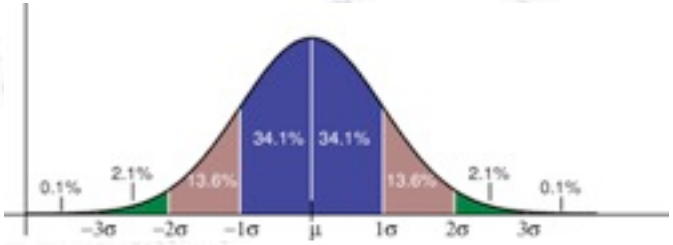
Applied Statistics

Multivariate Analysis - part II

Troels C. Petersen (NBI)



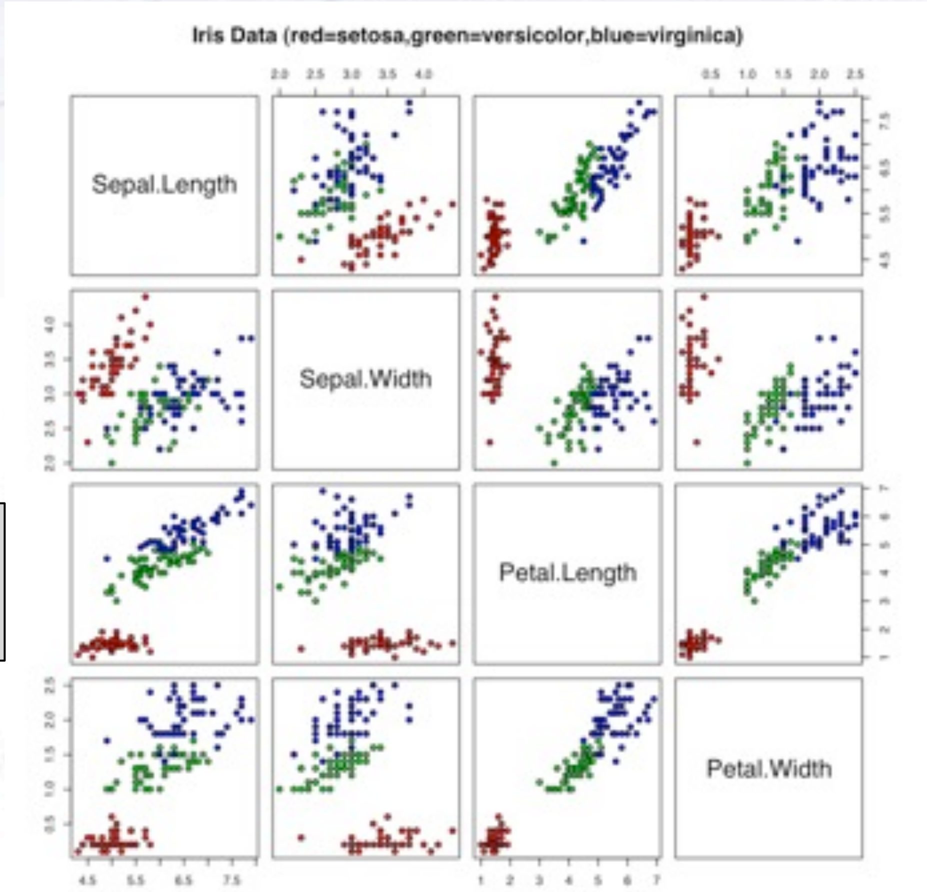
"Statistics is merely a quantization of common sense"

Fisher Discriminant

You want to separate two types/classes (A and B) of events using several measurements.

<u>Q</u>: How to combine the variables?
<u>A</u>: Use the Fisher Discriminant:

$$\boxed{\mathcal{F} = w_0 + \vec{w} \cdot \vec{x}}$$



<u>Q</u>: How to choose the values of w?
<u>A</u>: Inverting the covariance matrices:

$$\boxed{\vec{w} = \left(\mathbf{\Sigma}_A + \mathbf{\Sigma}_B\right)^{-1} \ \left(\vec{\mu}_A - \vec{\mu}_B\right)}$$

This can be calculated analytically, and incorporates the linear correlations into the separation capability.

Fisher Discriminant

The details of the formula are outlined below:

For each input variable (x), you calculate the mean (μ), and form a vector of these.

$$\vec{w} = (\mathbf{\Sigma}_A + \mathbf{\Sigma}_B)^{-1} \ (\vec{\mu}_A - \vec{\mu}_B)$$

Using the input variables (x), you calculate the covariance matrix (Σ) for each species (A/B), add these and invert.

