2 *'Event selection'*

Wouter Verkerke, NIKHEF

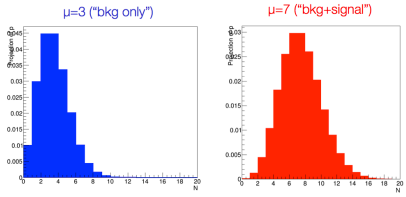
Roadmap for this course

- Start with basics, gradually build up to complexity of

Statistical tests with simple hypotheses for counting data	*"What do we mean with probabilities"*
Statistical tests with simple hypotheses for distributions	*"p-values"*
Hypothesis testing as basis for event selection	*"Optimal event selection & machine learning"*
Composite hypotheses (with parameters) for distributions	*"Confidence intervals, Maximum Likelihood"*
Statistical inference with nuisance parameters	*"Fitting the background"*
Response functions and subsidiary measurements	*"Sideband fits and systematic uncertainties"*

Probabilities vs conditional probabilities

- Note that probability models strictly give *conditional* probabilities (with the condition being that the underlying hypothesis is true)



*Definition: P(data|hypo) is called the **likelihood***

$$P(N) \rightarrow P(N \mid H_{bkg}) \qquad P(N) \rightarrow P(N \mid H_{sig+bkg})$$

- Suppose we measure N=7 then can calculate

$L(N=7|H_{bkg})=2.2\%$ $L(N=7|H_{sig+bkg})=14.9\%$

- *Data is more likely under sig+bkg hypothesis than bkg-only hypo*
- Is this what we want to know? Or do we want to know $L(H_{s+b}|N=7)$?

Wouter Verkerke, NIKHEF

Interpreting probabilities

- We have seen

  probabilities assigned observed experimental outcomes
  (probability to observed 7 events under some hypothesis)

  probabilities assigned to hypotheses
  (prior probability for hypothesis H_{sb} is 50%)

  which are conceptually different.

- How to interpret probabilities – two schools

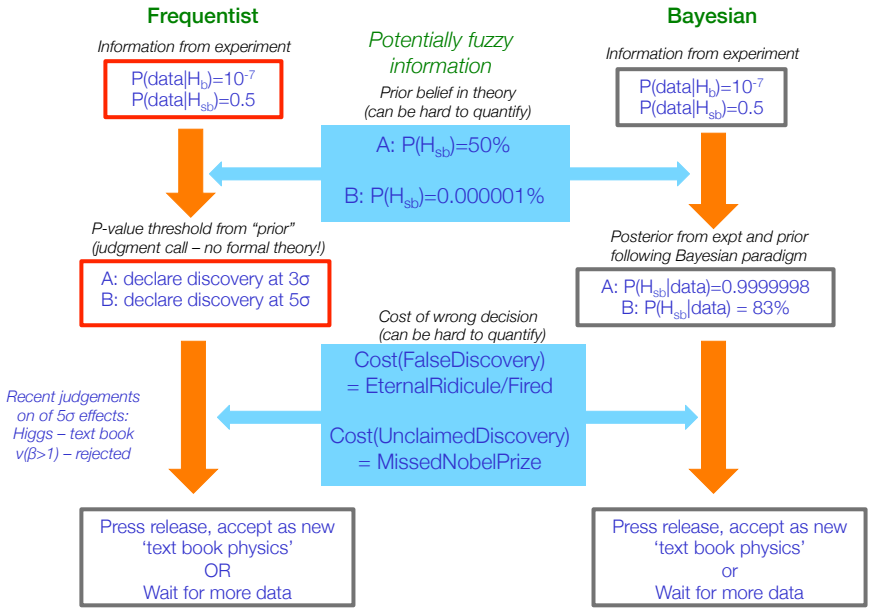
  Bayesian probability = (subjective) degree of belief — P(theo|data) P(data|theo)

  Frequentist probability = fraction of outcomes in future repeated identical experiments — P(data|theo)

"If you'd repeat this experiment identically many times, in a fraction P you will observe the same outcome"

