Statistical Methods in Particle Physics

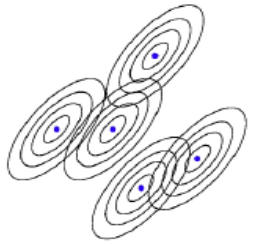
Lecture 9

December 5, 2011

Silvia Masciocchi, GSI Darmstadt
s.masciocchi@gsi.de

Winter Semester 2011 / 12

Outline



Multivariate data analysis methods:

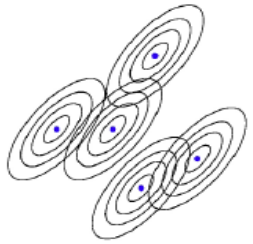
- Fisher discriminant *(linear decision boundary)*
- Neural networks
- Kernel density methods
- Support Vector Machines
- Decision trees:
  - Boosting
  - Bagging
- Toolkit for Multivariate Data Analysis: TMVA
  - Framework for “all” MVA-techniques, available in ROOT

Material from lectures by Helge Voss (May 2009)
http://tmva.sourceforge.net/talks.shtml

→ ***Nik !***

- **Significance tests**

Reminder: multivariate analysis



Neyman-Pearson lemma: statistic test

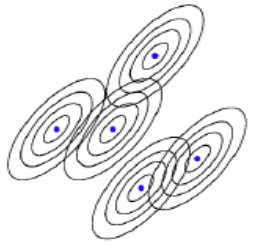
$$t(x) = \frac{P(x|S)}{P(x|B)}$$

Likelihood ratio

But most of the times we do NOT have access to the "true probability density":

- Try to estimate the $PDF_S(x)$ and $PDF_B(x)$ using the Kernel Density Estimators (in limited regions, from training events) → problem coming from the dimensionality!
- Or neglect correlations and use 1-dimensional PDFs
- Or try other approaches to determine the hyperplanes in the feature space, to separate S and B

Probability density estimation (PDE)



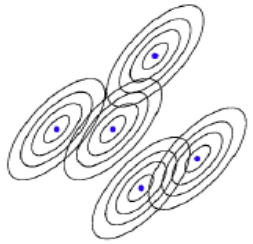
Construct non-parametric estimators of the pdfs $f(\vec{x}|H_0), f(\vec{x}|H_1)$ and use these to construct the likelihood ratio

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

***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 Density Estimator (KDE)

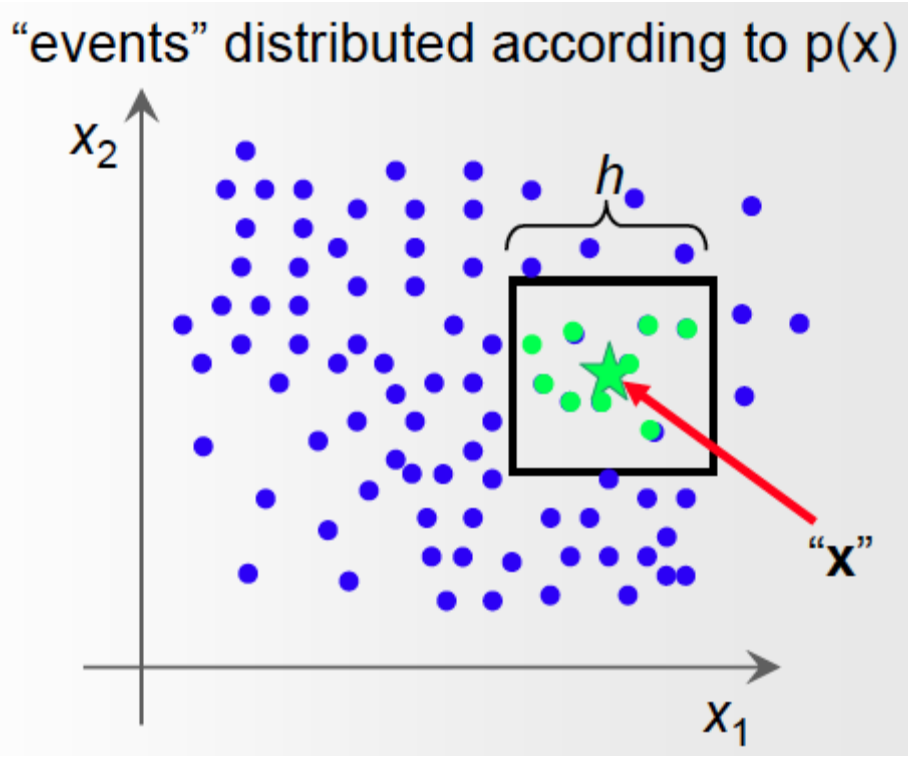


- Trying to estimate the probability density p(x) in D-dimensional space
- The only thing at our disposal is our “training data”
- Say we want to know p(x) at “this” point “x”
- One expects to find in a volume V around point “**x**” N*∫p(x)dx events from a dataset with N events

- Take

$$p(\vec{x}) = \frac{1}{N h^D} \sum_{i=1}^{N} K\left(\frac{\vec{x} - \vec{x}_i}{h}\right)$$

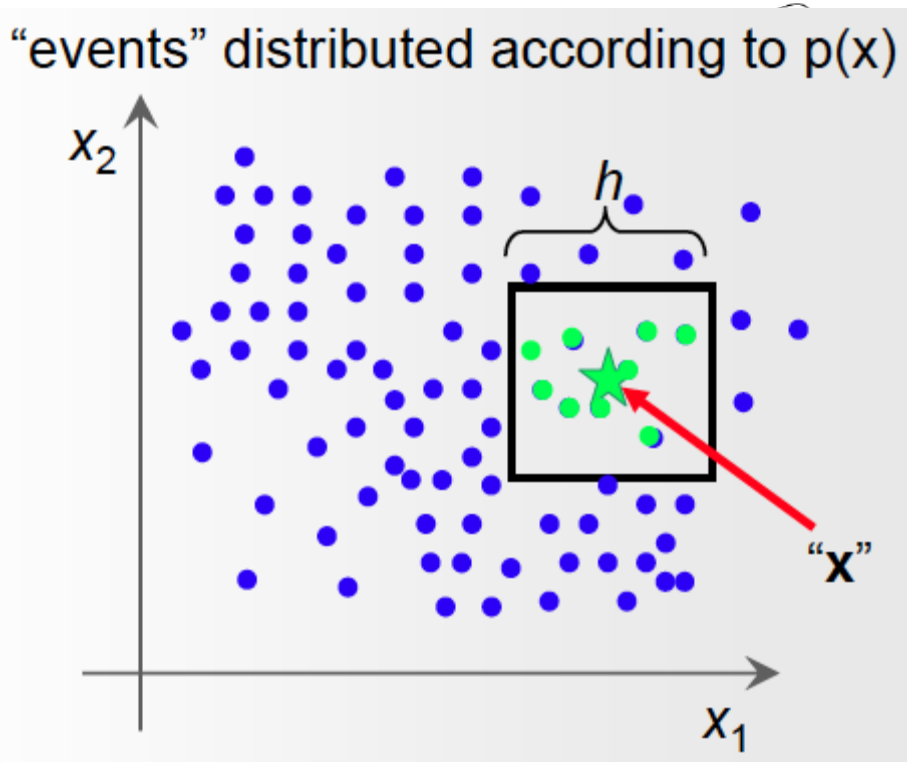
  - K kernel density estimator of the probability density
  - h bandwidth, smoothing parameter



KDE: Parzen window

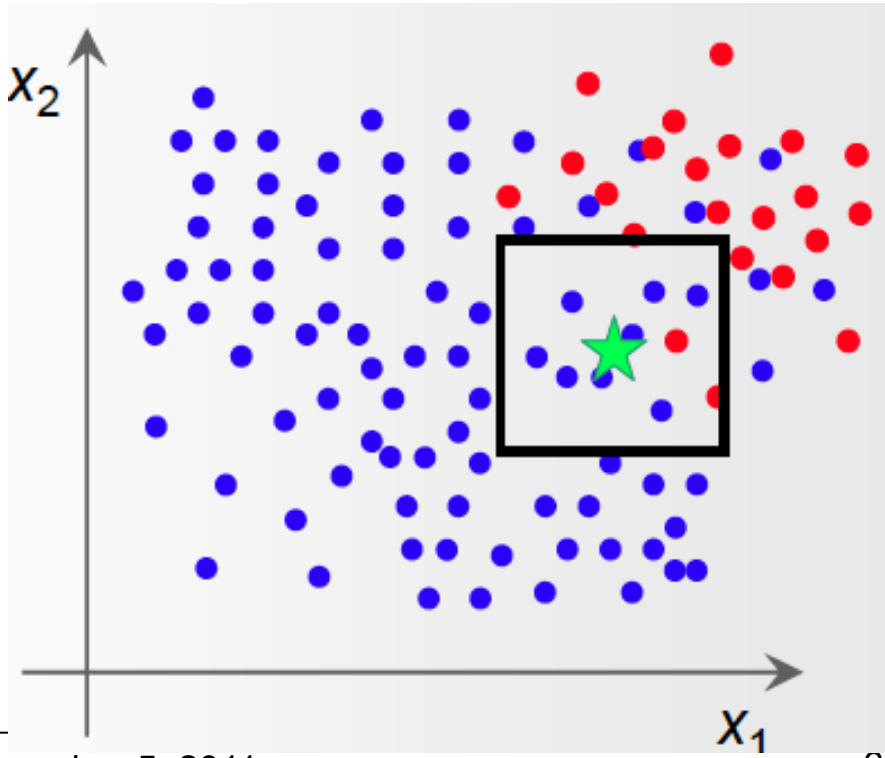
- we choose as volume the square drawn
  → rectangular kernel function
  Parzen window

$$K = \sum_{n=1}^{N} k\left(\frac{x - x_n}{h}\right), \quad \text{with} \quad k(u) = \begin{cases} 1, & |u_i| \le \frac{1}{2}, i = 1 \ldots D \\ 0, & \text{otherwise} \end{cases}$$

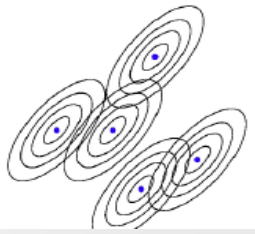


- Determine K from the training data with signal and background mixed together

- Slow: need to sum over N terms
  → take only k-Nearest Neighbours



Kernel Density Estimator



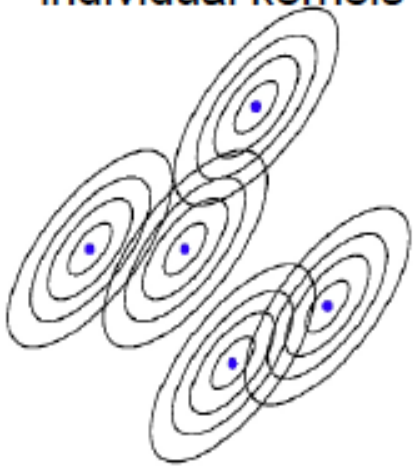
- Parzen Window: "rectangular Kernel" → discontinuities at window edges