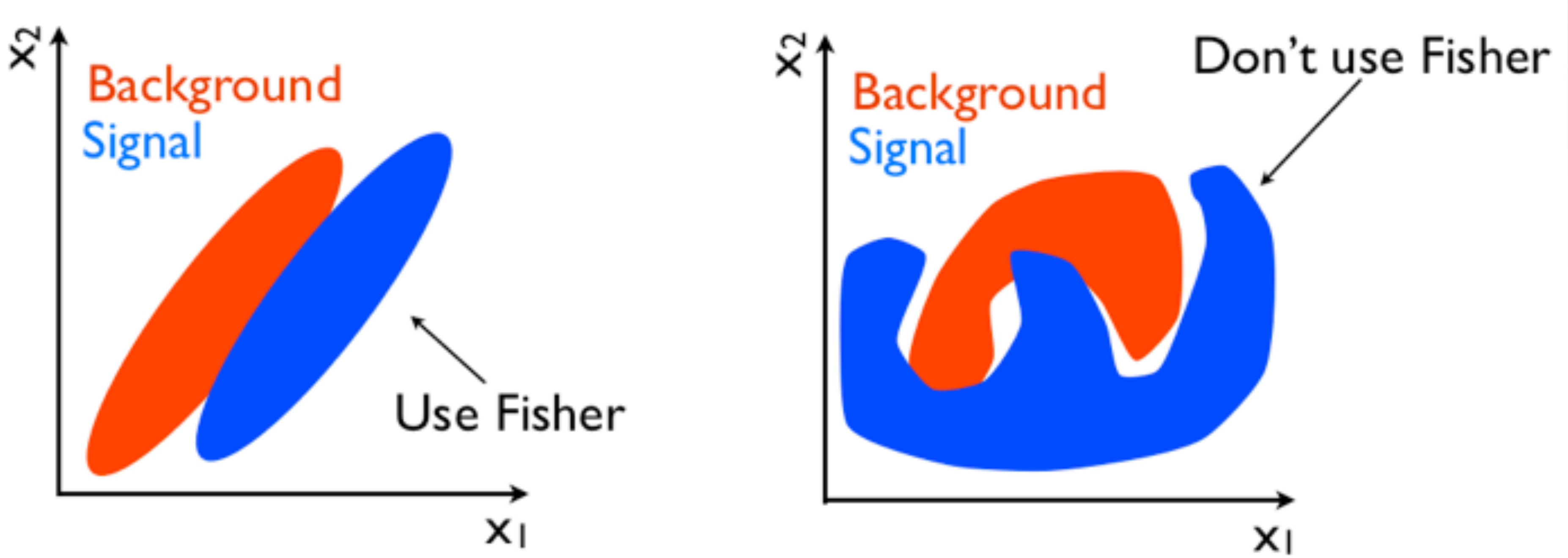
Given weights (w), you take your input variables (x) and combine them linearly as follows:

$$\mathcal{F} = w_0 + \vec{w} \cdot \vec{x}$$

F is what you base your decision on.

Non-linear MVAs

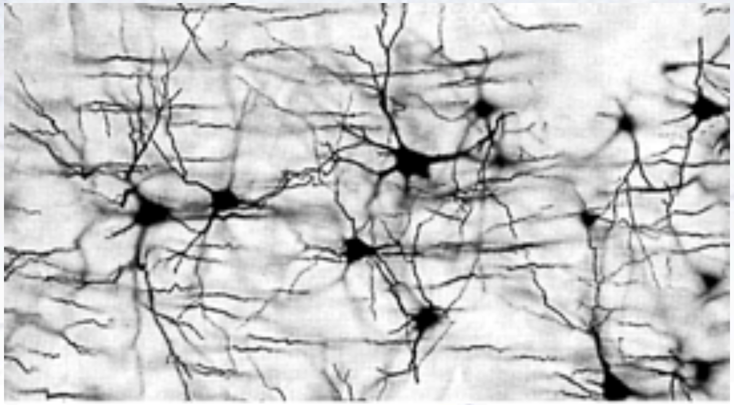
While the Fisher Discriminant uses all separations and **linear correlations**, it does not perform optimally, when there are **non-linear correlations** present:

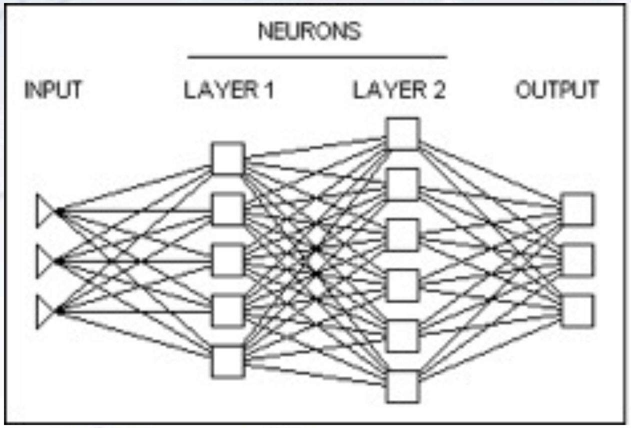


If the PDFs of signal and background are known, then one can use a likelihood.

But this is **very rarely** the case, and therefore more "tough" methods are needed...

Neural Networks (NN)





In machine learning and related fields, artificial neural networks (ANNs) are computational models inspired by an animal's central nervous systems (in particular the brain) which is capable of ***machine learning*** *as well as* ***pattern recognition****.*

Neural networks *have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including* ***computer vision*** *and* ***speech recognition****.*

[Wikipedia, Introduction to Artificial Neural Network]

Neural Networks

Neural Networks combine the input variables using a “activation” function s(x) to assign, if the variable indicates signal or background.