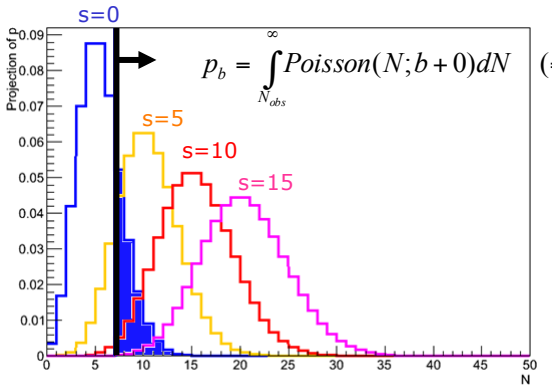
Wouter Verkerke, NIKHEF

How a theory becomes text-book physics



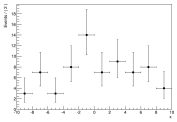
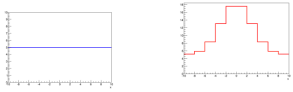
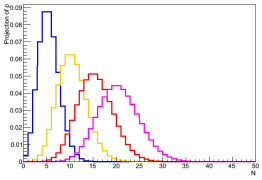
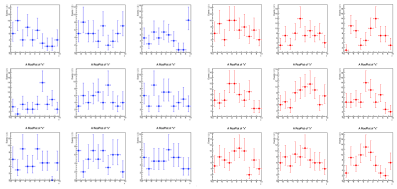
P-values for counting experiments

- Now make a measurement $N=N_{obs}$ (example $N_{obs}=7$)
- **Definition: p-value:**
  probability to obtain the observed data, or more extreme in future repeated identical experiments
  - Example: p-value for background-only hypothesis



Ordering distributions by 'signal-likeness' aka 'extremity'

- How to define 'extremity' if observed data is a distribution

	Counting	Histogram
Observation	$N_{obs}=7$	
Median expected by hypothesis	$N_{exp}(s=0) = 5$ $N_{exp}(s=5) = 10$	
Predicted distribution of observables		

Which histogram is more 'extreme'?

Likelihoods for distributions - summary

- **Bayesian inference unchanged**

  → simply insert L of distribution to calculate P(H|data)

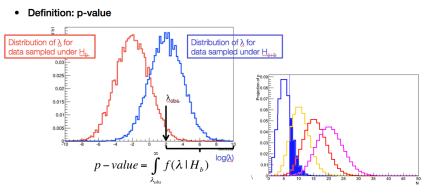
$$P(H_{s+b} \mid \vec{N}) = \frac{L(\vec{N} \mid H_{s+b})P(H_{s+b})}{L(\vec{N} \mid H_{s+b})P(H_{s+b}) + L(\vec{N} \mid H_b)P(H_b)}$$

- **Frequentist inference procedure *modified***

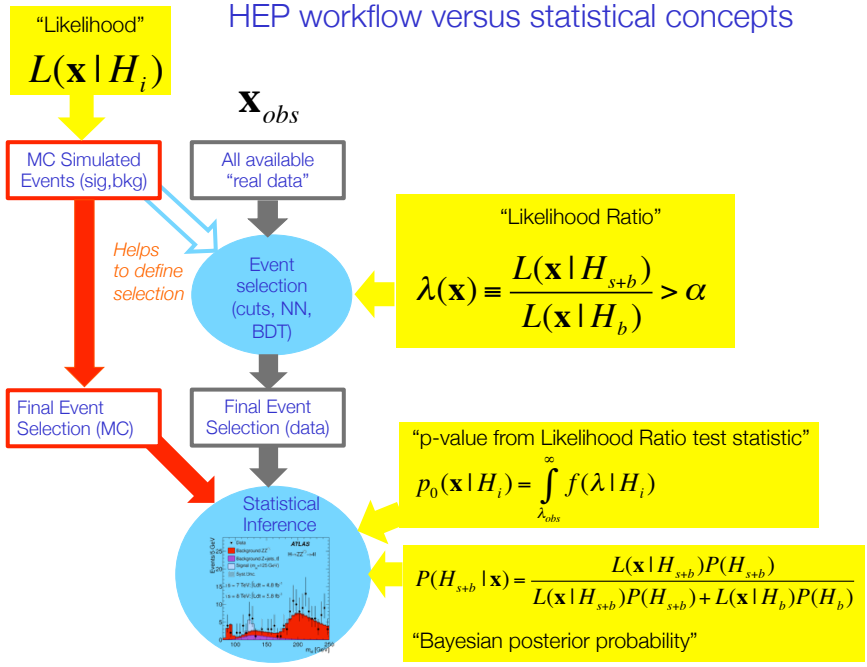
  → Pure P(data|hypo) not useful for non-counting data
  → Order all possible data with a (LR) test statistic in 'extremity'
  → Quote p(data|hypo) as 'p-value' for hypothesis
  Probability to obtain observed data, *or more extreme,* is X%

'Probability to obtain 13 or more 4-lepton events under the no-Higgs hypothesis is 10^{-7}'

'Probability to obtain 13 or more 4-lepton events under the SM Higgs hypothesis is 50%'



