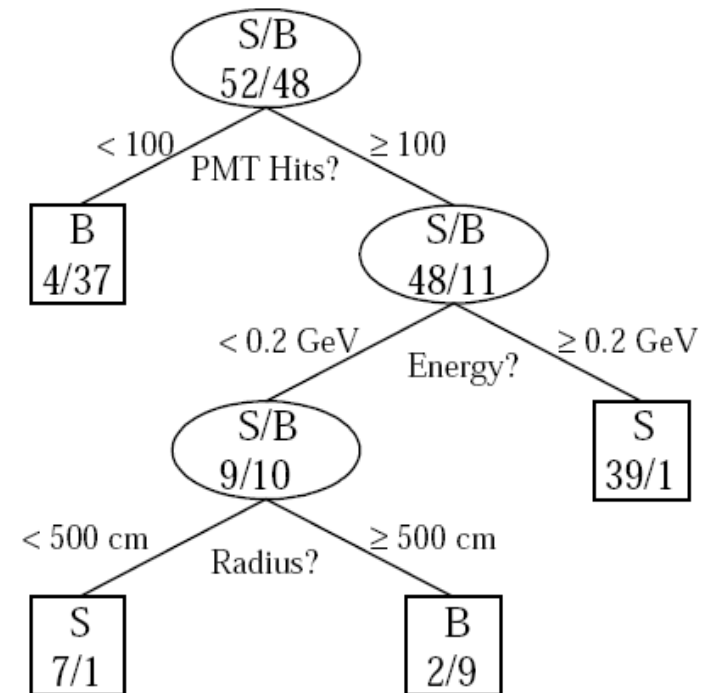
$$P = \frac{\sum_{\text{signal}} w_i}{\sum_{\text{signal}} w_i + \sum_{\text{background}} w_i}$$

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 trees (2)

The terminal nodes (**leaves**) are classified as 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 the discriminant function

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

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.

AdaBoost

A successful boosting algorithm is AdaBoost (Freund & Schapire, 1997).

First initialize the training sample T_1 using the original

$\mathbf{x}_1, \dots, \mathbf{x}_N$ event data vectors

y_1, \dots, y_N true class labels (+1 or -1)

$w_1^{(1)}, \dots, w_N^{(1)}$ event weights

with the weights equal and normalized such that $\sum_{i=1}^N w_i^{(1)} = 1$.

Train the classifier $f_1(\mathbf{x})$ (e.g. a decision tree) using the weights $w^{(1)}$ so as to minimize the classification error rate,

$$\varepsilon_1 = \sum_{i=1}^N w_i^{(1)} I(y_i f_1(\mathbf{x}_i) \leq 0),$$

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

Updating the event weights (AdaBoost)

Assign a score to the k th classifier based on its error rate:

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

Define the training sample for step $k+1$ from that of k by updating the event weights according to