→ smoother model for p(x) when using smooth Kernel Fuctions: e.g. Gaussian

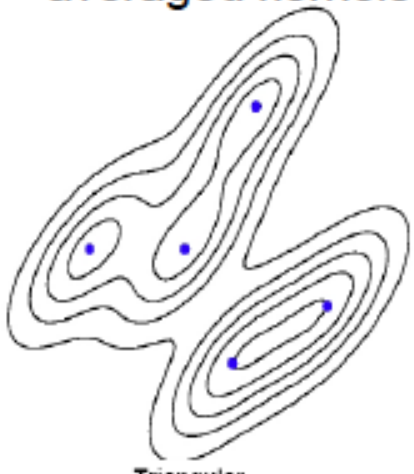
$$p(\mathrm{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{1/2}}\exp\left(\frac{\|\mathrm{x}-\mathrm{x}_n\|^2}{2h^2}\right) \quad \leftrightarrow \text{probability density estimator}$$

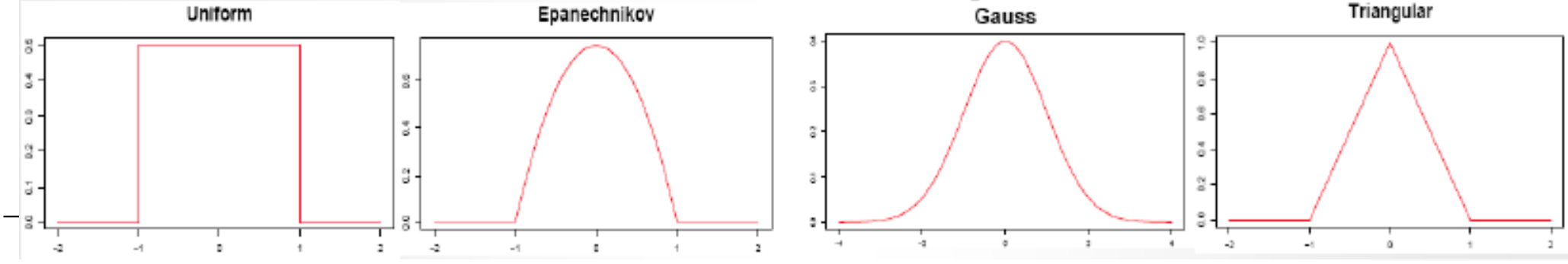
- place a "Gaussian" around each "training data point" and sum up their contributions at arbitrary points "**x**" → p(**x**)
- h: "size" of the Kernel → "smoothing parameter"
- there is a large variety of possible Kernel functions

individual kernels

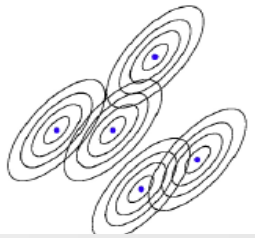


averaged kernels



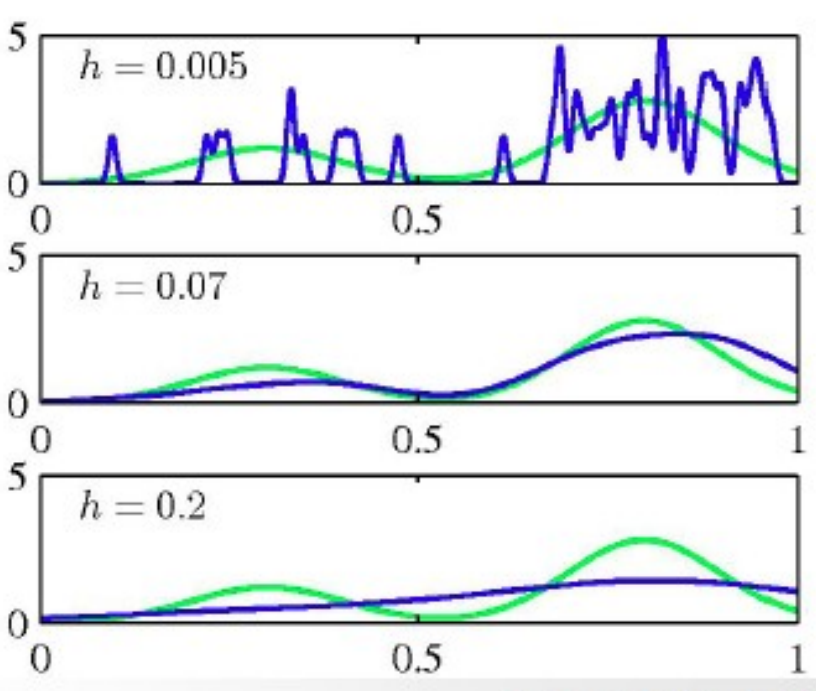


Kernel Density Estimator



$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} K_h(\mathbf{x} - \mathbf{x}_n)$$: a general probability density estimator using kernel K

- h: "size" of the Kernel → "smoothing parameter"
- chosen size of the "smoothing-parameter" → more important than kernel function
- h too small: overtraining
- h too large: not sensitive to features in p(x)



(Christopher M.Bishop)

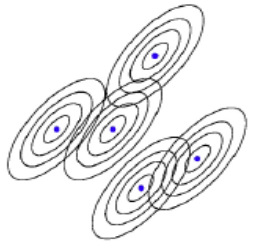
for Gaussian kernels, the optimum in terms of MISE (mean integrated squared error) is given by: $h_{xi}=(4/(3N))^{1/5}\,\sigma_{xi}$ with σ_{xi}=RMS in variable x_i

- a drawback of Kernel density estimators:

Evaluation for any test events involves ALL TRAINING DATA → typically very time consuming

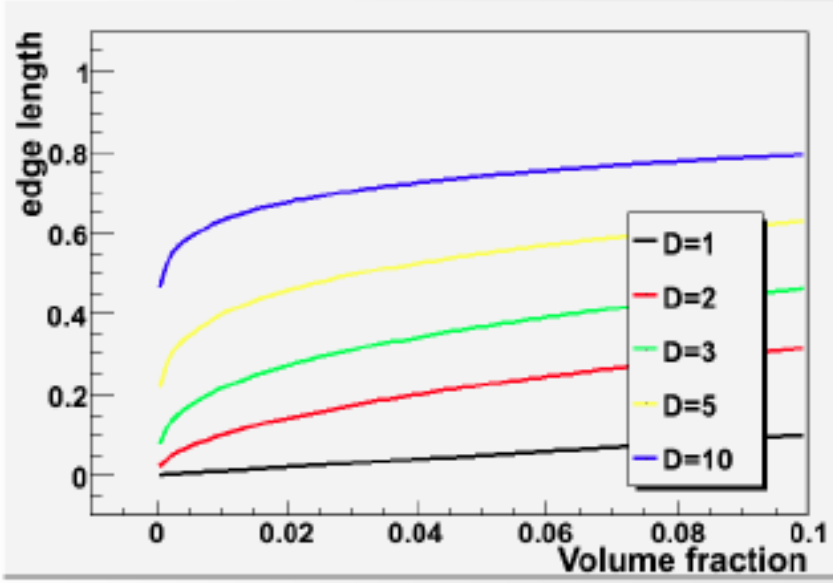
- binary search trees (i.e. Kd-trees) are typically used in kNN methods to speed up searching

“Curse of dimensionality”



Filling a D-dimensional histogram to get a mapping of the PDF is typically unfeasable due to lack of Monte Carlo events.

Shortcoming of nearest-neighbour strategies:



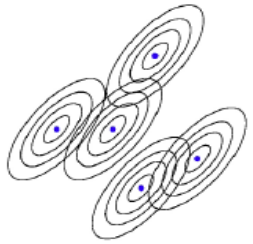
- in higher dimensional classification/regression cases the idea of looking at “training events” in a reasonably small “vicinity” of the space point to be classified becomes difficult:

consider: total phase space volume $V=1^D$
for a cube of a particular fraction of the volume:

$$\text{edge length} = (\text{fraction of volume})^{1/D}$$

- In 10 dimensions: in order to capture 1% of the phase space
→ 63% of range in each variable necessary → that’s not “local” anymore..☹

Correlation versus independence



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

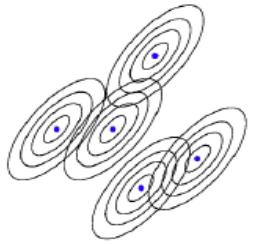
$$p(\vec{x}) = \prod_{i=1}^{N} p_i(x_i)$$

Holds only if the components of x are independent

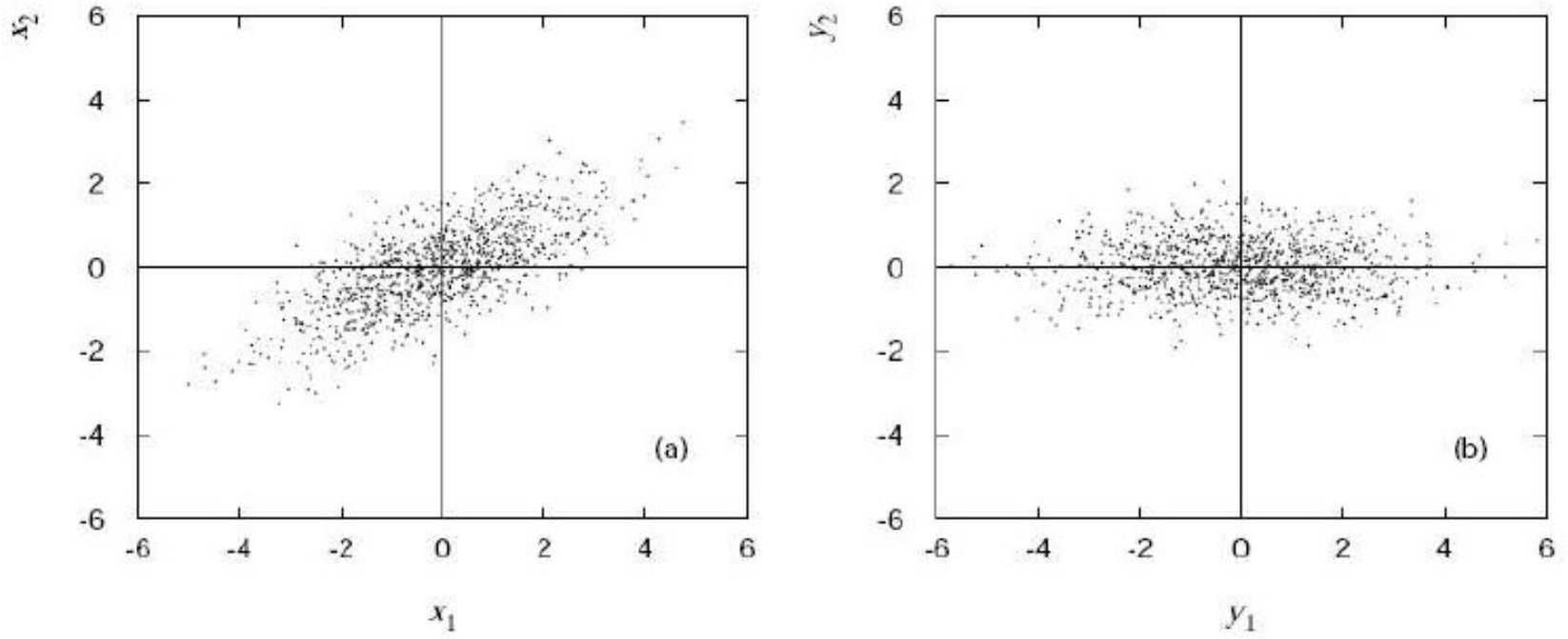
Most importantly, the components of x will generally have non-zero covariances (i.e. they are CORRELATED):