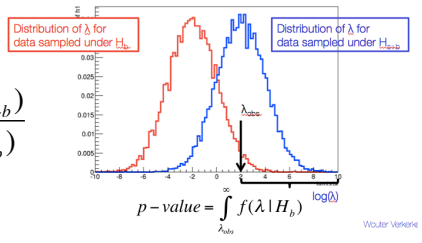
HEP workflow versus statistical concepts



The Likelihood Ratio test statistic as tool for event selection

- Note that hypothesis testing with two simple hypotheses for observable distributions, exactly describes 'event selection' problem
- In fact we have already 'solved' the optimal event selection problem! Given two hypothesis H_{s+b} and H_b that predict an complex multivariate distribution of observables, **you can always classify all events in terms of 'signal-likeness' (a.k.a 'extremity') with a likelihood ratio**

$$\lambda(\vec{x},\vec{y},\vec{z},...) = \frac{L(\vec{x},\vec{y},\vec{z},...\,|\,H_{s+b})}{L(\vec{x},\vec{y},\vec{z},...\,|\,H_b)}$$



- So far we have exploited λ to calculate a frequentist p-value **tomorrow now explore properties 'cut on λ' as basis of (optimal) event selection**

Wouter Verkerke, NIKHEF

Event selection

- The event selection problem:
  - Input: Two classes of events “signal” and “background”
  - Output: Two categories of events “selected” and “rejected”
- Goal: select as many signal events as possible,
  reject as many background events as possible
- Note that optimization goal as stated is ambiguous.
  - But can choose a well-defined by optimization goal by e.g. fixing desired background acceptance rate, and then choose procedure that has highest signal acceptance.
- Relates to “classical hypothesis testing”
  - Two competing hypothesis (traditionally named ‘null’ and ‘alternate’)
  - Here null = background, alternate = signal

Terminology of classical hypothesis testing

- Definition of terms
  - Rate of type-I error = α
  - Rate of type-II error = β
  - Power of test is 1-β

		Actual condition	
		Guilty	**Not guilty**
Decision	**Verdict of 'guilty'**	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

- Treat hypotheses asymmetrically
  - Null hypo is usually special → Fix rate of type-I error
  - Criminal convictions: Fix rate of unjust convictions
  - Higgs discovery: Fix rate of false discovery
  - Event selection: Fix rate of background that is accepted
- Now can define a well stated goal for optimal testing
  - Maximize the power of test (minimized rate of type-II error) for given α
  - Event selection: Maximize fraction of signal accepted

The Neyman-Pearson lemma

- In 1932-1938 Neyman and Pearson developed a theory in which one must consider competing hypotheses
  - Null hypothesis (H_0) = Background only
  - Alternate hypotheses (H_1) = e.g. Signal + Background

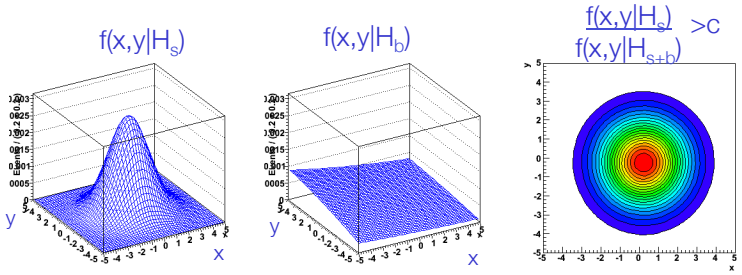
  and proved that
- The region W that minimizes the rate of the type-II error (not reporting true discovery) is a contour of the Likelihood Ratio

$$\frac{P(x|H_1)}{P(x|H_0)} > k_\alpha$$

- Any other region of the same size will have less power

The Neyman-Pearson lemma

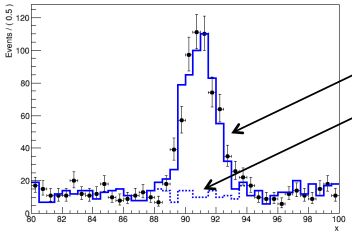
- Example of application of NP-lemma with two observables



- Cut-off value c controls type-I error rate ('size' = bkg rate)
  Neyman-Pearson: LR cut gives best possible 'power' = signal eff.
- So why don't we *always* do this? (instead of training neural networks, boosted decision trees etc)