$$w_i^{(k+1)} = w_i^{(k)} \frac{e^{-\alpha_k f_k(\mathbf{x}_i) y_i / 2}}{Z_k}$$

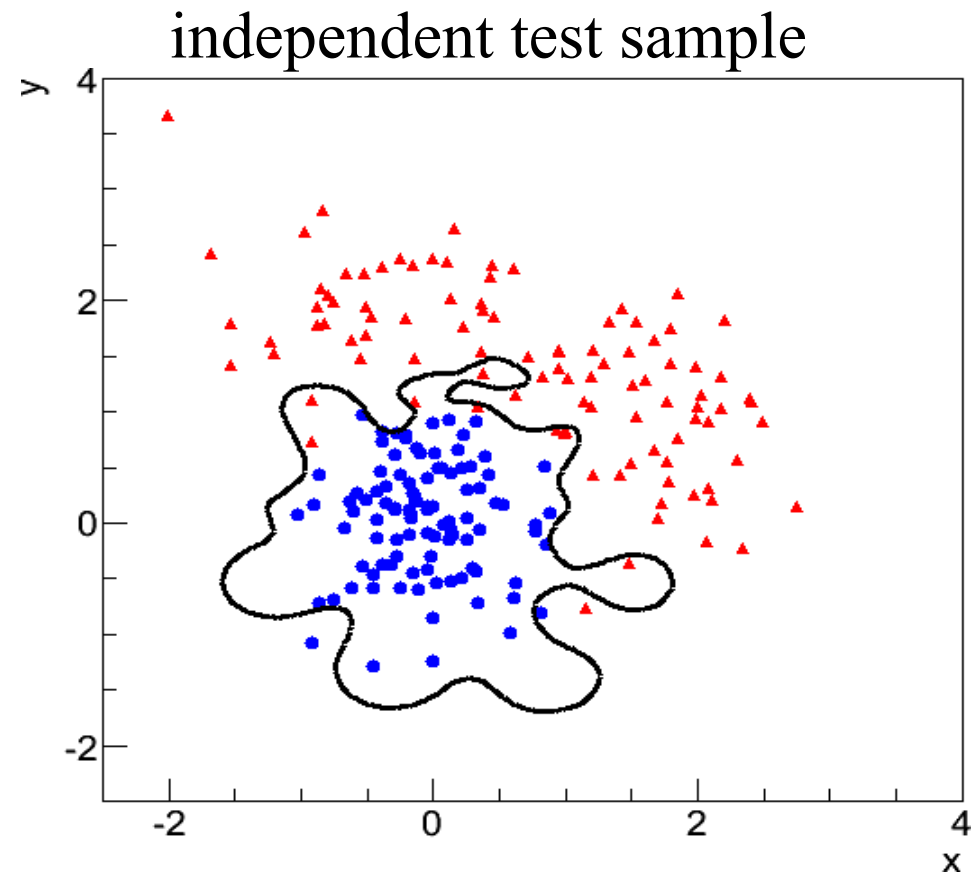
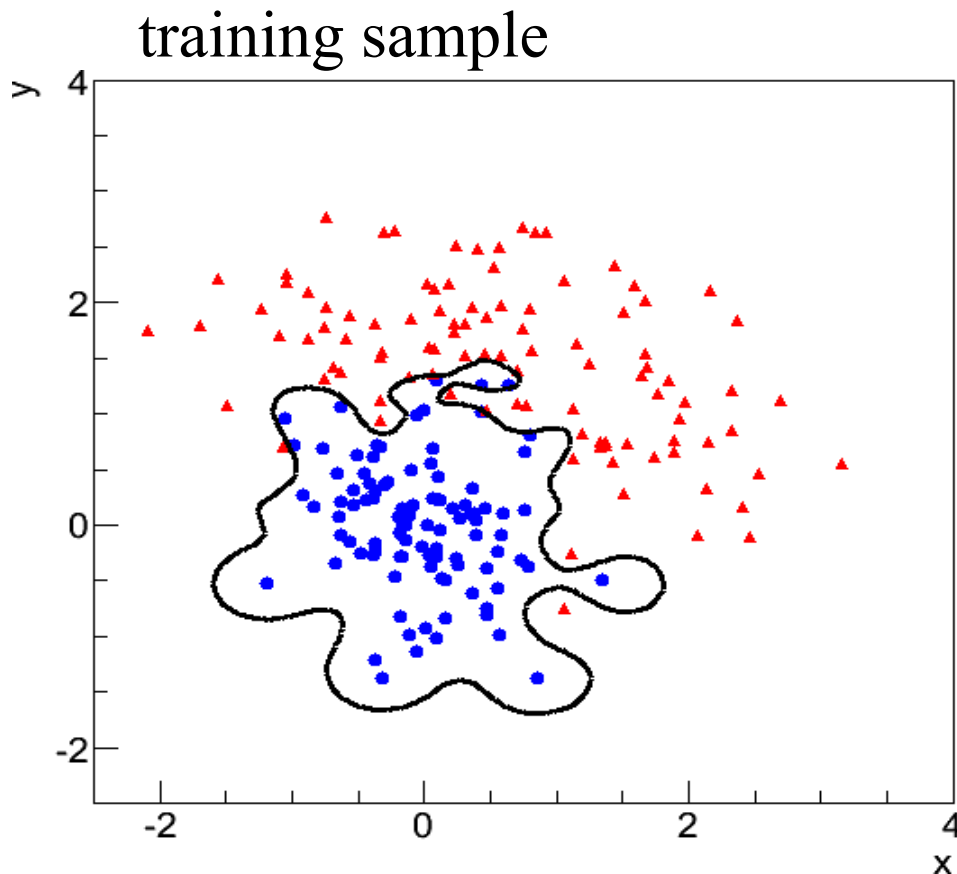
$i = \text{event index}$ $k = \text{training sample index}$ Normalize so that $\sum_i w_i^{(k+1)} = 1$

Iterate K times, final classifier is $y(\mathbf{x}) = \sum_{k=1}^K \alpha_k f_k(\mathbf{x}, T_k)$

Overtraining

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

Monitor by applying classifier to independent test sample.