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

Decorrelation of input variables

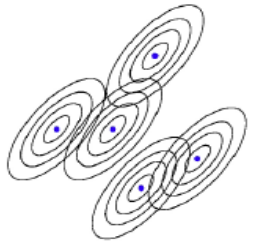


But we can define a set of uncorrelated input variables by a linear transformation, i.e. find the matrix A such that for $\vec{y} = A\,\vec{x}$ the covariances are $\mathrm{cov}[x_i, x_j] = 0$

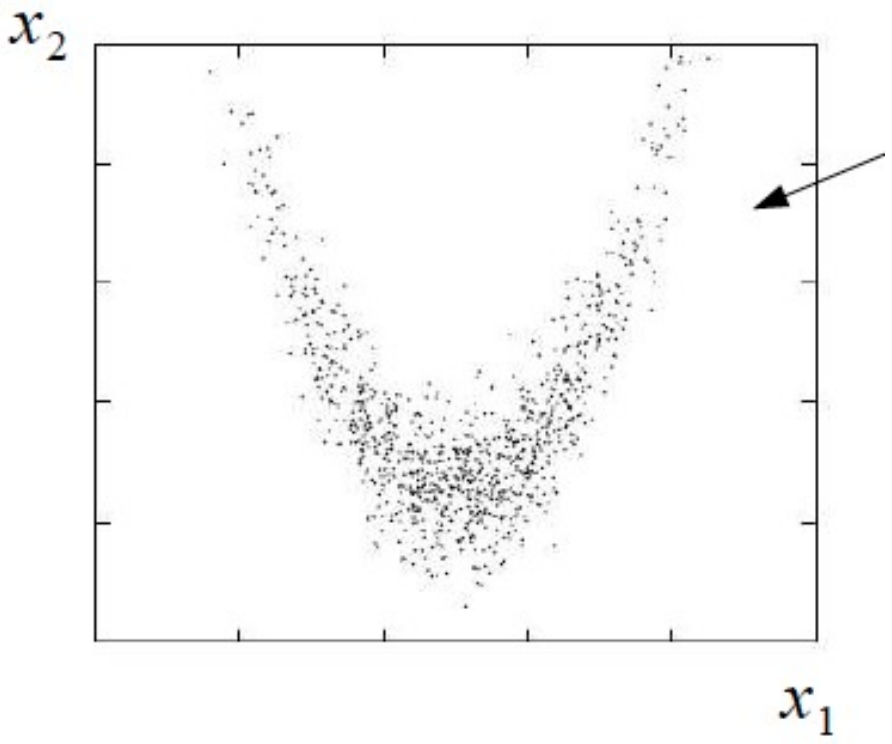


For the following suppose that the variables are "decorrelated" in this way for each of $p(\vec{x}|H_0), p(\vec{x}|H_1)$ separately (since in general their correlations are different).

Decorrelation is not enough



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



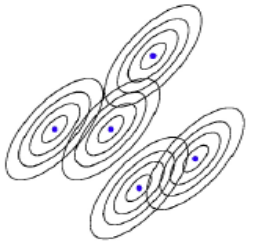
pdf with zero covariance but components still not independent.
In fact:

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

And therefore:

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

Naïve Bayes



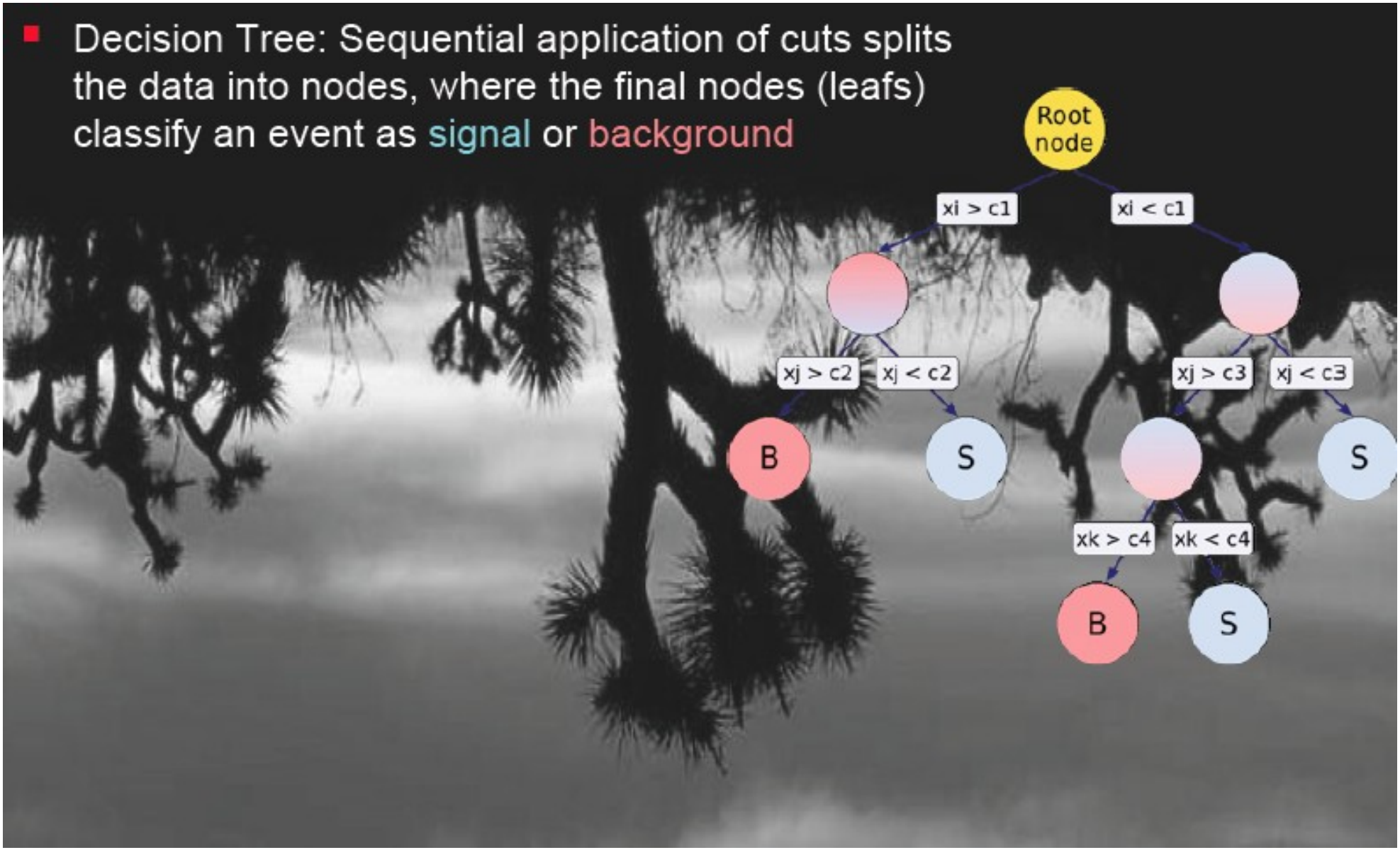
But if the non-linearities are not too great, it is reasonable to first decorrelate the inputs, and take as our estimator for each pdf:

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

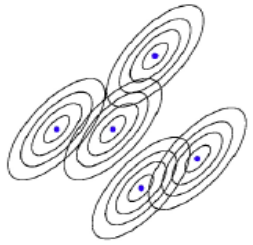
So this at least reduces the problem to one of finding estimates of one-dimensional pdf's.

The resulting estimated likelihood ratio gives the naïve Bayes classifier (in HEP sometimes called the “likelihood method”).

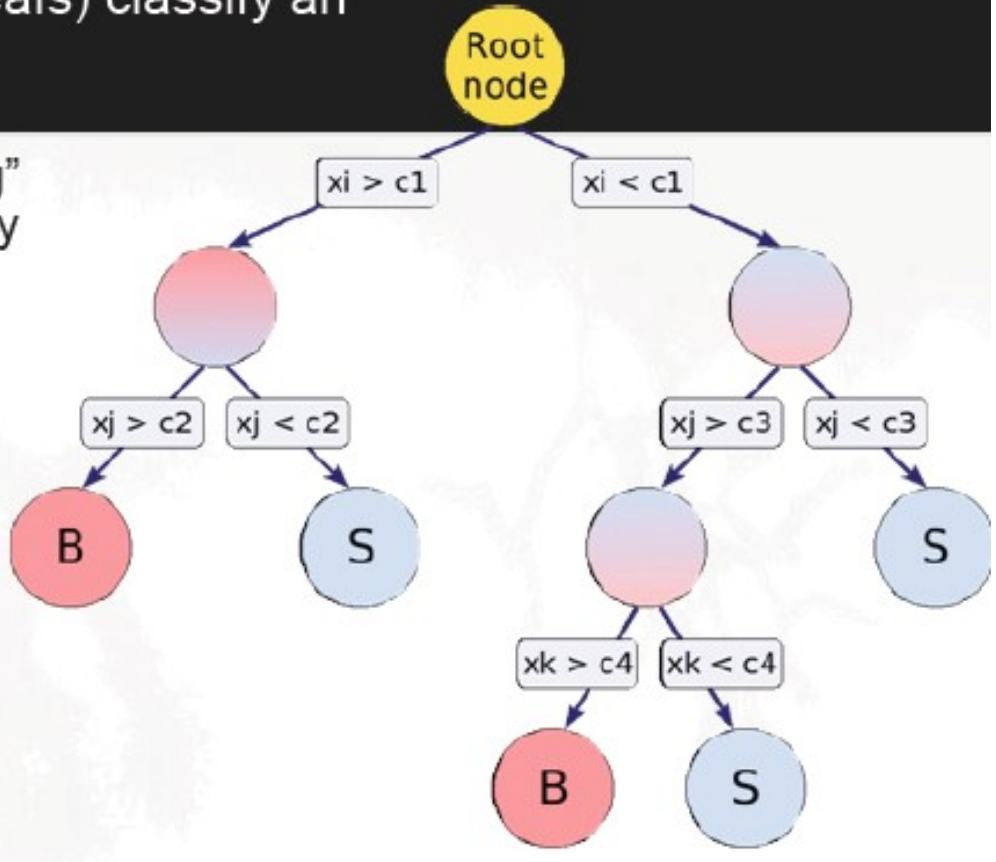
Decision trees



Decision trees



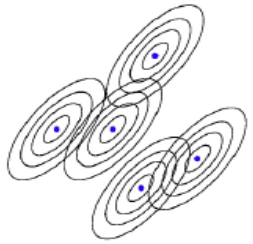
- Decision Tree: Sequential application of cuts splits the data into nodes, where the final nodes (leafs) classify an event as signal or background



- used since a long time in general "data-mining" applications, less known in HEP (although very similar to "simple Cuts")
  - easy to interpret, visualised
  - independent of monotonous variable transformations, immune against outliers
  - weak variables are ignored (and don't (much) deteriorate performance)
- Disadvatage → very sensitive to statistical fluctuations in training data
- Boosted Decision Trees (1996): combine a whole forest of Decision Trees, derived from the same sample, e.g. using different event weights.
  - overcomes the stability problem

→ became popular in HEP since MiniBooNE, B.Roe et.a., NIM 543(2005)

Decision trees



First: in a very simplified way:

- At each step: out of ALL the input variables, find the one for which with a single cut we obtain the best improvement in signal purity

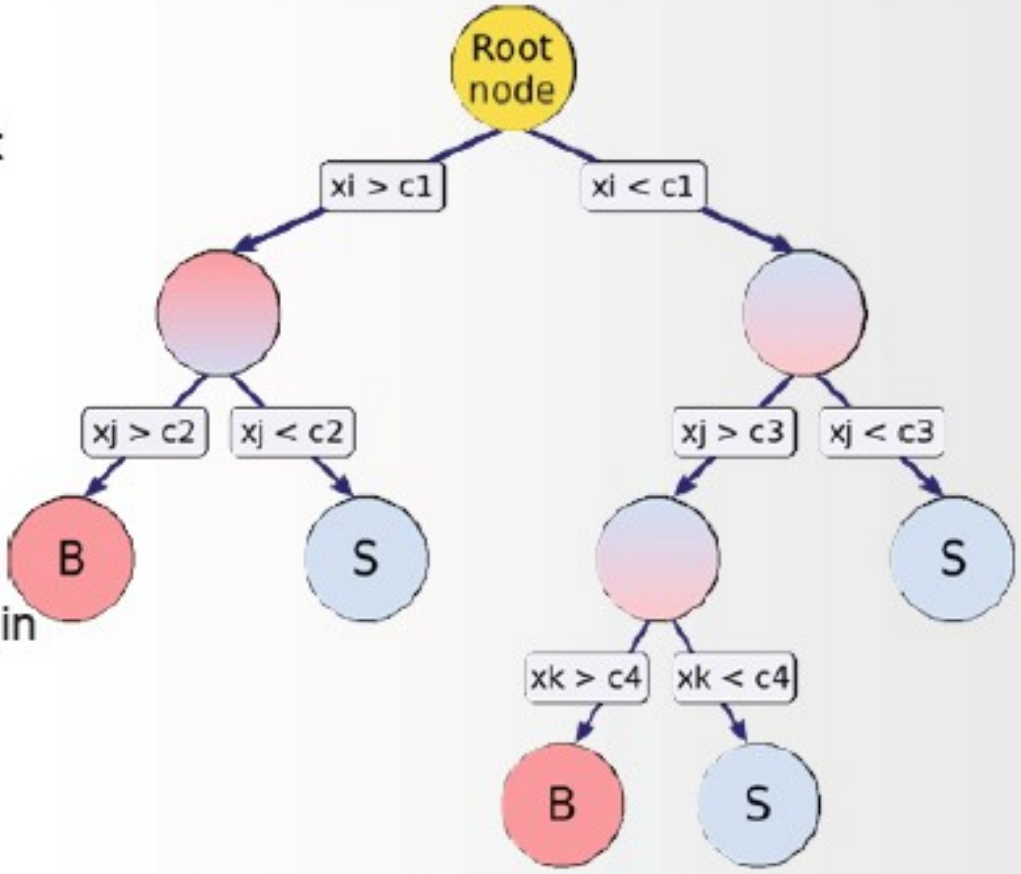
$$P = \frac{\sum_{signal} w_i}{\sum_{signal} w_i + \sum_{background} w_i}$$

  where w_i is the weight of the i'th event
- Iterate until stop criterion reached
  based on e.g. purity reached, or minimum number of events in a node

- The set of cuts defines the decision boundary

Growing a Decision Tree

- start with training sample at the root node
- split training sample at node into two, using a cut in the variable that gives best separation gain
- continue splitting until:
  - minimal #events per node
  - maximum number of nodes
  - maximum depth specified
  - a split doesn't give a minimum separation gain
- leaf-nodes classify S,B according to the majority of events or give a S/B probability



- Why no multiple branches (splits) per node ?
  - → Fragments data too quickly; also: multiple splits per node = series of binary node splits
- What about multivariate splits?
  - → Time consuming
  - → other methods more adapted for such correlatios

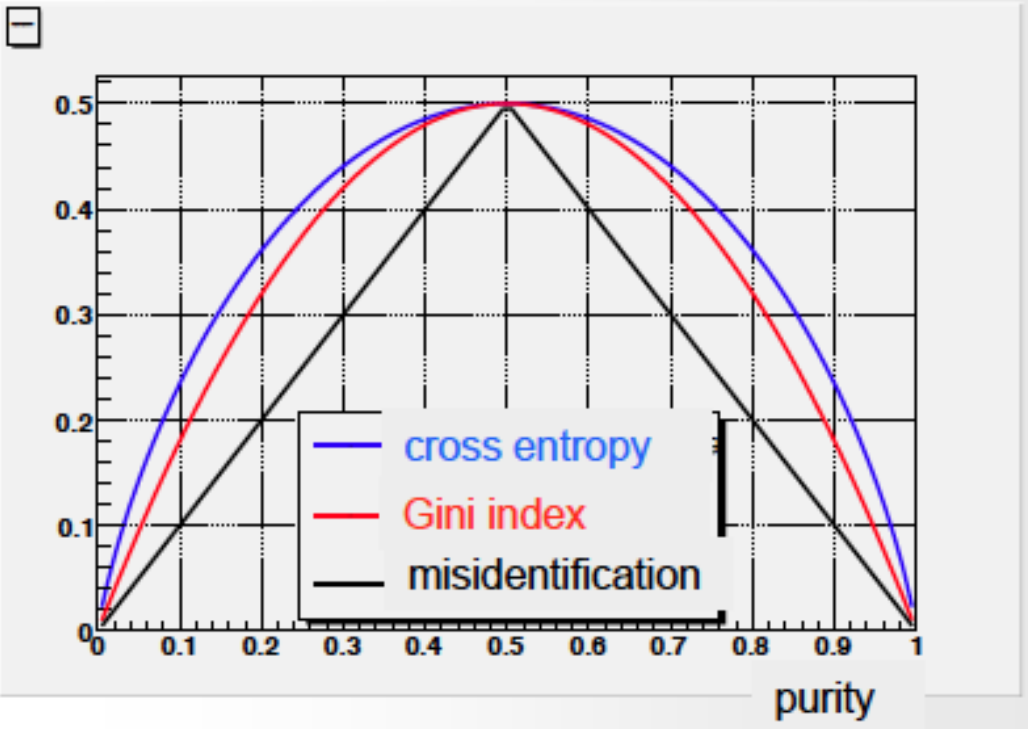
Separation Gain

- What do we mean by "best separation gain"?
- define a measure on how mixed S and B in a node are:
  - Gini-index: (Corrado Gini 1912, typically used to measure income inequality)

    p (1-p) : p=purity
  - Cross Entropy:

    -(plnp + (1-p)ln(1-p))
  - Misidentification:

    1-max(p,1-p)



- difference in the various indices are small, most commonly used: Gini-index

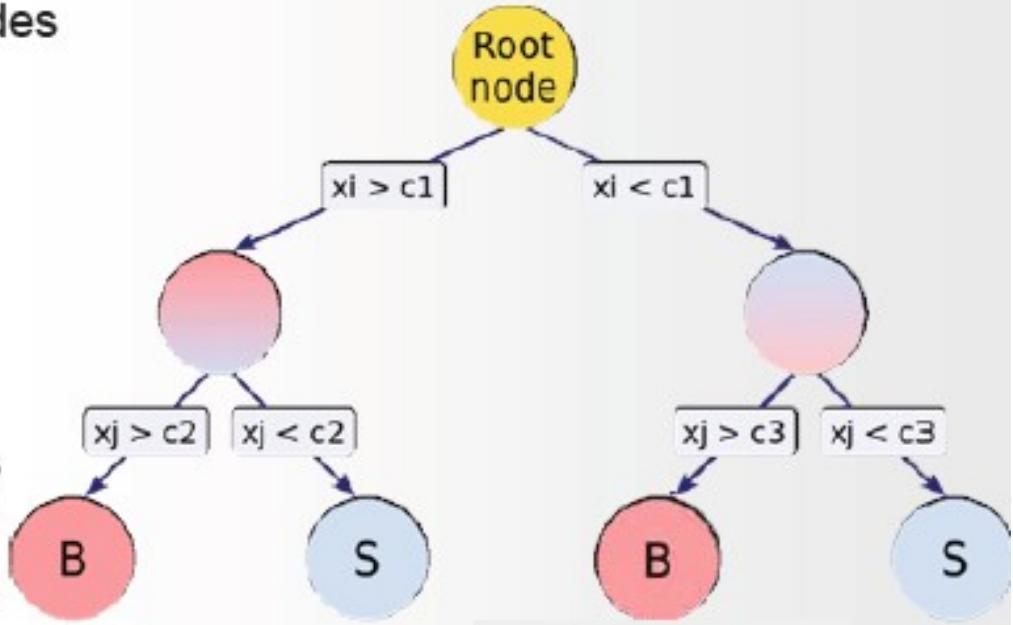
separation gain: e.g. $N_{Parent}*Gini_{Parent} - N_{left}*Gini_{LeftNode} - N_{right}*Gini_{RightNode}$

- Choose amongst all possible variables and cut values the one that maximised the this.

Decision Tree Pruning

- One can continue node splitting until all leaf nodes are basically pure (using the training sample)

  → obviously: that's overtraining



- Two possibilities:
  - stop growing earlier

    generally not a good idea, useless splits might open up subsequent usefull splits
  - grow tree to the end and "cut back", nodes that seem statistically dominated:

    → pruning

- e.g. Cost Complexity pruning:

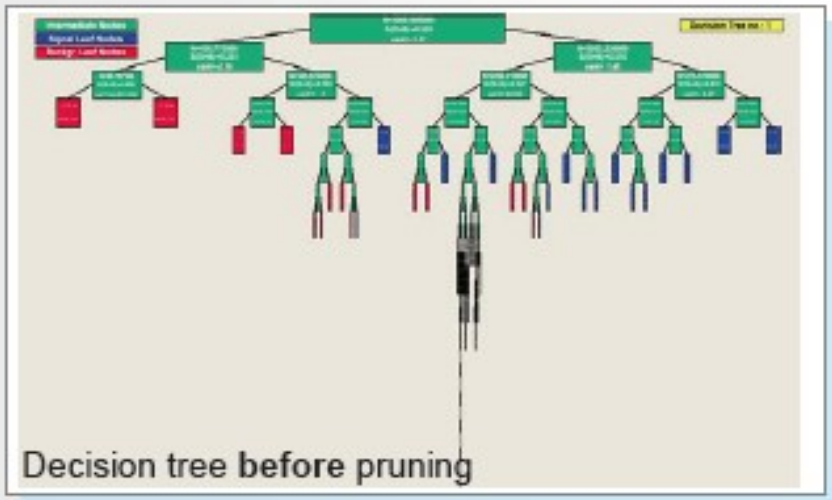
$$C(T,\alpha) = \sum_{\text{leafs of T}} \sum_{\text{events in leaf}} |\, y(x) - y(C) \,| + \alpha N_{\text{leaf nodes}}$$

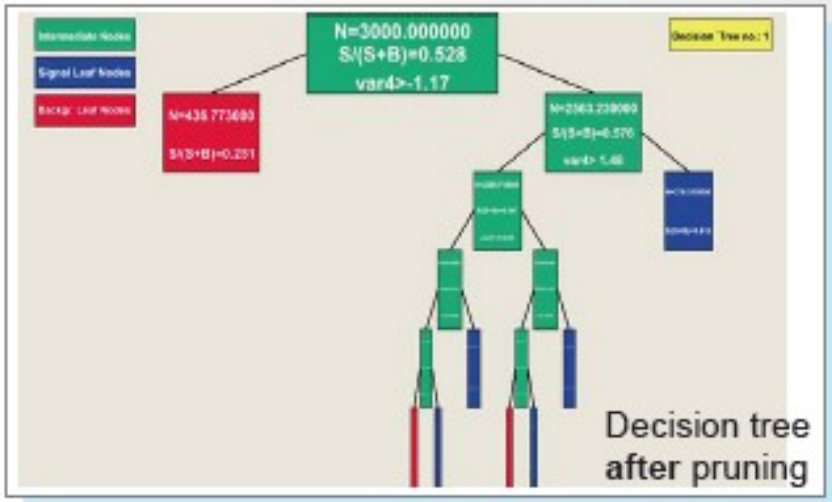
Loss function — regularisaion/ cost parameter

  - assign to every sub-tree, T $C(T,\alpha)$:
  - find subtree T with minmal $C(T,\alpha)$ for given α
  - prune up to value of a that does not show overtraining in the test sample

Decision Tree Pruning

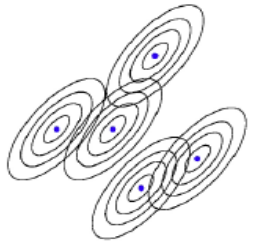
- "Real life" example of an optimally pruned Decision Tree:





- Pruning algorithms are developed and applied on individual trees
  - optimally pruned single trees are not necessarily optimal in a forest !