Why Neyman-Pearson doesn't always help

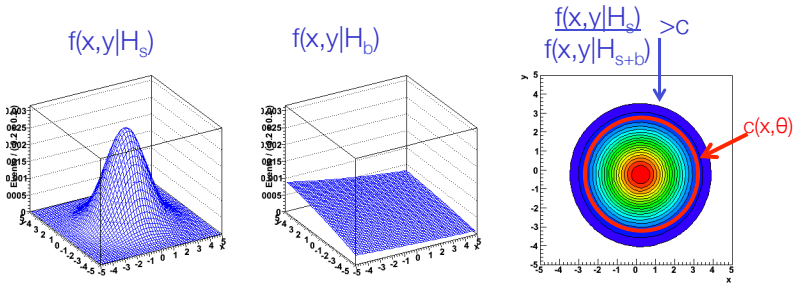
- The problem is that we usually don't have explicit formulae for the pdfs $f(\vec{x}|\mathrm{s})$, $f(\vec{x}|\mathrm{b})$.
- Instead we may have Monte Carlo samples for signal and background processes
  - Difficult to reconstruct analytical distributions of pdfs from MC samples, especially if number of dimensions is large
- If physics problem has only few observables can still estimate estimate pdfs with histograms or kernel estimation,
  - But in such cases one can also forego event selection and go straight to hypothesis testing / paramater estimation with all events



Hypothesis testing with a large number of observables

- When number of observables is large follow different strategy
- Instead of aiming at approximating p.d.f.s f(x|s) and f(x|b) aim to approximate decision boundary with an empirical parametric form

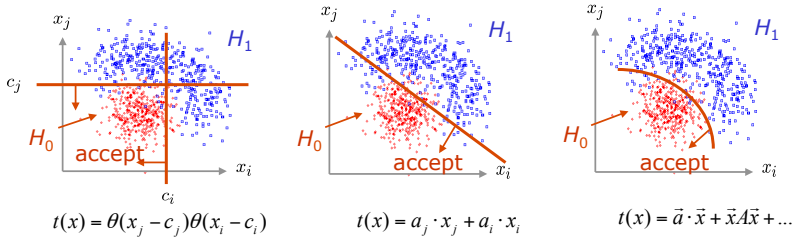
$$A_\alpha(\vec{x}) = \left[\frac{f(\vec{x}\,|\,s)}{f(\vec{x}\,|\,s+b)} > \alpha \right] \quad \Rightarrow \quad A_\alpha(\vec{x}) = c(\vec{x},\vec{\theta})$$



Empirical parametric forms of decision boundaries

- Can in principle choose any type of Ansatz parametric shape

Rectangular cut | *Linear cut* | *Non-linear cut*

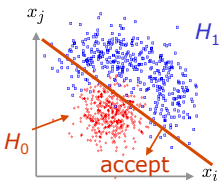


- Goal of Ansatz form is estimate of a 'signal probability' for every event in the observable space x (just like the LR)
- Choice of desired type-I error rate (selected background rate), can be set later by choosing appropriate cut on Ansatz test statistic.

The simplest Ansatz – A linear disciminant

- A linear discriminant constructs t(x) from a linear combination of the variables x_i

$$t(\vec{x}) = \sum_{i=1}^{N} a_i x_i = \vec{a} \cdot \vec{x}$$



  - A cut on t(x) results in a linear decision plane in x-space

- What is optimal choice of direction vector a?
- Solution provided by the Fisher – The Fisher discriminant

$$F(\vec{x}) = \underbrace{\left(\vec{\mu}_S - \vec{\mu}_B\right)^T V^{-1}}_{\vec{a}} \vec{x}$$

Mean values in x_i for sig,bkg

Inverse of variance matrix of signal/background (assumed to be the same)

R.A. Fisher
Ann. Eugen. 7(1936) 179.

The simplest Ansatz – A linear disciminant

- Operation advantage of Fisher discrimant is that test statistic parameters can be *calculated* (no iterative estimation is required)

$$F(\vec{x}) = \underbrace{\left(\vec{\mu}_S - \vec{\mu}_B\right)^T V^{-1}}_{\vec{a}} \vec{x}$$

Mean values in x_i for sig,bkg

Inverse of variance matrix of signal/background (assumed to be the same)

R.A. Fisher
Ann. Eugen. 7(1936) 179.