The simplest is a single layer perceptron:

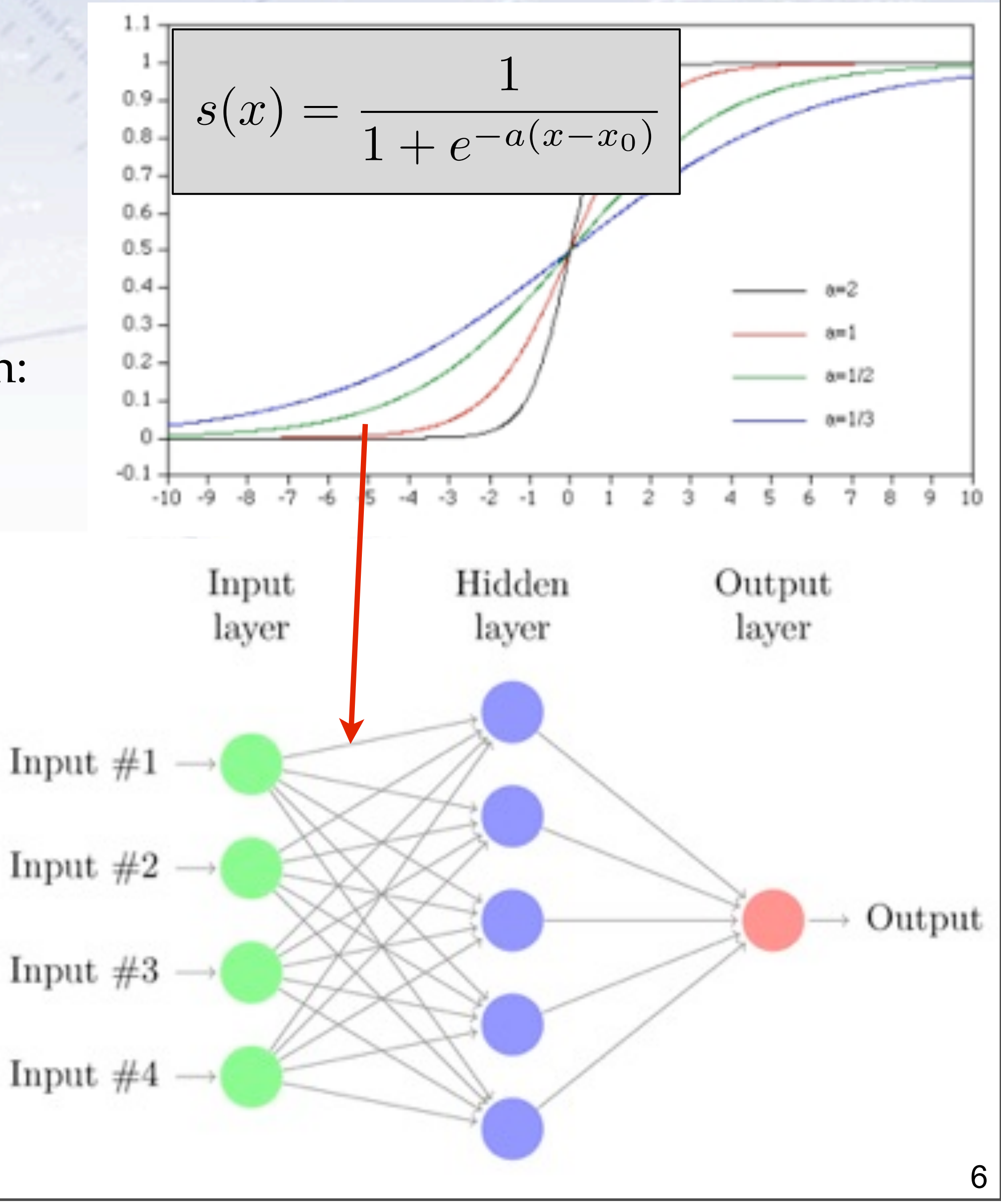
$$t(x) = s\left(a_0 + \sum a_i x_i\right)$$

This can be generalized to a multilayer perceptron:

$$t(x) = s\left(a_i + \sum a_i h_i(x)\right)$$

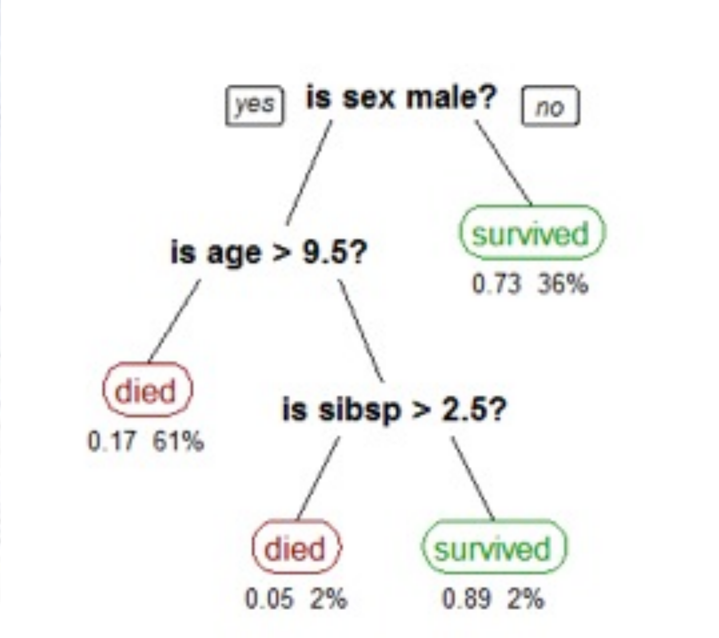
$$h_i(x) = s\left(w_{i0} + \sum w_{ij} x_j\right)$$