Decision trees: boosting



The terminal nodes (or 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 the input-variable space (or feature space) as either signal or background

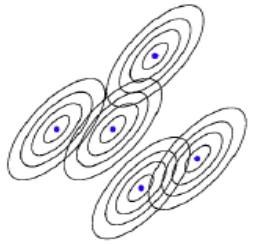
→ A **decision tree classifier**, the with discriminant function:

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \in \text{signal region} \\ -1 & \text{otherwise} \end{cases}$$

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.

Decision trees: 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

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

$x_1, ..., x_N$ event data vectors (each x multivariate)

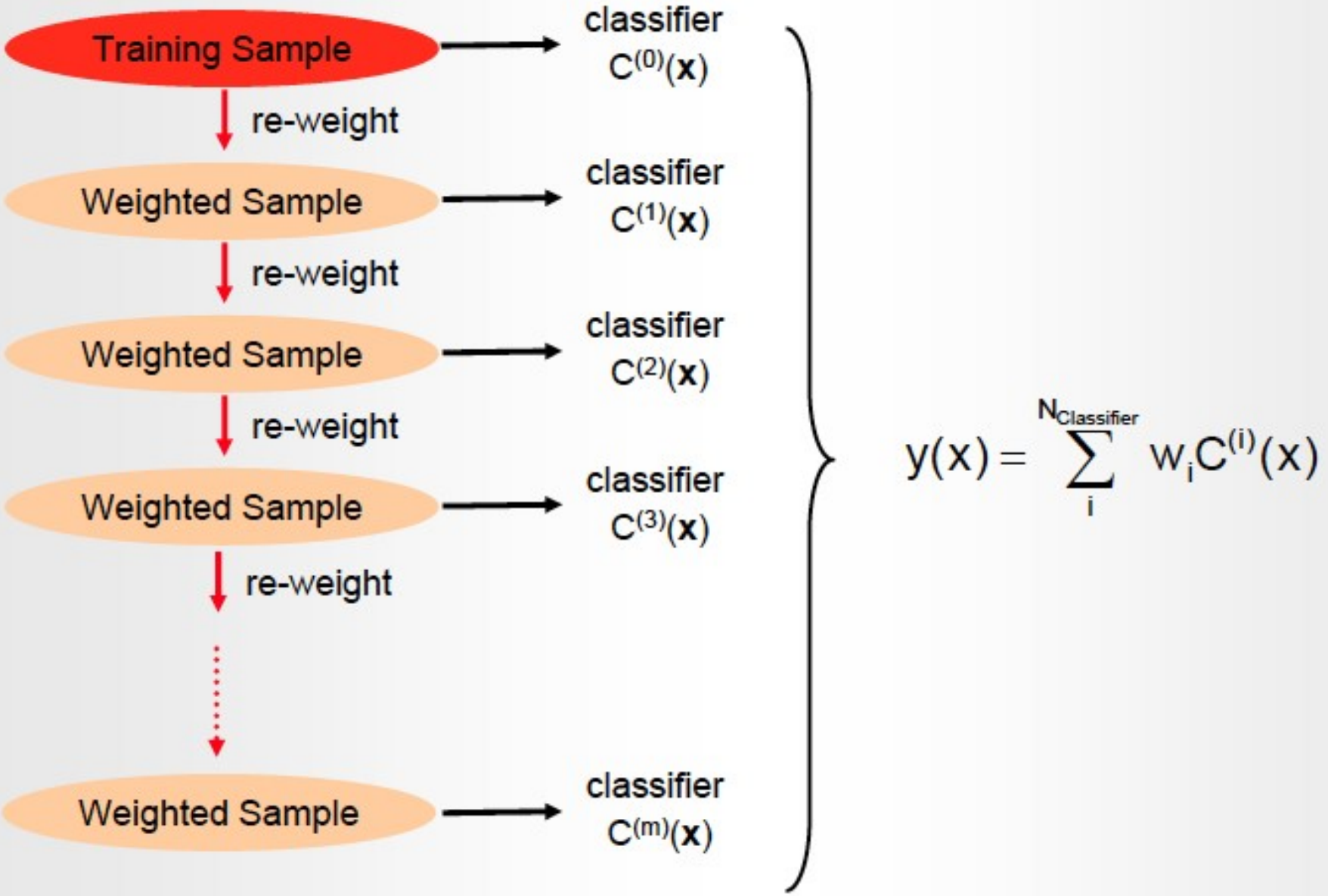
$y_1, \ldots, y_N$ true class labels (+1 for signal, -1 for background)

$w_1, ..., w_N$ event weights

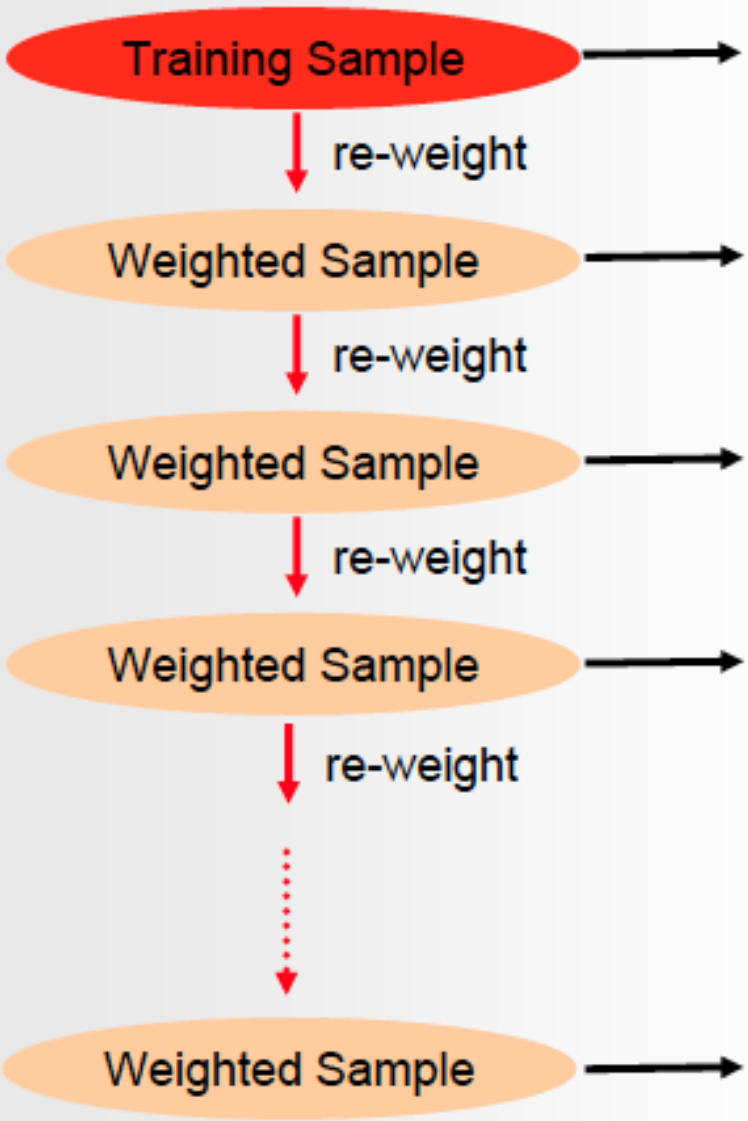
Now define a rule to create from this an ensemble of training samples T1, T2, …

Derive a classifier from each and average them

Boosting


Training Sample
re-weight
Weighted Sample
re-weight
Weighted Sample
re-weight
Weighted Sample
re-weight
Weighted Sample
classifier
$C^{(0)}(\mathbf{x})$
classifier
$C^{(1)}(\mathbf{x})$
classifier
$C^{(2)}(\mathbf{x})$
classifier
$C^{(3)}(\mathbf{x})$
classifier
$C^{(m)}(\mathbf{x})$
$y(x) = \sum_{i}^{N_{Classifier}} w_i C^{(i)}(x)$

Adaptive Boosting (AdaBoost)



- AdaBoost re-weights events misclassified by previous classifier by:

$$\frac{1-f_{err}}{f_{err}} \quad \text{with :}$$

$$f_{err} = \frac{\text{misclassified events}}{\text{all events}}$$

- AdaBoost weights the classifiers also using the error rate of the individual classifier according to:

$$y(x) = \sum_{i}^{N_{Classifier}} \log\left(\frac{1-f_{err}^{(i)}}{f_{err}^{(i)}}\right) C^{(i)}(x)$$

Bagging and Randomised Trees

other classifier combinations:

- Bagging:
  - combine trees grown from "bootstrap" samples

  (i.e re-sample training data with replacement)

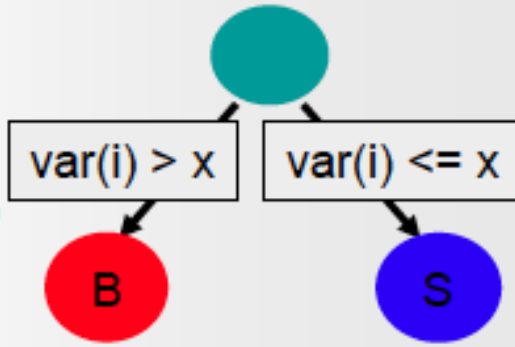
- Randomised Trees: **(Random Forest: trademark L.Breiman, A.Cutler)**
  - combine trees grown with:
    - random subsets of the training data only
    - consider at each node only a random subsets of variables for the split
    - NO Pruning!