- Fisher discriminant is optimal test statistic (i.e. maps to Neyman Pearson Likelihood Ratio) for case where both hypotheses are multivariate Gaussian distributions with the same variance, but diffferent means

$$f(x \mid s) = Gauss(\vec{x} - \vec{\mu}_s, V)$$
$$f(x \mid b) = Gauss(\vec{x} - \vec{\mu}_b, V)$$

Multivariate Gaussian distributions with **different means** but **same width** for signal and background

The simplest Ansatz – A linear disciminant

- How the Fisher discriminant follows from the LR test statistic

$$-\log\left(\frac{f(x \mid s)}{f(x \mid b)}\right) = 0.5\left(\frac{x-\mu_s}{\sigma^2}\right)^2 - 0.5\left(\frac{x-\mu_b}{\sigma^2}\right)^2 + C$$
$$= 0.5\frac{x^2 - 2x\mu_s + \mu_s^2 - x^2 + 2x\mu_b - \mu_b^2}{\sigma^2} + C$$
$$= \frac{x(\mu_s - \mu_b)}{\sigma^2} + C'$$

- Generalization for multidimensional Gaussian distributions

$$\log\lambda(x) = \frac{x(\mu_s - \mu_b)}{\sigma^2} + C' \xrightarrow{\sigma^2 \to V} \lambda(x) = \vec{x}(\vec{\mu}_s - \vec{\mu}_b)V^{-1} + C'$$

- Note that since we took -log of λ, F(x) is not signal probability, but we can trivially recover this

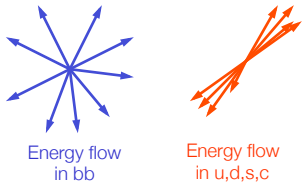
$$P_s(F) = \frac{1}{1+e^{-F}}$$

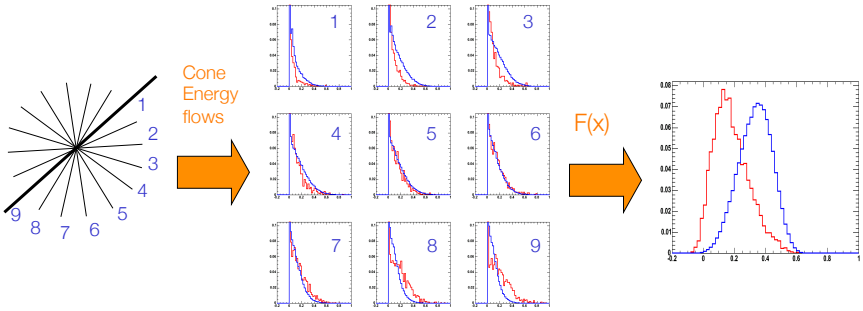
"Logistic sigmoid function"

If λ=1, x is equally likely under s,b
Then F = -log(λ)=0 → P = 50%

Example of Fisher discriminant use in HEP

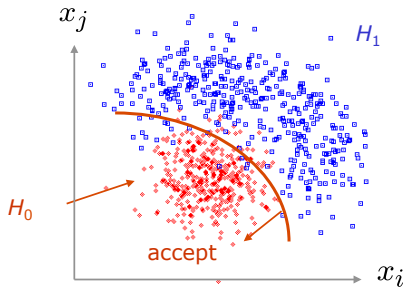
- The "CLEO" Fisher discriminant
  - Goal: distinguish between e+e- → Y4s → $b\bar{b}$ and $u\bar{u}$,$d\bar{d}$,$s\bar{s}$,$c\bar{c}$
  - Method: Measure energy flow in 9 concentric cones around direction of B candidate





Non-linear test statistics

- In most real-life HEP applications signal and background are not multi-variate Gaussian distributions with different means
- Will need more complex Ansatz shapes than Fisher discriminant
- Loose ability analytically calculate parameters of Ansatz model from Likelihood Ratio test statistic (as was done for Fisher)
- Choose an Ansatz shapes with tunable parameters
  - Artificial Neural Networks
  - Decision Trees
  - Support Vector Machines
  - Rule Ensembles
- Need numeric procedure to estimate Ansatz parameters → Machine learning or Bayesian Learning



Machine Learning – General Principles

- Given a Ansatz parametric test statistic T(x|θ), quantify 'risk' due 'loss of performance' due to misclassifications by T as follows

Loss function (~ log of Gaussian Likelihood)

$$R(\theta) = \int \left(T(\vec{x}\,|\,\theta) - 0\right)^2 f(\vec{x}\,|\,b)\,d\vec{x} \quad + \quad \int \left(T(\vec{x}\,|\,\theta) - 1\right)^2 f(\vec{x}\,|\,s)\,d\vec{x}$$

Risk function

Target value of T for background classification

Target value of T for signal classification

- Practical issue: *since f(x|s,b) not analytically available, cannot evaluate risk function*. Solution → Substitute risk with 'empirical risk' which substitutes integral with Monte Carlo approximation

$$E(\theta) = \frac{1}{N_b} \sum_{D(x|b)} \left(T(\vec{x}_i\,|\,\theta) - 0\right)^2 \quad + \quad \frac{1}{N_s} \sum_{D(x|s)} \left(T(\vec{x}_i\,|\,\theta) - 1\right)^2$$