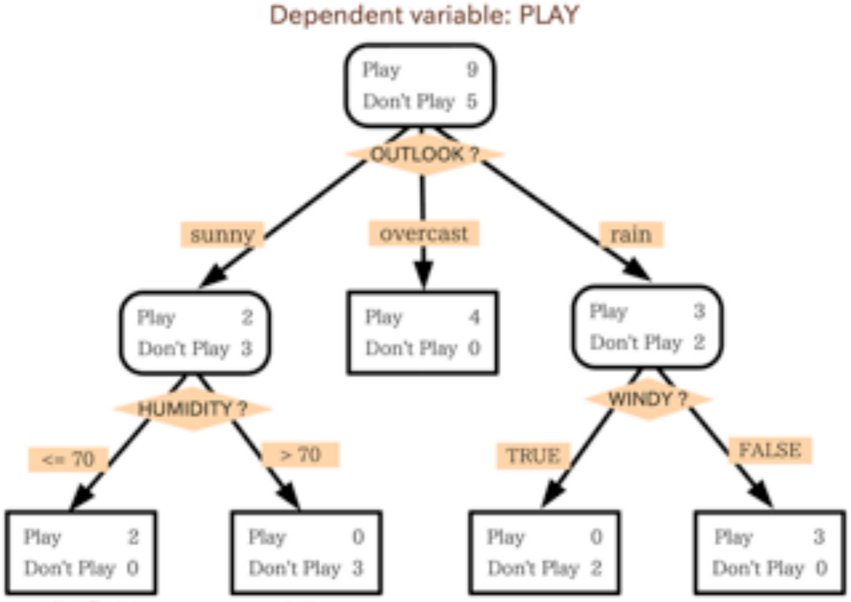
Activation function can be any sigmoid function.



$$s(x) = \frac{1}{1 + e^{-a(x-x_0)}}$$

Boosted Decision Trees (BDT)

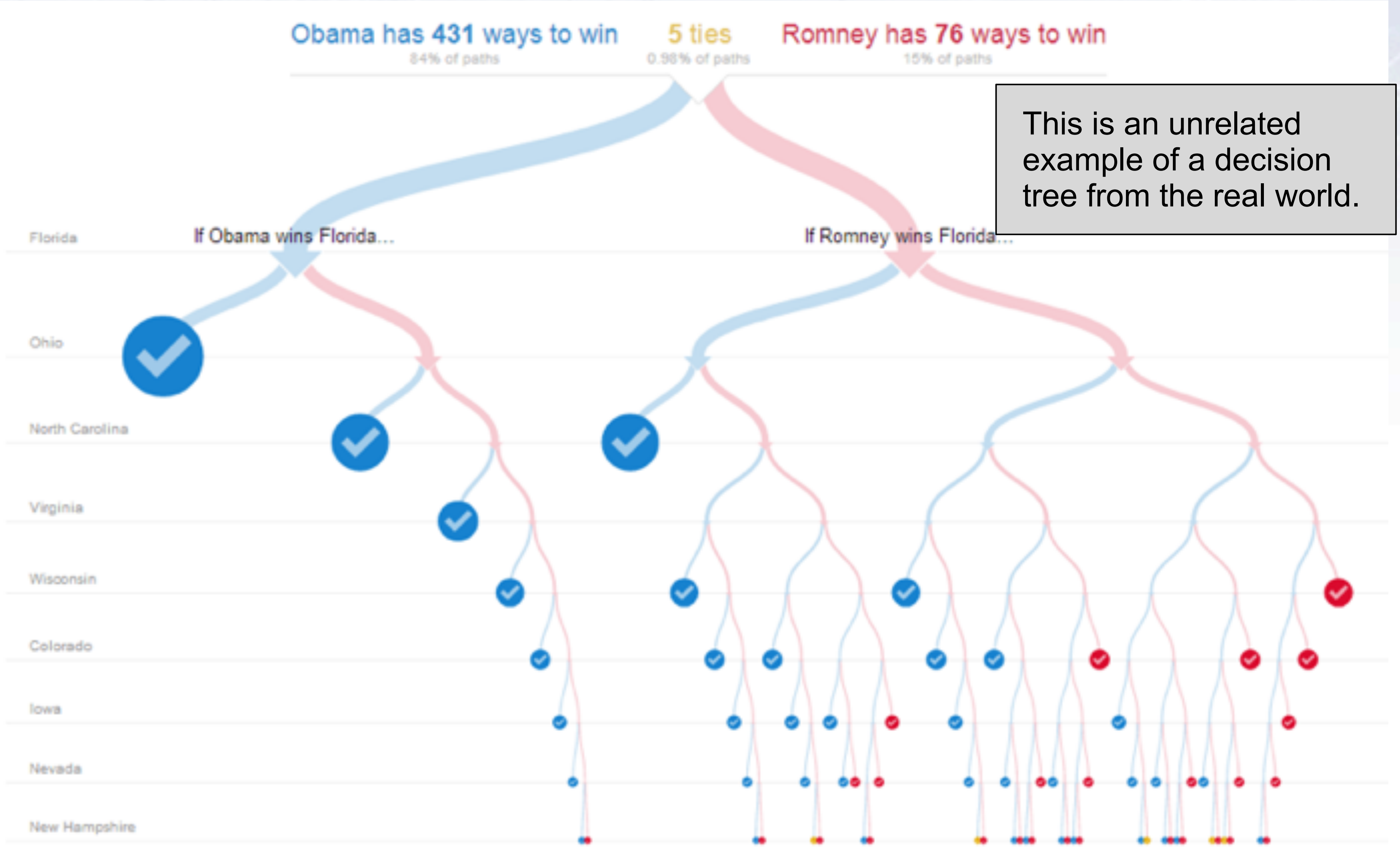




***Decision tree learning** uses a **decision tree** as a **predictive model** which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in **statistics**, **data mining** and **machine learning**.*

[Wikipedia, Introduction to Decision Tree Learning]

Boosted Decision Trees (BDT)



This is an unrelated example of a decision tree from the real world.