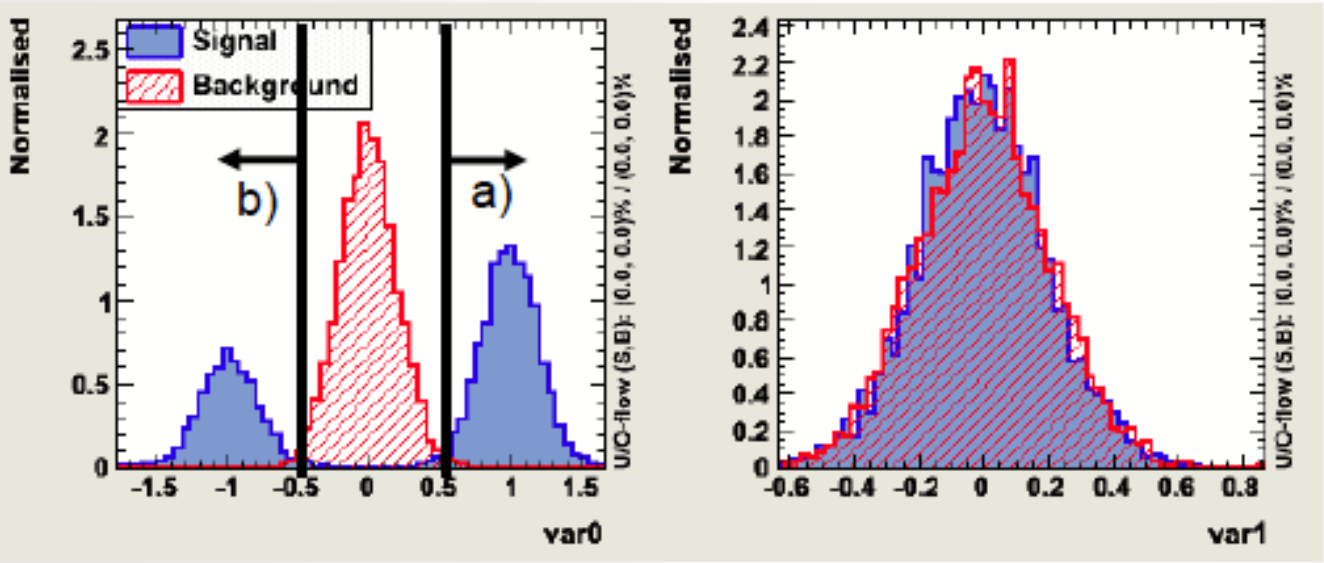
- These combined classifiers work surprisingly well, are very stable and almost perfect "out of the box" classifiers

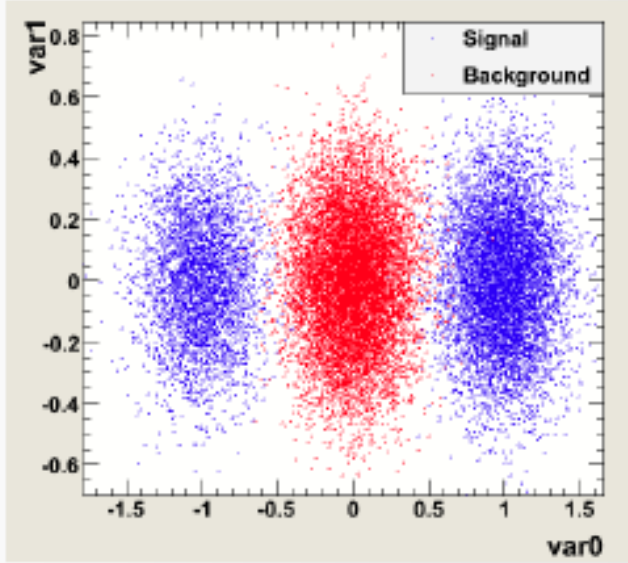
AdaBoost: A simple demonstration

The example: (somewhat artificial…but nice for demonstration) :

- Data file with three "bumps"
- Weak classifier (i.e. one single simple "cut" ↔ decision tree stumps)







Two reasonable cuts: a) Var0 > 0.5 → ε_{signal}=66% ε_{bkg} ≈ 0% misclassified events in total 16.5%

or

b) Var0 < -0.5 → ε_{signal}=33% ε_{bkg} ≈ 0% misclassified events in total 33%

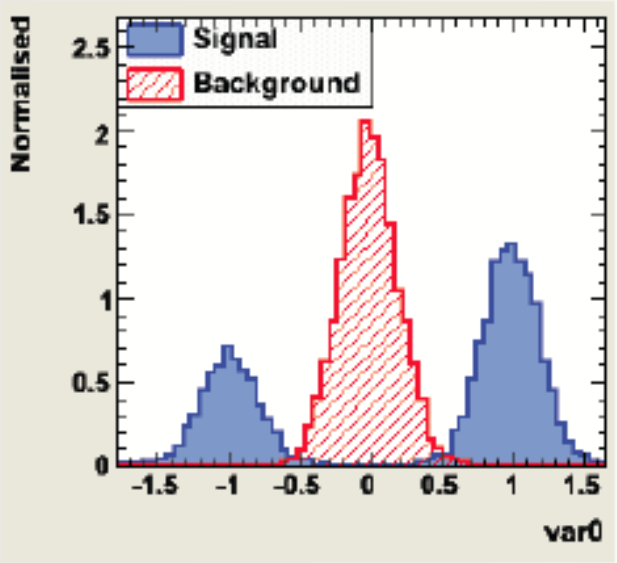
the training of a single decision tree stump will find "cut a)"

AdaBoost: A simple demonstration

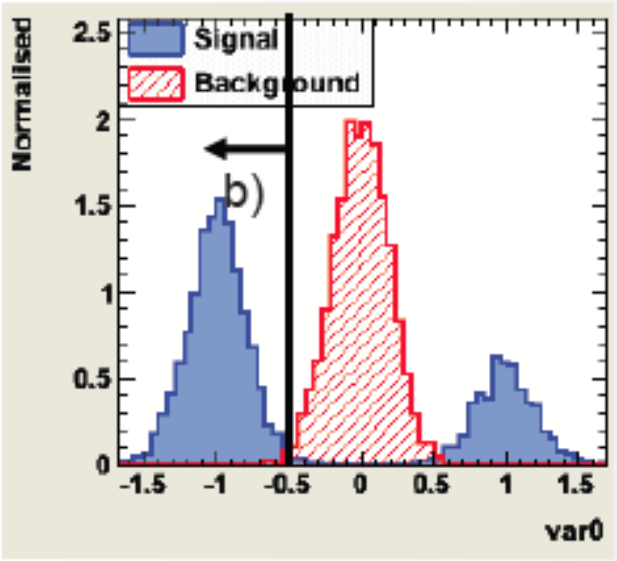
The first "tree", choosing cut a) will give an error fraction: err = 0.165

➔ before building the next "tree": weight wrong classified training events by (1-err/err)) ≈ 5

➔ the next "tree" sees essentially the following data sample:

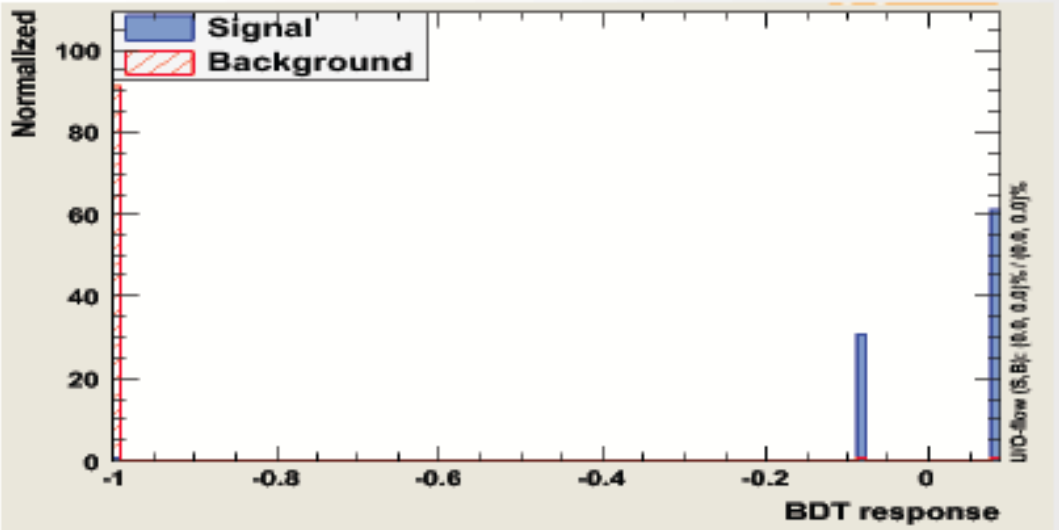


re-weight



.. and hence will chose: "cut b)": Var0 < -0.5

The combined classifier: Tree1 + Tree2
the (weighted) average of the response to a test event from both trees is able to separate signal from background as good as one would expect from the most powerful classifier



AdaBoost vs other Combined Classifiers

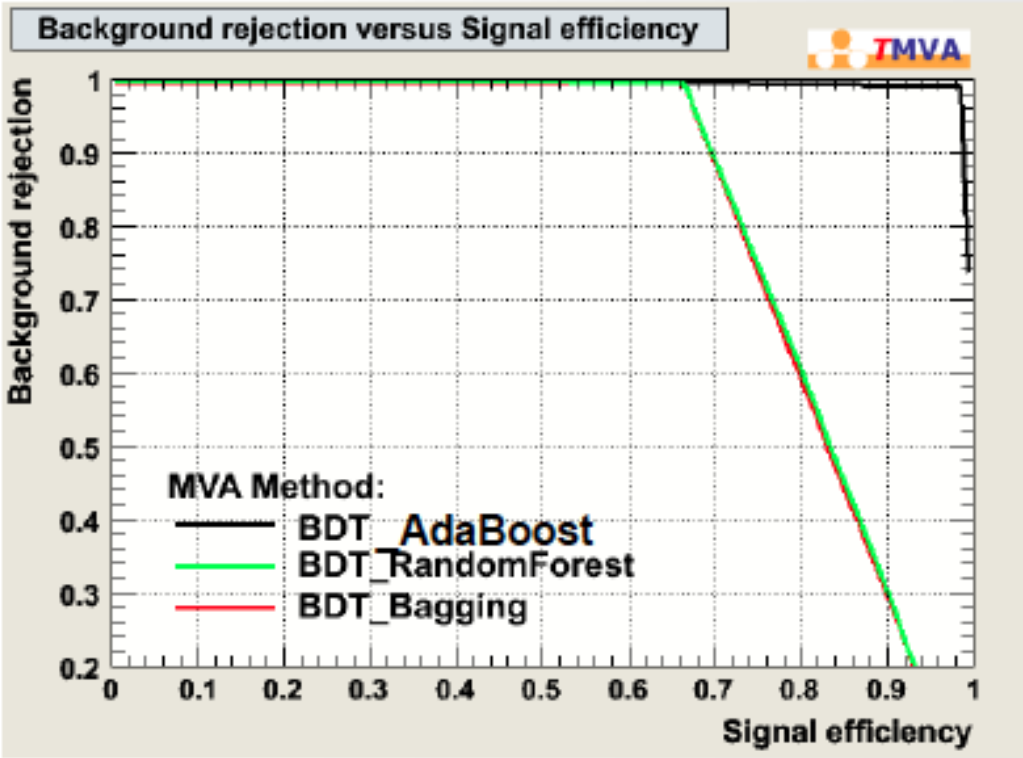
Sometimes people present "boosting" as nothing else then just "smearing" in order to make the Decision Trees more stable w.r.t statistical fluctuations in the training.