Empirical Risk function

x_i is a set of points sampled from f(x|b)

x_i is a set of points sampled from f(x|s)

Machine Learning – General Principles

- Minimization of empirical risk E(θ) can be performed with numerical methods (many tools are available, e.g. TMVA)
- But approximation of empirical risk w.r.t analytical risk introduces possibility for 'overtraining':

  If MC samples for signal and background are small, and number of parameters θ, one can always reduce empirical risk to zero ('perfect selection')

  (Conceptually similar to χ^2 fit : if you fit a 10^{th} order polynomial to 10 points – you will always perfectly describe the data. You will however not perfectly describe an independent dataset sampled from the same parent distribution)
- Even if empirical risk is not reduced to zero by training, it may still be smaller than true risk → Control effect by evaluating empirical risk also on independent validation sample during minimization. If ER on samples start to diverge, stop minimization

Bayesian Learning – General principles

- Can also applied Bayesian methodology to learning process of decision boundaries
- Given a dataset D(x,y) and a Ansatz model with parameters w, aim is to estimate parameters w

P(w) = posterior density on parameters of discriminant

Likelihood of the data under hypothesis w

$$P(w \mid \vec{x}, y) = \frac{L(\vec{x}, y \mid w) P(w)}{P(\vec{x}, y)}$$

$$= \frac{L(y \mid w, \vec{x}) L(x \mid w) P(w)}{\int L(y \mid w, \vec{x}) dw L(\vec{x})}$$ ← L(a,b)=L(a|b)L(b)

$$= \frac{L(y \mid w, \vec{x}) P(w)}{\int L(y \mid w, \vec{x}) dw L(\vec{x})}$$ ← L(x|w)=1 since input observables independent of model

Training data
x: inputs
y: class label
(S/B) typically

Bayesian Learning – General principles

- Inserting a binomial likelihood function to model classification the classification problem

$$L(y \mid x, w) = \prod_i T(x_i, w)^y \left[1 - T(x_i, w)\right]^{1-y}$$

- The parameters w are thus estimated from the Bayesian posteriors densities

$$P(w \mid \vec{x}, y) = \frac{L(y \mid w, \vec{x}) P(w)}{\int L(y \mid w, \vec{x}) dw L(\vec{x})}$$

  - No iterative minimization, but Note that integrals over 'w-space' can usually only be performed numerically and if w contains many parameters, this is computationally challenging

- If class of function T(x,w) is large enough it will contain a function T(x,w*) that represents the true minimum in E(w)
  - I.e. T(x,w*) is the Bayesian equivalent of of Frequentist TS that is NP L ratio
  - In that case the test statistic is

$$T(x, w^*) = \int y L(y \mid x) dy$$

$$= L(y = 1 \mid x) = \frac{L(x \mid y = 1) P(y = 1)}{L(x \mid y = 0) P(y = 0) + L(x \mid y = 1) P(y = 1)}$$

$$L(y \mid x, w) = \prod_i T(x_i, w)^y \left[1 - T(x_i, w)\right]^{1-y}$$

With y=0,1 only

Machine/Bayesian learning – Non-linear Ansatz functions

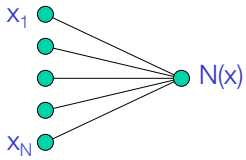
- Artificial Neural Network is one of the most popular non-linear ansatz forms. In it simplest incarnation the classifier function is

$$N(\vec{x}) = s\left(a_0 + \sum_i a_i x_i\right)$$

s(t) is the activation function, usually a logistic sigmoid

$$s(t) = \frac{1}{1+e^{-t}}$$

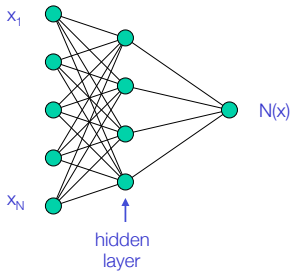
- This formula corresponds to the 'single layer perceptron'
  - Visualization of single layer network topology



Since the activation function s(t) is monotonic,
a single layer N(x) is equivalent
to the Fisher discriminant F(x)

Neural networks – general structure

- The single layer model and easily be generalized to a ***multilayer*** perceptron