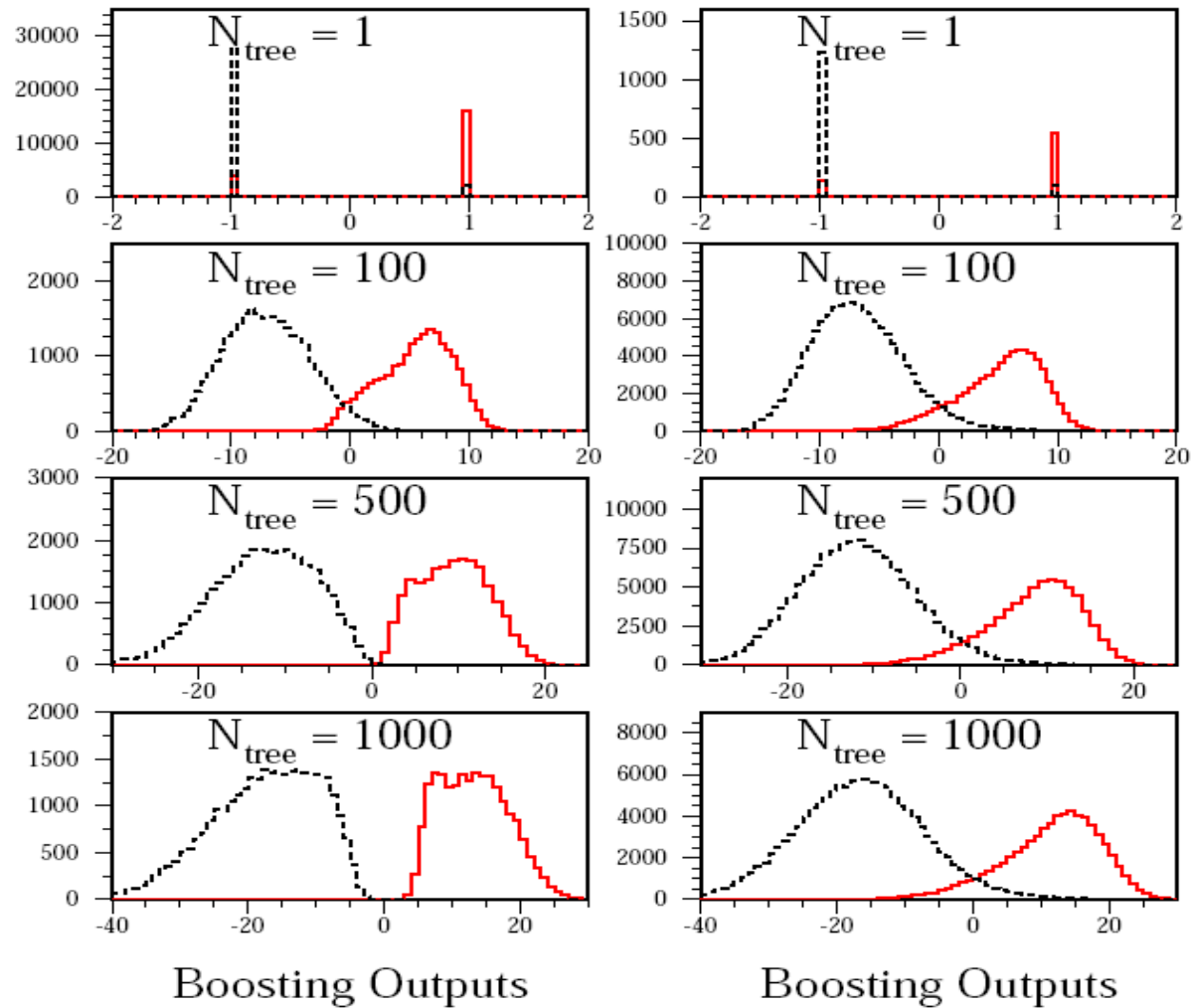


Monitoring overtraining

From MiniBooNE
example:

Performance stable
after a few hundred
trees.

Training MC Samples .VS. Testing MC Samples $\times 10^2$



Boosted decision tree summary

Advantage of boosted decision tree is it can handle a large number of inputs. Those that provide little/no separation are rarely used as tree splitters are effectively ignored.