→clever "boosting" however can do more, than for example:

- Random Forests
- Bagging

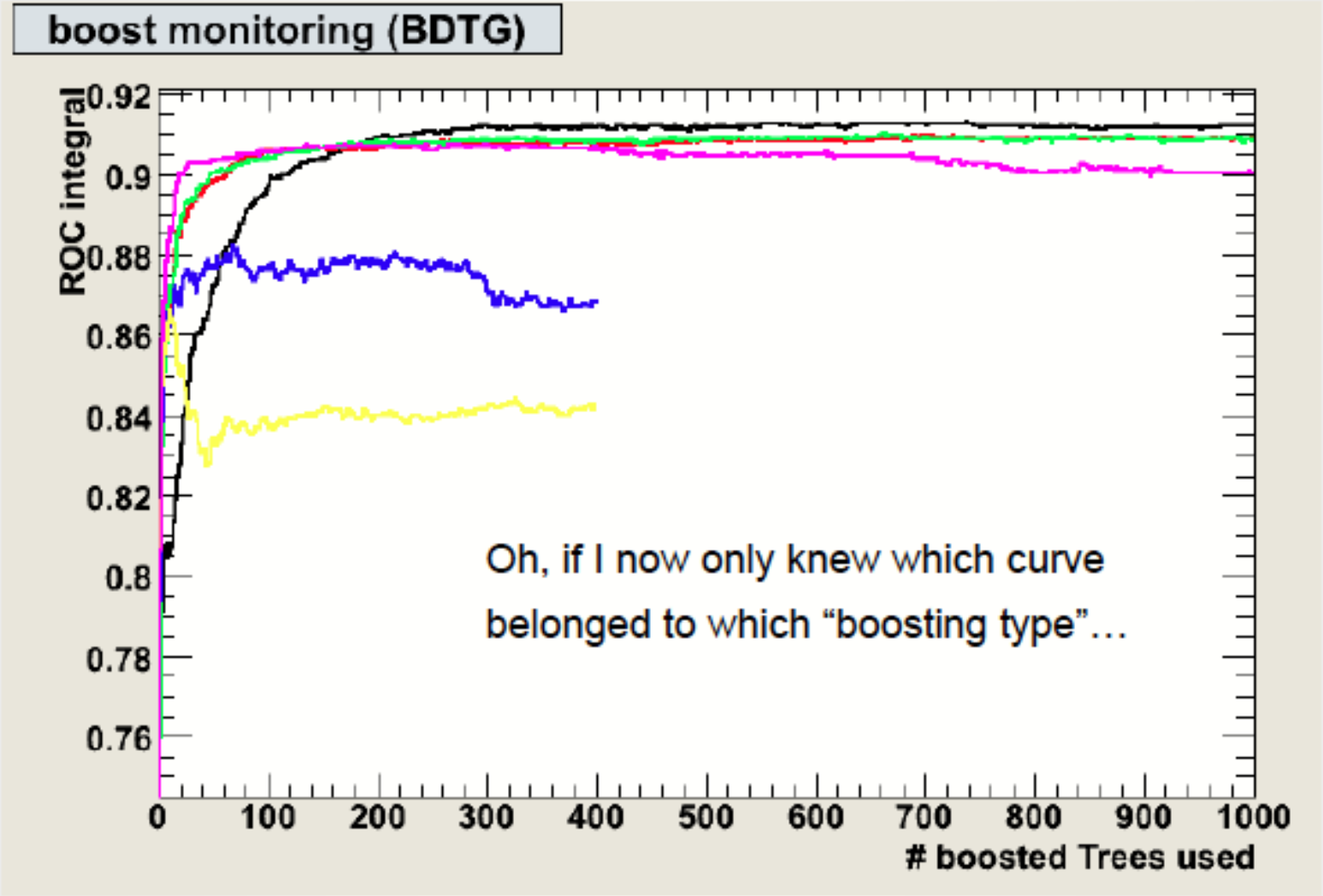
as in this case, pure statistical fluctuations are not enough to enhance the 2nd peak sufficiently

however: a "fully grown decision tree" is much more than a "weak classifier"
→ "stabilization" aspect is more important

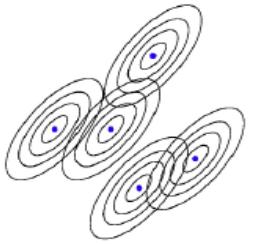


Surprisingly: Often using smaller trees (weaker classifiers) in AdaBoost and other clever boosting algorithms (i.e. gradient boost) seems to give overall significantly better performance !

Boosting at Work

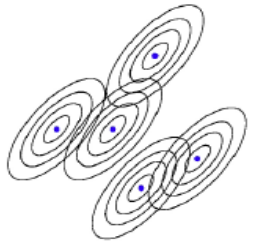


Decision trees: summary



- 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 and are effectively ignored.
- Easy to deal with inputs of mixed types (real, integer, categorical, …)
- It a tree had only a few leaves it is easy to visualize (but rarely we 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 the weaker classifiers: Bagging (Bootstrap-Aggregating) generates the ensemble of classifiers by random sampling with replacement from the full training sample.

Significance tests, goodness-of-fit

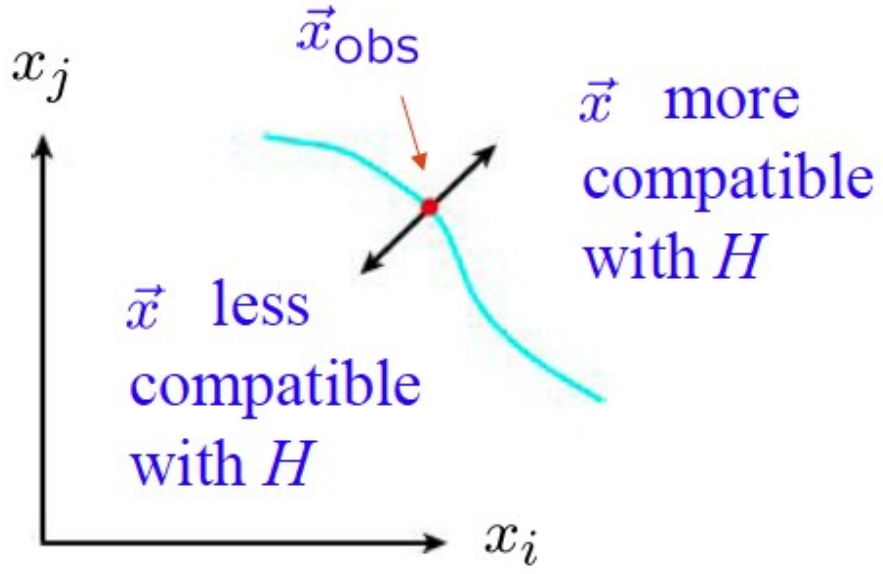


Suppose hypothesis *H* predicts pdf $f(\vec{x}|H)$ for a set of observations $\vec{x} = (x_1, \ldots, x_n)$

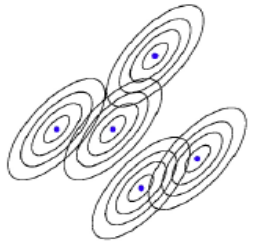
We observe a single point in this space: $\vec{x}_{obs}$

What can we say about the validity of *H* in light of the data?

Decide what part of the data space represents less compatibility with *H* than does the point $\vec{x}_{obs}$ (Not unique!)



p-values



Express 'goodness-of-fit' by giving the p-value for H:

p = probability, under assumption of H, to observe data with equal or lesser compatibility with H relative to the data we got

NOTE! This is NOT the probability that H is true!

In frequentist statistics we don't talk about P(H) (unless H represents a repeatable observation).

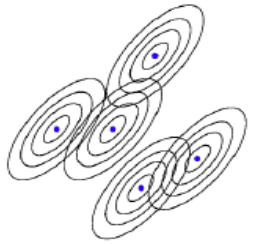
In Bayesian statistics we do. Use Bayes' theorem to obtain

$$P(H|\vec{x}) = \frac{P(\vec{x}|H)\,\pi(H)}{\int P(\vec{x}|H)\,\pi(H)\,dH}$$

where π(H) is the prior probability for H.

For now stick with the frequentist approach.

p-value example



Testing whether a coin is "fair"

Probability to observe n heads in N coin tosses is binomial:

$$P(n;p,N) = \frac{N!}{n!\,(N-n)!}\,p^n\,(1-p)^{N-n}$$

Hypothesis H: the coin is fair (p=0.5)

Suppose we toss the coin N=20 times and get n=17 heads.

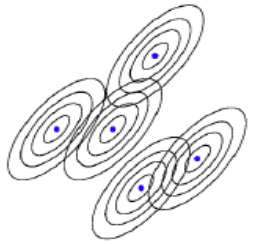
Region of data space with equal or lesser compatibility with H relative to n=17 is: n= 17, 18, 19, 20, 0, 1, 2, 3

Adding up the probabilities for these values gives:

P(n=0, 1, 2, 3, 17, 18, 19, or 20) = 0.0026

i.e. p=0.0026 is the probability of obtaining such a bizarre result (or more so) "by chance", under the assumption of H

Significance of an observed signal



Suppose we observe n events. These can consist of:

n_b events from known processes (background)

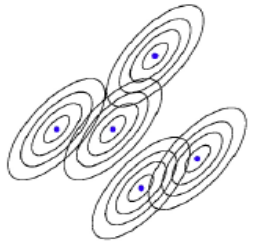
n_s events from a new process (signal)

If n_s, n_b are Poisson random variables with means s, b, then $n=n_s+n_b$ is also Poisson, with mean s+b

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

Suppose b=0.5, and we observe nobs=5. Should we claim evidence for a new discovery?

Significance of an observed signal



Suppose we observe n events. These can consist of:

- n_b events from known processes (background)
- n_s events from a new process (signal)

If n_s, n_b are Poisson random variables with means s, b, then $n=n_s+n_b$ is also Poisson, with mean s+b

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

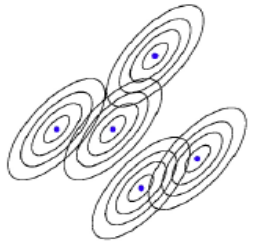
Suppose b=0.5, and we observe nobs=5. Should we claim evidence for a new discovery?

Give p-value for hypothesis s=0:

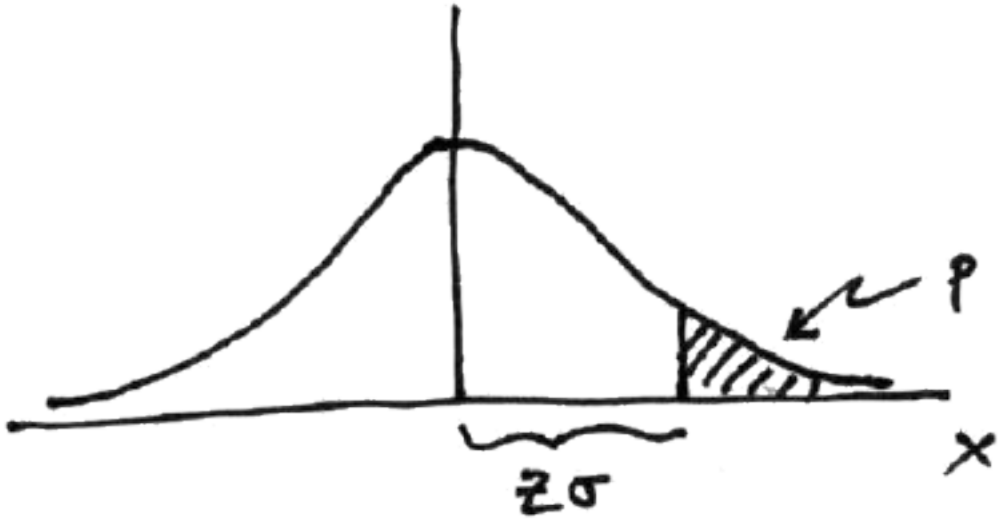
p-value = P($n \geq 5$; b=0.5, s=0)

= 1.7×10^{-4} $\neq$ P(s=0) !!