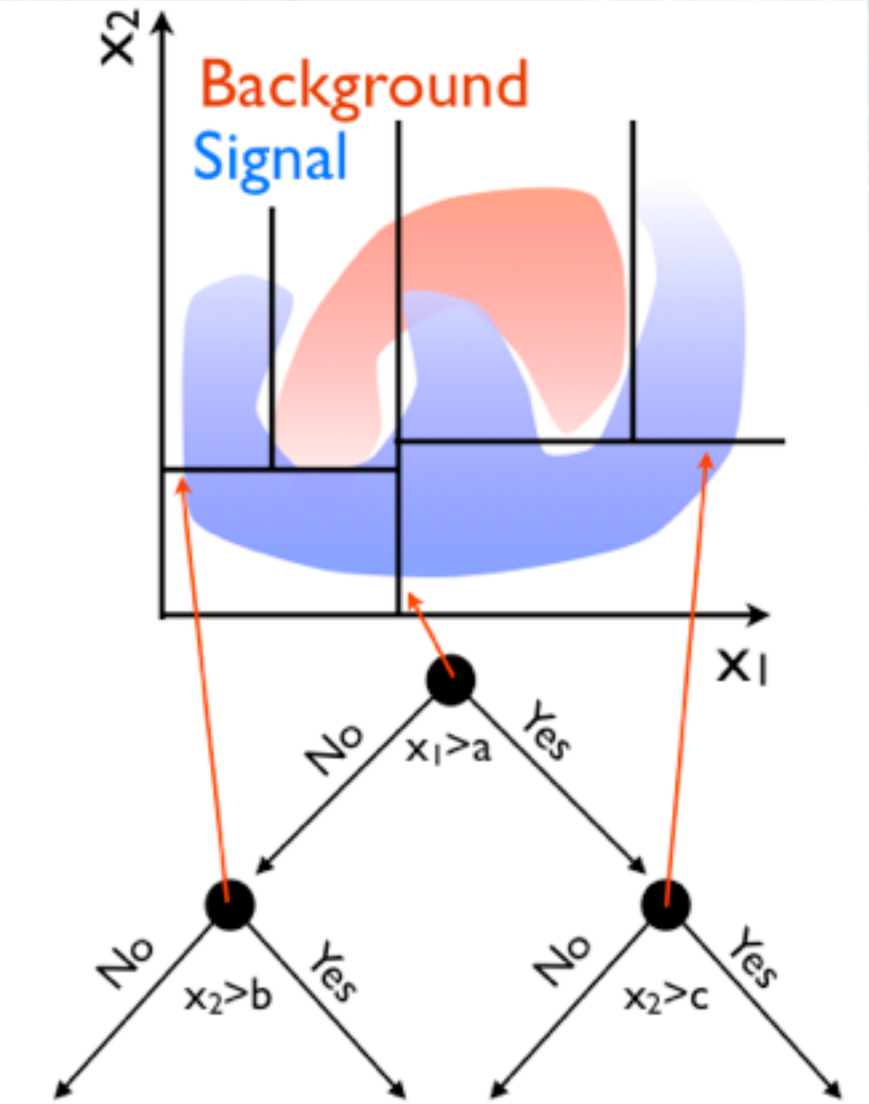
Boosted Decision Trees

A decision tree divides the parameter space, starting with the maximal separation. In the end each part has a probability of being signal or background.

- Works in 95+% of all problems!
- Fully uses non-linear correlations.

But BDTs require a lot of data for training, and is sensitive to overtraining (see next slide).

Overtraining can be reduced by limiting the number of nodes and number of trees.



Boosting...

There is no reason, why you can not have more trees. Each tree is a simple classifier, but many can be combined!

To avoid N identical trees, one assigns a higher weight to events that are hard to classify, i.e. boosting:

Boost weight

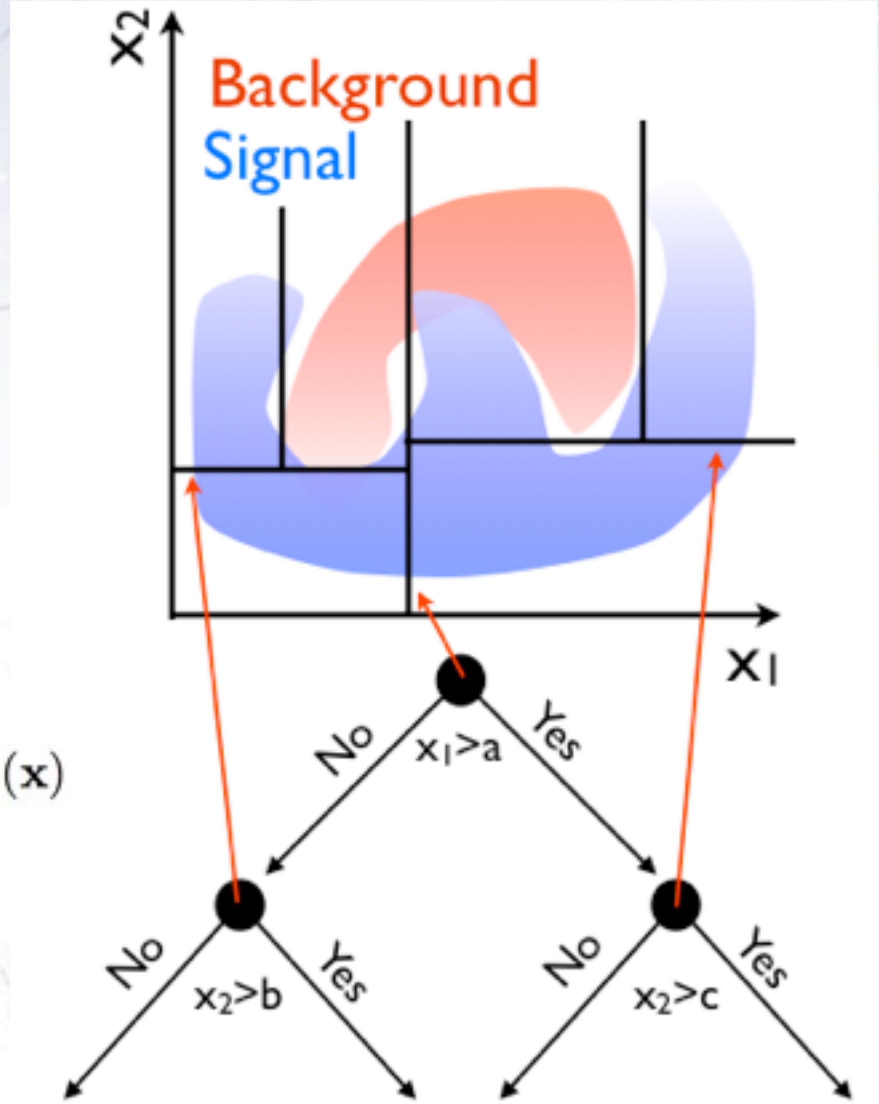
$$\alpha = \frac{1 - \text{err}}{\text{err}}$$

First classifier

$$y_{\text{Boost}}(\mathbf{x}) = \frac{1}{N_{\text{collection}}} \cdot \sum_{i}^{N_{\text{collection}}} \ln(\alpha_i) \cdot h_i(\mathbf{x})$$

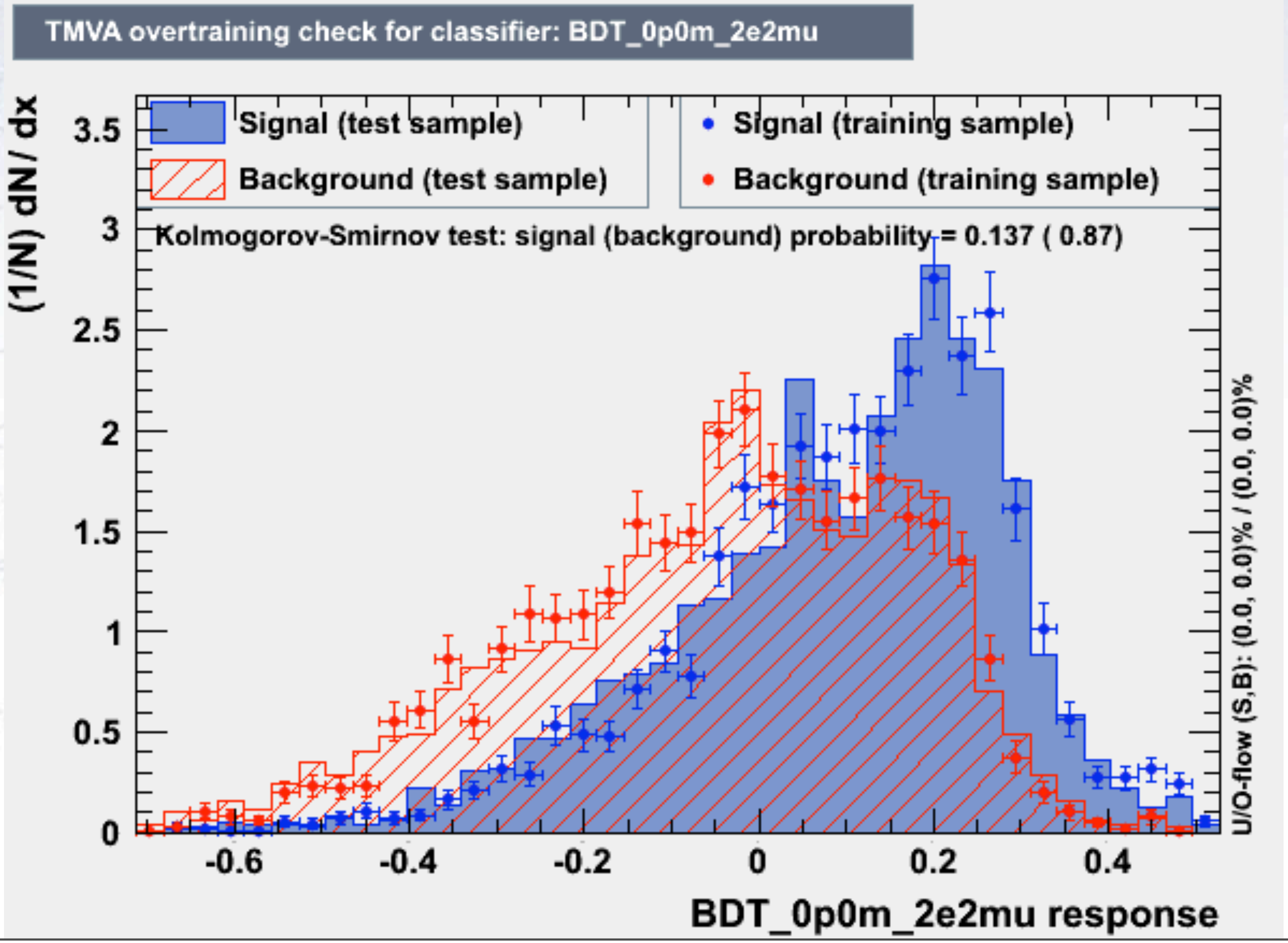
Parameters in event N

Individual tree



Test for overtraining

In order to test for overtraining, half the sample is used for training, the other for testing:



TMVA - MVA in ROOT

The main **parameters to consider** in setting up the training in TMVA are:

Generally:

- Definition of signal and background (selection and/or separate files)
- Variables to use ("vardict")
- Which methods to use ("factory.BookMethod", i.e. Fisher, NN, BDT, etc.)
- Settings/options (see below)

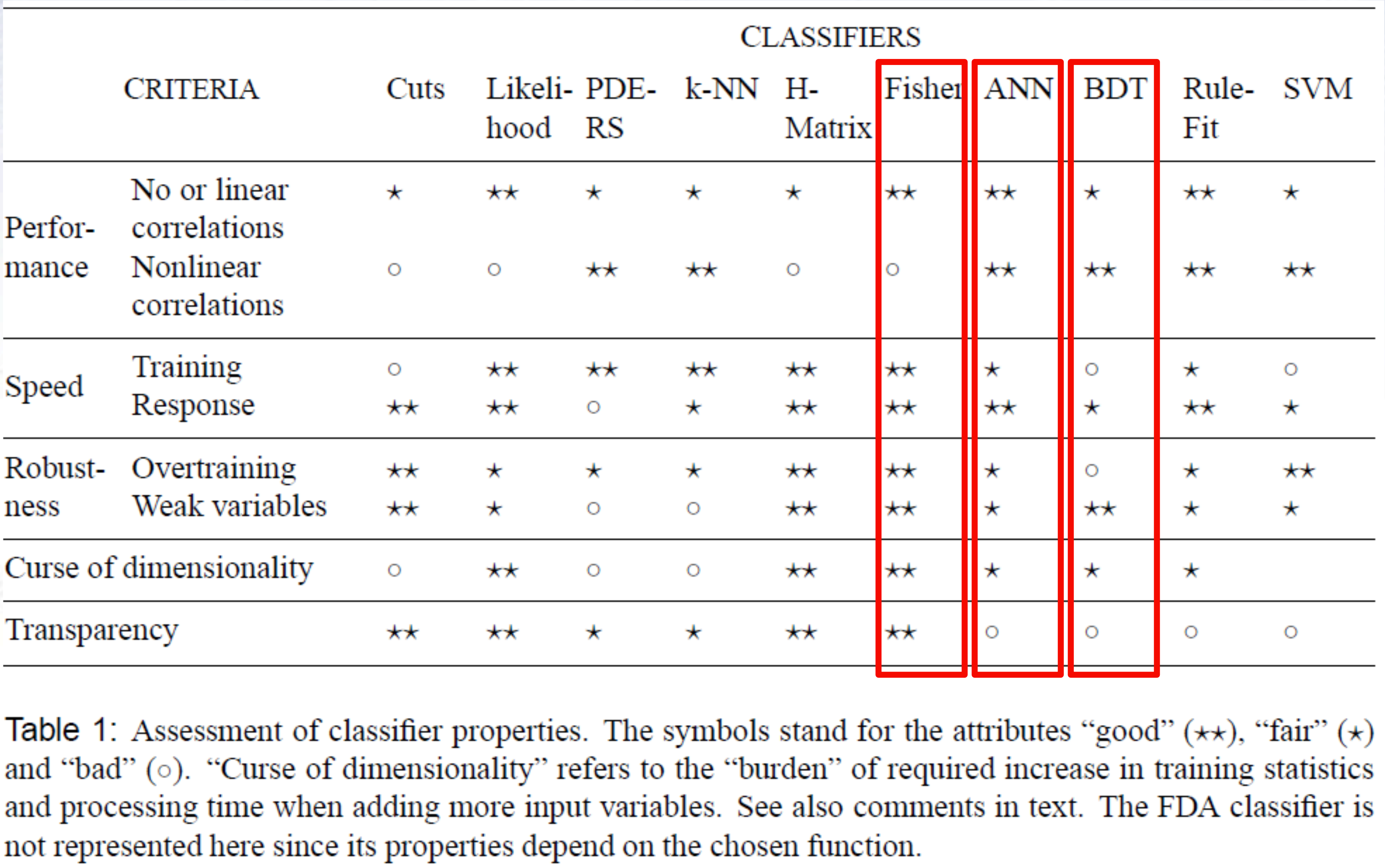
Boosted Decision Tree:

- Transformation of variables? (i.e. Principle Component Analysis (PCA))
- Number of trees ("NTrees").
- Maximum depth of trees ("MaxDepth").

Neural Network:

- Transformation of variables? (i.e. Principle Component Analysis (PCA)).
- Number of cycles ("NCycles", i.e. number of training rounds).
- Number of hidden layers ("N", i.e. "N+1" or "N+1, N").