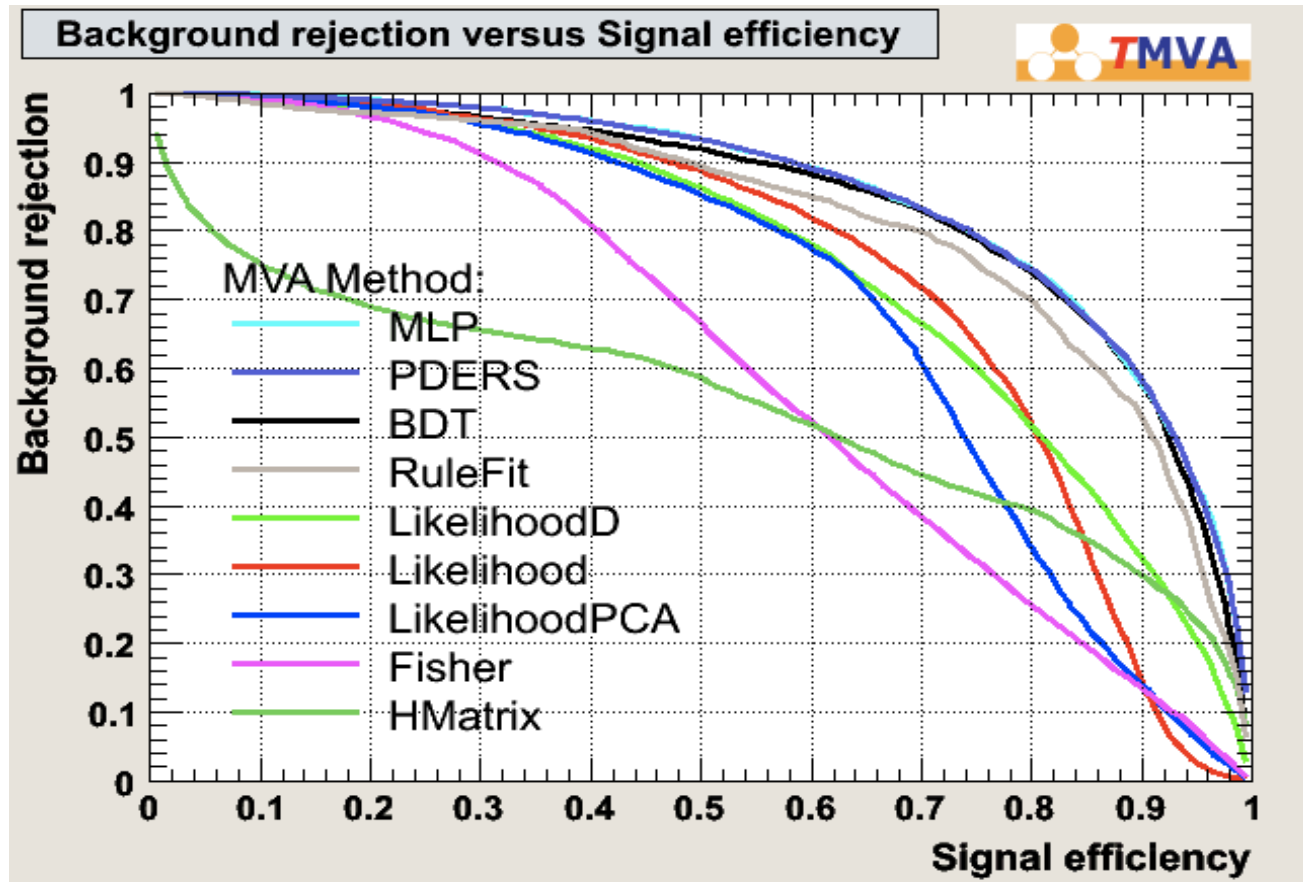
Easy to deal with inputs of mixed types (real, integer, categorical...).

If a tree has only a few leaves it is easy to visualize (but rarely use only a single tree).

There are a number of boosting algorithms, which differ primarily in the rule for updating the weights (ϵ -Boost, LogitBoost,...)

Other ways of combining weaker classifiers: Bagging (Bootstrap-Aggregating), generates the ensemble of classifiers by random sampling with replacement from the full training sample.

Comparing multivariate methods (TMVA)



Choose the best one!

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

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

Wrapping up lecture 6

We looked at statistical tests and related issues:

discriminate between event types (hypotheses),
determine selection efficiency, sample purity, etc.

Some modern (and less modern) methods were mentioned:

Fisher discriminants, neural networks,
PDE, KDE, decision trees, ...

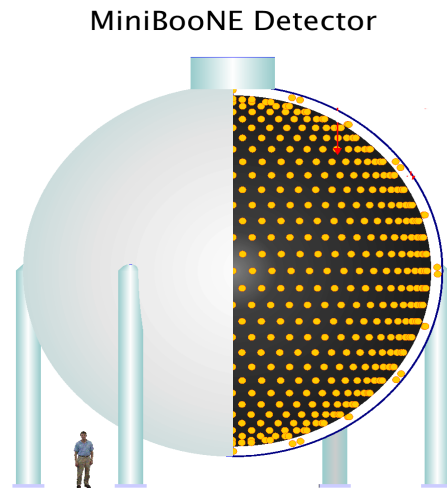
Next we will talk about **significance (goodness-of-fit) tests**:

p -value expresses level of agreement between data
and hypothesis

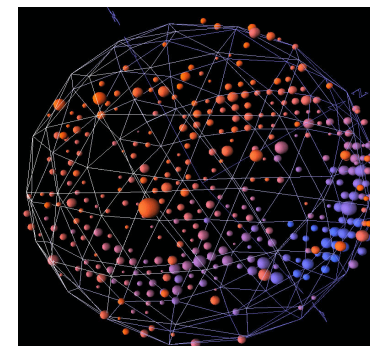
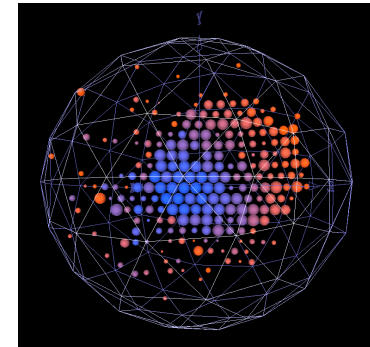
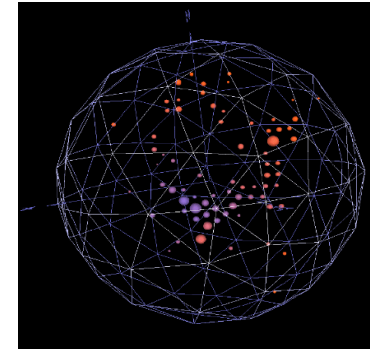
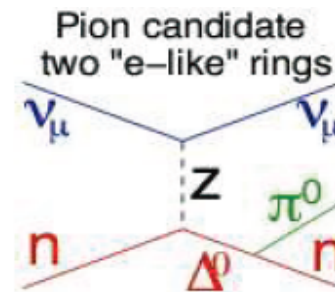
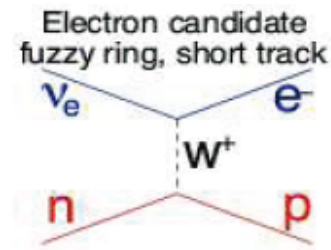
Extra slides

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.