Significance from p-value



Often define significance Z as the number of standard deviations that a Gaussian variable would fluctuate in one direction to give the same p-value

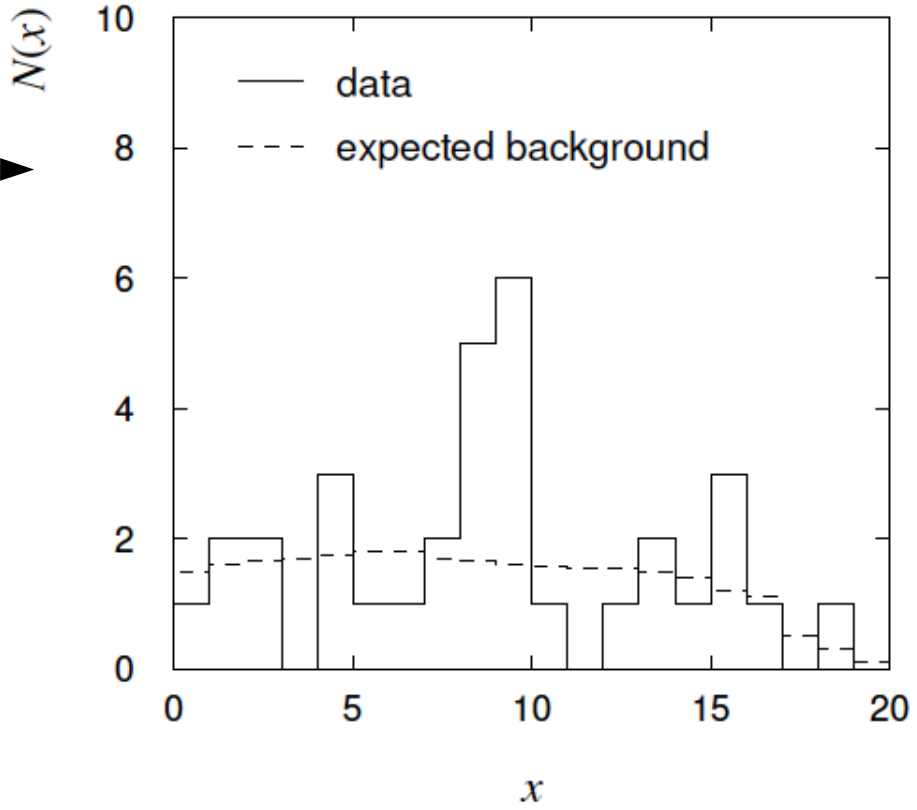


$$p = \int_Z^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx = 1 - \Phi(Z)$$ `1 - TMath::Freq`

$$Z = \Phi^{-1}(1-p)$$ `TMath::NormQuantile`

The significance of a peak

Suppose we measure a value x for each event and find:



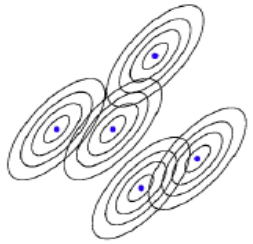
Each bin (observed) is a Poisson r.v., means are given by the dashed line

In the two bins with the peak, 11 entries found with b = 3.2

The p-value for the s=0 hypothesis is:

$$P(n \geq 11; b=3.2, s=0) = 5.0 \times 10^{-4}$$

The significance of a peak - 2



But … did we know where to look for the peak?

→ give P(n≥11) in any 2 adjacent bins

Is the observed width consistent with the expected x resolution?

→ take x window several times the expected resolution

How many bins x distributions have we looked at?

→ look at a thousand of them, you'll find a 10^{-3} effect

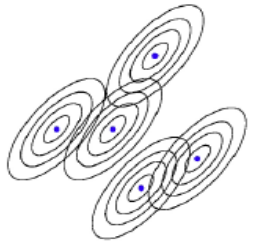
Did we adjust the cuts to “enhance” the peak?

→ freeze cuts, repeat analysis with new data

How about the bins to the sides of the peak … (too low!)

Should we publish ??

When to publish



HEP folklore is to claim discovery when p= 2.9 x 10^{-7}, corresponding to a significance Z=5.

This is very subjective and really should depend on the prior probability of the phenomenon in question, e.g.

Phenomenon	Reasonable p-value for discovery
D^0D^0 mixing	~0.05
Higgs	~10^{-7} (?)
Life on Mars	~10^{-10}
Astrology	~10^{-20}

One should also consider the degree to which the data are compatible with the new phenomenon, not only the level of disagreement with the null-hypothesis: p-value is only the first step !!!

Distribution of the p-value

The p-value is a function of the data, and is thus itself a random variable with a given distribution. Suppose the p-value of H is found from a test statistic t(x) as:

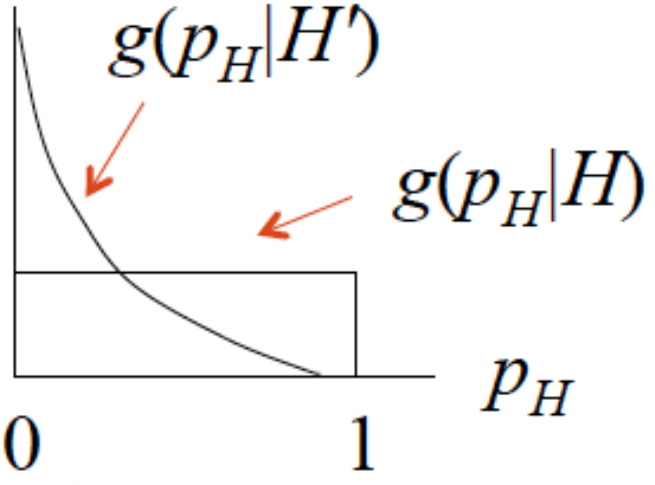
$$p_H = \int_t^{\infty} f(t'|H)\, dt'$$

The pdf of p_H under assumption of H is:

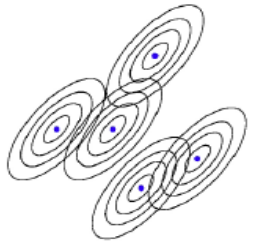
$$g(p_H|H) = \frac{f(t|H)}{\left|\partial p_H / \partial t\right|} = \frac{f(t|H)}{f(t|H)} = 1 \qquad (0 \le p_H \le 1)$$

In general for continuous data, under assumption of H, p_H ~ uniform in [0,1]

And is concentrated toward zero for some (broad) class of alternatives



Using a p-value to define test of H_0



The probability to find the *p*-value of H_0, p_0, less than α is

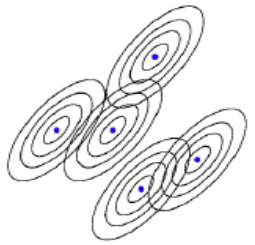
$$P(p_0 \leq \alpha | H_0) = \alpha$$

We started by defining critical region in the original data space (***x***), then reformulated this in terms of a scalar test statistic $t(\boldsymbol{x})$.

We can take this one step further and define the critical region of a test of H_0 with size α as the set of data space where $p_0 \leq \alpha$.

Formally the *p*-value relates only to H_0, but the resulting test will have a given power with respect to a given alternative H_1.

Pearson's X^2 statistics



Test statistic for comparing observed data $\vec{n} = (n_1, \ldots, n_N)$
(n_i independent) to predicted mean values $\vec{\nu} = (\nu_1, \ldots, \nu_N)$

$$\chi^2 = \sum_{i=1}^{N} \frac{(n_i - \nu_i)^2}{\sigma_i^2}, \text{ where } \sigma_i^2 = V[n_i]$$

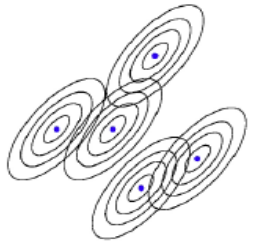
Pearson's χ^2 statistic

χ^2 = sum of squares of the deviations of the I'th measurement from the I'th prediction, using σ_i as the 'yardstick' for comparison

For $n_i \sim$ Poisson (ν_i) we have $V[n_i] = \nu_i$, so this becomes:

$$\chi^2 = \sum_{i=1}^{N} \frac{(n_i - \nu_i)^2}{\nu_i}$$

Pearson's X^2 test



If n_i are Gaussian with mean v_i and standard deviation σ_i, namely $n_i \sim N(v_i, \sigma_i^2)$, then Pearson's X^2 will follow the X^2 pdf (here for $X^2 = z$)

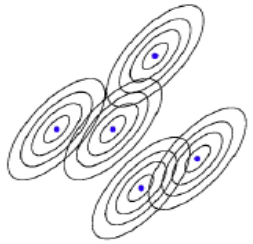
$$f_{X^2}(z;N) = \frac{1}{2^{N/2}\Gamma(N/2)} z^{N/2-1} e^{-z/2}$$

If the n_i are Poisson with $v_i >> 1$ (in practice OK for $v_i > 5$) then the Poisson distribution becomes Gaussian and therefore Pearson's X^2 statistic here as well follows the X^2 pdf.

The X^2 value obtained from the data then gives the p-value:

$$p = \int_{X^2}^{\infty} f_{X^2}(z;N)\,dz$$

The X^2 per degree of freedom



Recall that for the chi-square pdf for N degrees of freedom

$$E[z] = N, \quad V[z] = 2N$$

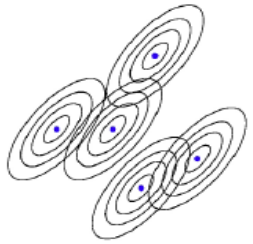
This makes sense; if the hypothesized v_i are right, the rms deviation of n_i from v_i is σ_i, so each term in the sum contributes ~ 1.

One often sees χ^2/N reported as a measure of goodness-of-fit.
BUT! Better to give the χ^2 and N separately. Consider e.g.:

$$X^2 = 15, \text{ N=10} \quad \rightarrow \quad \text{p-value} = 0.13$$
$$X^2 = 150, \text{ N=100} \quad \rightarrow \quad \text{p-value} = 9.0 \times 10^{-4}$$

i.e. for N large, even a X^2 per dof only a bit greater than one can imply a small p-value, i.e., poor goodness-of-fit

Summary of p-value



- Introduction to significance tests:
  - p-value expresses the level of agreement between data and hypothesis
  - p-value is NOT the probability of the hypothesis!
- P-value can be used to define a critical region, i.e. region of data space where $p<\alpha$
- Widely used X^2 test:
  - Statistic = sum of (data – prediction)2 / variance
  - Often X^2 chi-square pdf → use to get p-value
  - Otherwise may need to use MC