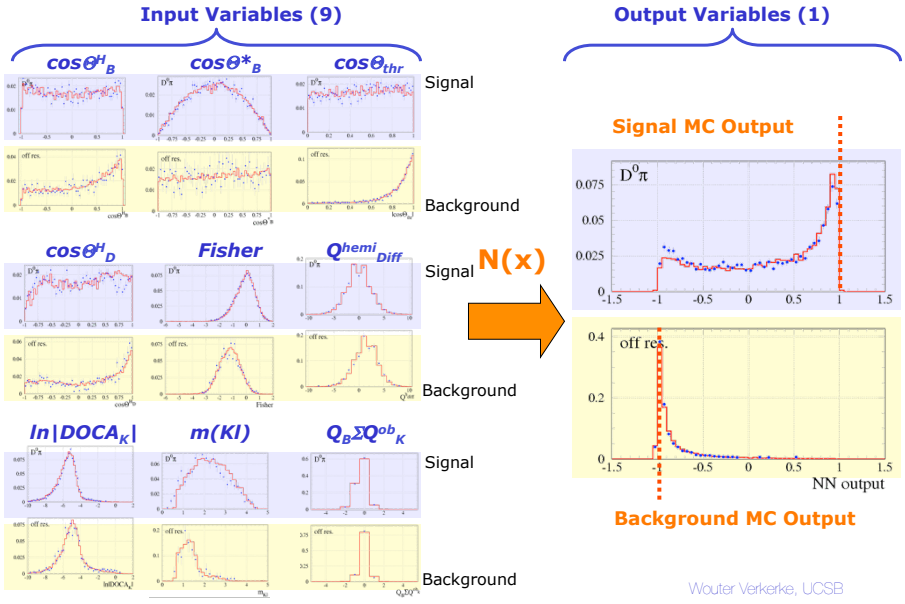
$$N(\vec{x}) = s(a_0 + \sum_{i=1,}^{m} a_i h_i(\vec{x}))$$

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

with a_i and w_{ij} weights (connection strengths)

- Easy to generalize to arbitrary number of layers
- Feed-forward net: values of a node depend only on earlier layers (usually only on preceding layer) 'the network architecture'
- More nodes bring N(x) allow it to be closer to optimal (Neyman Pearson / Bayesian posterior) but with much more parameters to be determined

Neural networks – training example



Practical aspects of machine learning

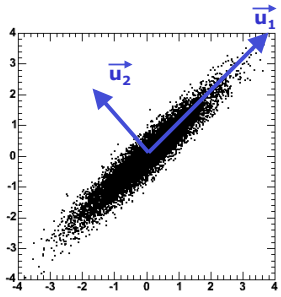
- Choose input variables sensibly
  - Don't include badly understood observables (such as #tracks/evt), variables that are not expected carry useful information
  - Generally: "Garbage in = Garbage out"
- Traditional Machine learning provides no guidance of useful complexity of test statistic (e.g. NN topology, layers)
  - Usually better to start simple and gradually increase complexity and see how that pays off
- Bayesian learning can (in principle) provide guidance on model complexity through Bayesian model selection
  - Bayes factors automatically includes a penalty for including too much model structure.

$$K = \frac{P(D \mid H_1)}{P(D \mid H_2)} = \frac{\int L(D \mid \theta_1, H_1) P(\theta_2 \mid H_1) d\theta_2}{\int L(D \mid \theta_2, H_2) P(\theta_2 \mid H_2) d\theta_2}$$

  - But availability of Bayesian model selection depends in practice on the software that you use.

Practical aspects of machine learning

- Don't make the learning problem unnecessarily difficult for the machine
- E.g. remove strong correlation with **explicit decorrelation** before learning step
  - Can use Principle Component Analysis
  - Or Cholesky decomposition (rotate with square-root of covariance matrix)



- Also: remember that for 2-class problem (sig/bkg) that each have multivariate Gaussian distributions with different means, the optimal discriminant is known analytically
  - Fisher discriminant is analytical solution. NN solution reduces to single-layer perceptron
- Thus, you can help your machine by transforming your inputs in a form **as close as possible to the Gaussian form** by transforming your input observables

Gaussianization of input observables

- You can transform *any* distribution in a Gaussian distribution in two steps
- 1 – Probability integral transform

"...seems likely to be one of the most fruitful conceptions introduced into statistical theory in the last few years" –Egon Pearson (1938)

$$y(x) = \int_{-\infty}^{x} f(x' \mid H)\,dx'$$

turns any distribution f(x) into a flat distribution in y(x)