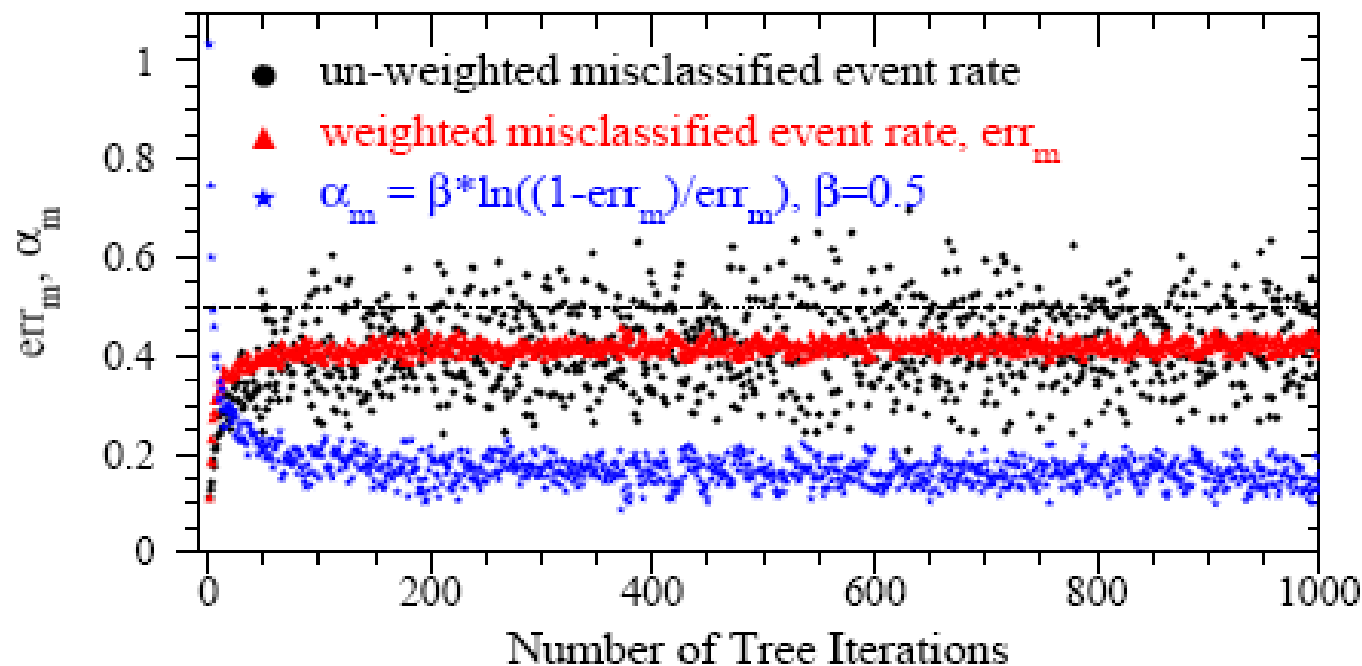


H.J. Yang, MiniBooNE PID, DNP06

BDT example from MiniBooNE

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

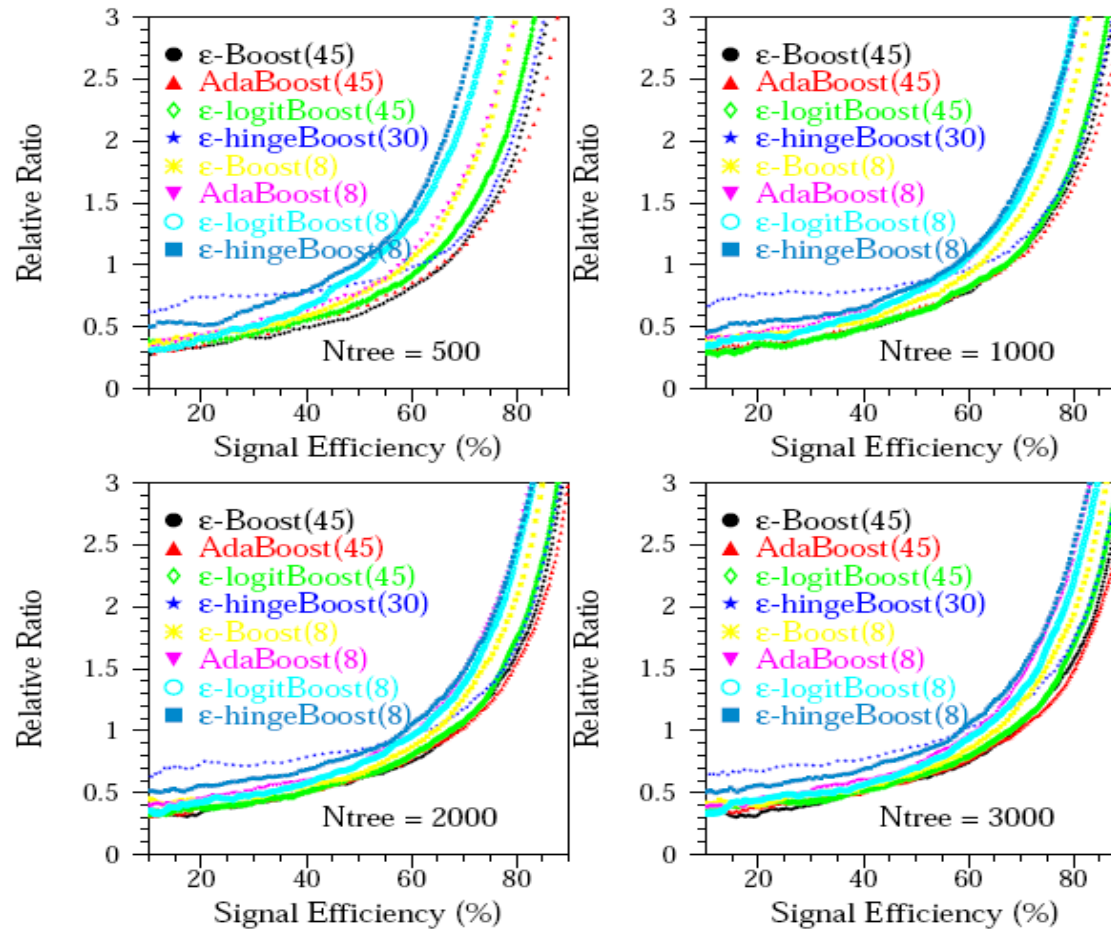
Each individual tree is relatively weak, with a misclassification error rate $\sim 0.4 - 0.45$