Method’s (dis-)advantages

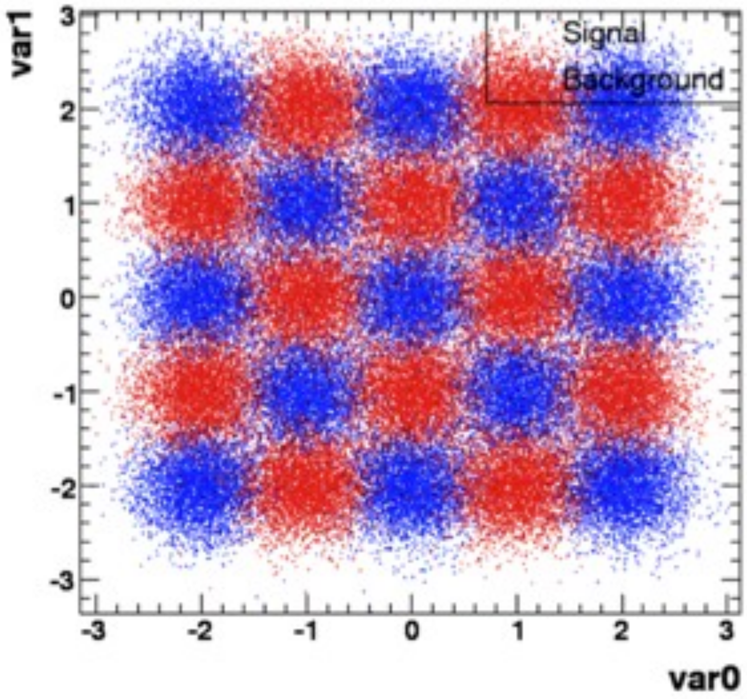


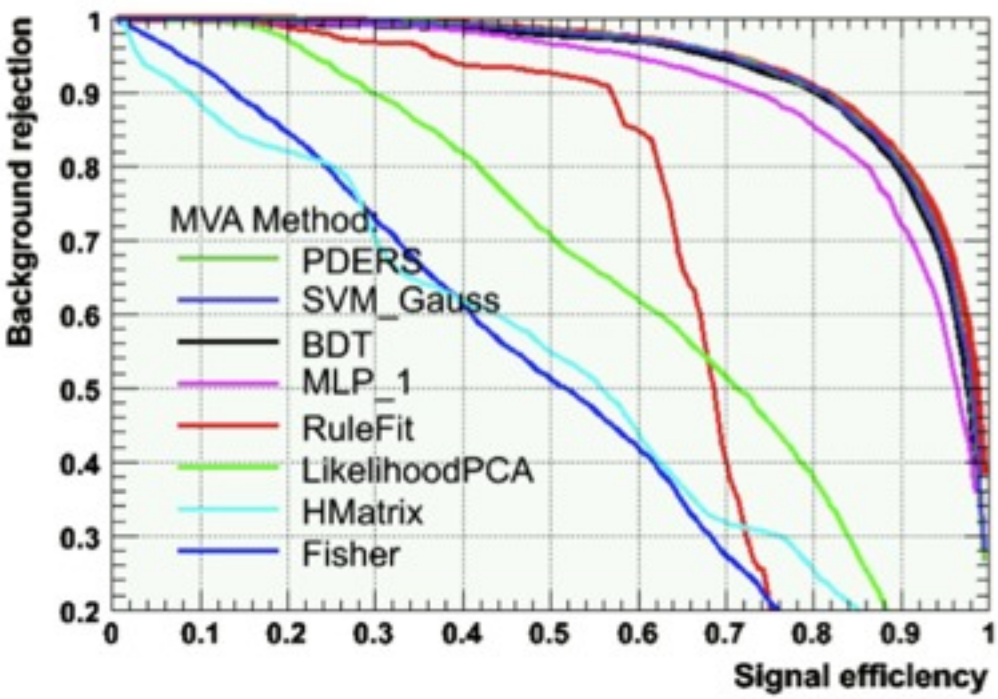
		CLASSIFIERS									
	CRITERIA	Cuts	Likelihood	PDE-RS	k-NN	H-Matrix	Fisher	ANN	BDT	RuleFit	SVM
Performance	No or linear correlations	⋆	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
	Nonlinear correlations	○	○	⋆⋆	⋆⋆	○	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆
Speed	Training	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆	○	⋆	○
	Response	⋆⋆	⋆⋆	○	⋆	⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
Robustness	Overtraining	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	○	⋆	⋆⋆
	Weak variables	⋆⋆	⋆	○	○	⋆⋆	⋆⋆	⋆	⋆⋆	⋆	⋆
Curse of dimensionality		○	⋆⋆	○	○	⋆⋆	⋆⋆	⋆	⋆	⋆	
Transparency		⋆⋆	⋆⋆	⋆	⋆	⋆⋆	⋆⋆	○	○	○	○

Table 1: Assessment of classifier properties. The symbols stand for the attributes “good” (⋆⋆), “fair” (⋆) and “bad” (○). “Curse of dimensionality” refers to the “burden” of required increase in training statistics and processing time when adding more input variables. See also comments in text. The FDA classifier is not represented here since its properties depend on the chosen function.

Example of method comparison

Left figure shows the distribution of signal and background used for test.
Right figure shows the resulting separation using various MVA methods.





The theoretical limit is known from the Neyman-Pearson lemma using the (known/correct) PDFs in a likelihood.
In all fairness, this is a case that is great for the BDT...