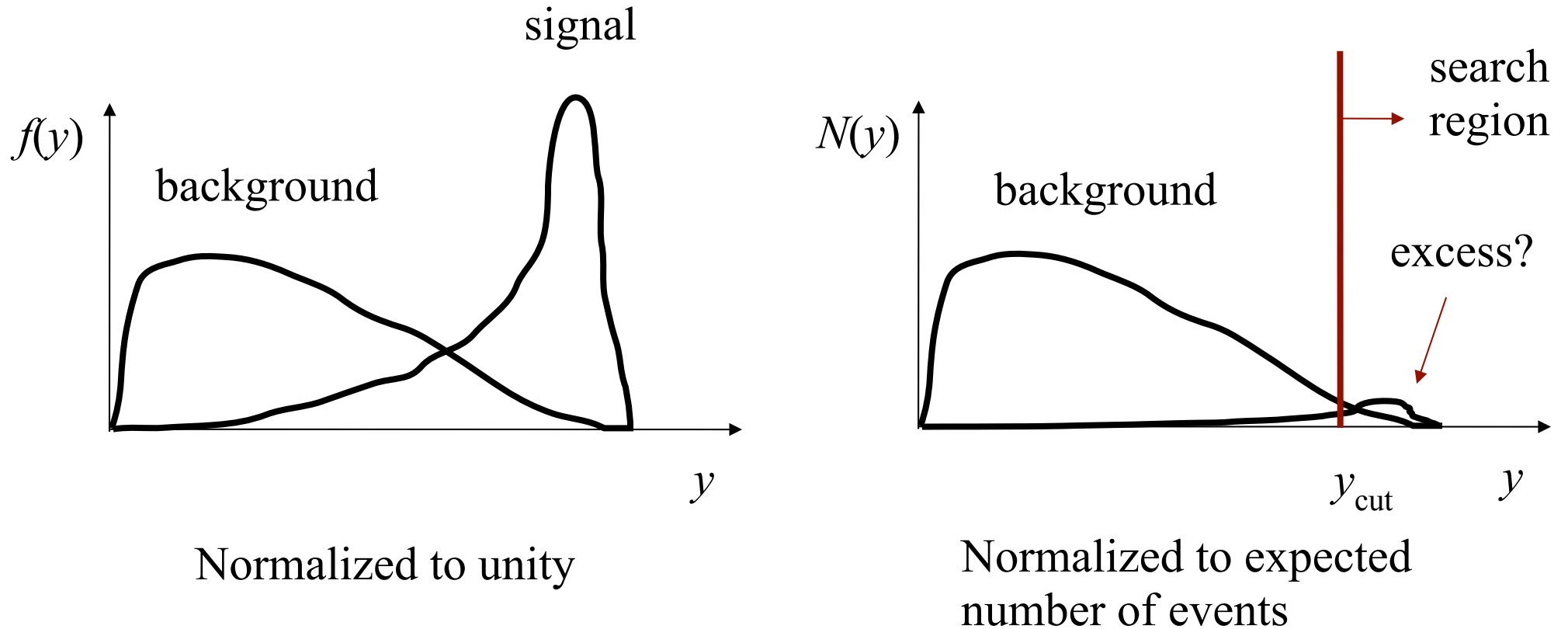
B. Roe et al., NIM 543 (2005) 577

Comparison of boosting algorithms

A number of boosting algorithms on the market; differ in the update rule for the weights.



Using classifier output for discovery

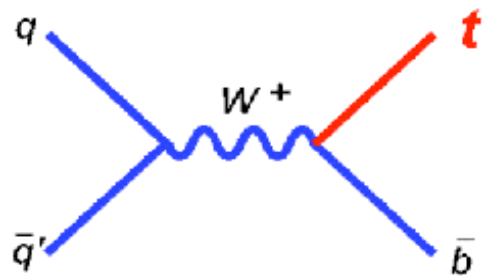


Discovery = number of events found in search region incompatible with background-only hypothesis.

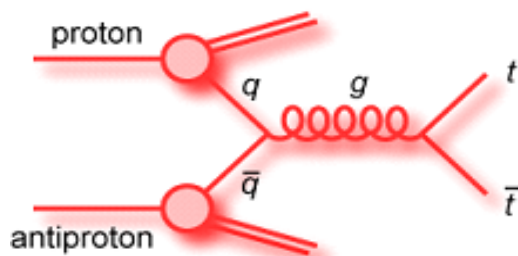
p -value of background-only hypothesis can depend crucially distribution $f(y|b)$ in the "search region".

Single top quark production (CDF/D0)

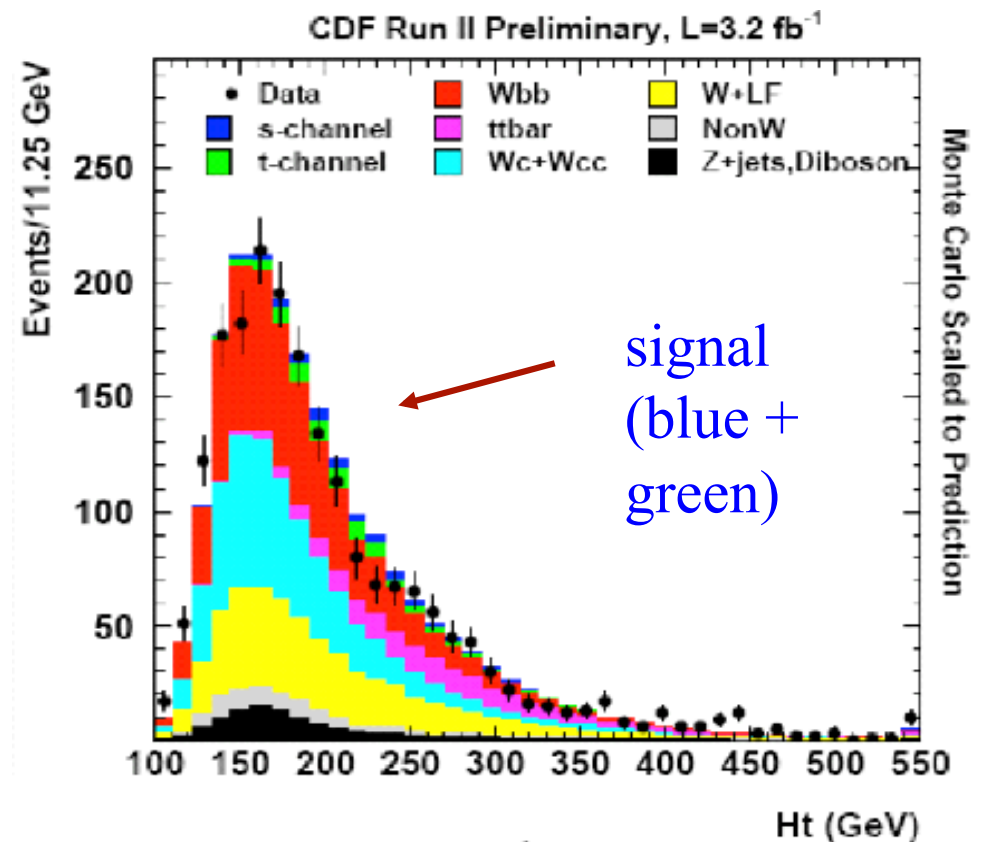
Top quark discovered in pairs, but SM predicts single top production.



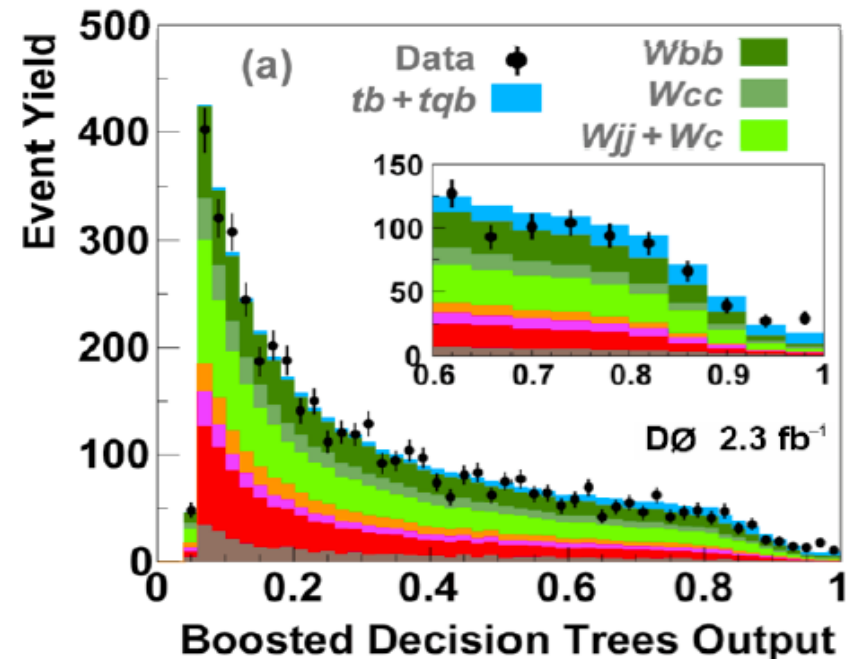
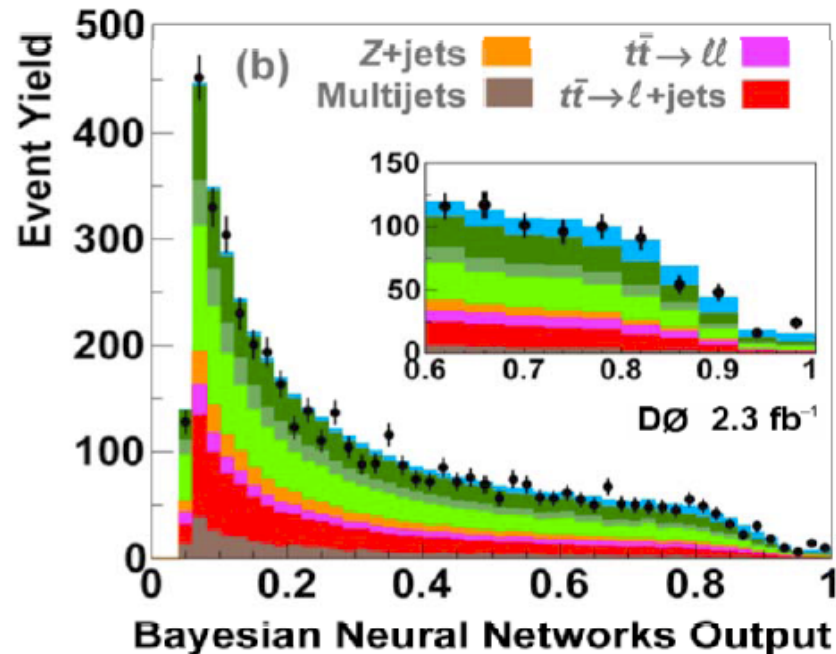
Pair-produced tops are now a background process.



Use many inputs based on jet properties, particle i.d., ...



Different classifiers for single top



Also Naive Bayes and various approximations to likelihood ratio,....

Final combined result is statistically significant ($>5\sigma$ level) but not easy to understand classifier outputs.