- 2 – Inverse error function

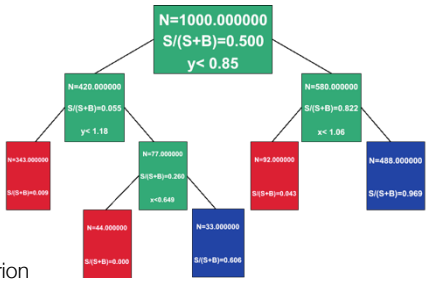
$$x^{\text{Gauss}} = \sqrt{2}\cdot \text{erf}^{-1}\left(2x^{\text{flat}} - 1\right) \qquad \text{erf}\left(x\right) = \frac{2}{\sqrt{\pi}}\int_{0}^{x} e^{-t^2}\,dt$$

turns flat distribution into a Gaussian distribution

- Note that you can make either signal or background Gaussian, but usually not *both*

A very different type of Ansatz - Decision Trees

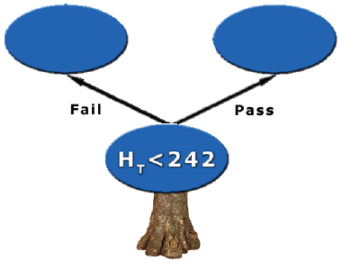
- A Decision Tree encodes sequential rectangular cuts
  - But with a lot of underlying theory on training and optimization
  - Machine-learning technique, widely used in social sciences
  - L. Breiman et al., "Classification and Regression Trees" (1984)



- Basic principle
  - Extend cut-based selection
  - Try not to rule out events failing a particular criterion
  - Keep events rejected by one criterion and see whether other criteria could help classify them properly

Building a tree – splitting the data

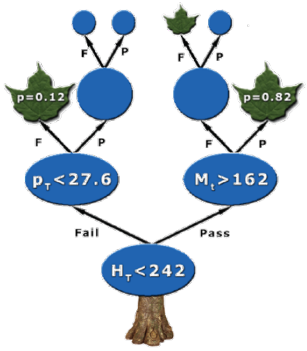
- Essential operation :
  splitting the data in 2 groups using a single cut, e.g. $H_T<242$



- Goal: find 'best cut' as quantified through best separation of signal and background (requires some metric to quantify this)
- Procedure:
  1) Find cut value with best separation for *each* observable
  2) Apply **only** cut on observable that results in best separation

Building a tree – recursive splitting

- Repeat splitting procedure on sub-samples of previous split



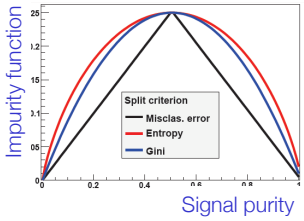
- **Output** of decision tree:
  - 'signal' or 'background' (0/1) or
  - probability based on *expected purity* of leaf (s/s+b)

Parameters in the construction of a decision tree

- Normalization of signal and background before training
  - Usually *same total weight* for signal and background events
- In the selection of splits
  - list of questions ($var_i < cut_i$) to consider
  - Separation metric (quantifies how good the split is)
- Decision to stop splitting (declare a node terminal)
  - Minimum leaf size (e.g. 100 events)
  - Insufficient improvement from splitting
  - Perfect classification (all events in leaf belong to same class)
- Assignment of terminal node to a class
  - Usually: purity>0.5 = signal, purity<0.5 = background

Machine learning with Decision Trees

- Instead of '(Empirical) Risk' **minimize 'Impurity Function'** of leaves
  - Impurity function *i(t)* quantifies (im)purity of a sample, but is not uniquely defined
  - Simplest option: i(t) = misclassification rate



- For a proposed split *s* on a node *t*, decrease of impurity is

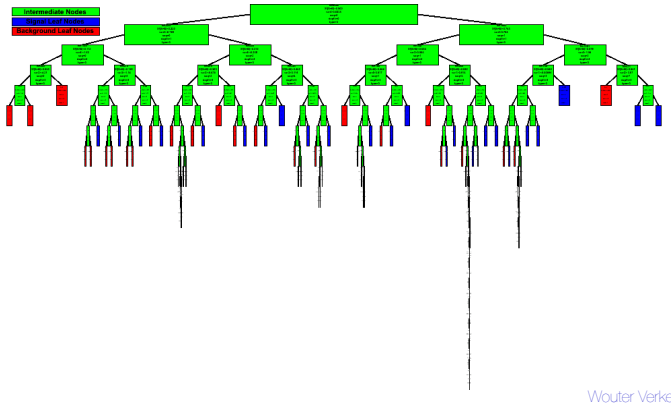
$$\Delta i(s,t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R)$$

Impurity of sample before split — Impurity of 'left' sample — Impurity of 'right' sample

- Take split that results in largest Δi

Machine learning with Decision Trees

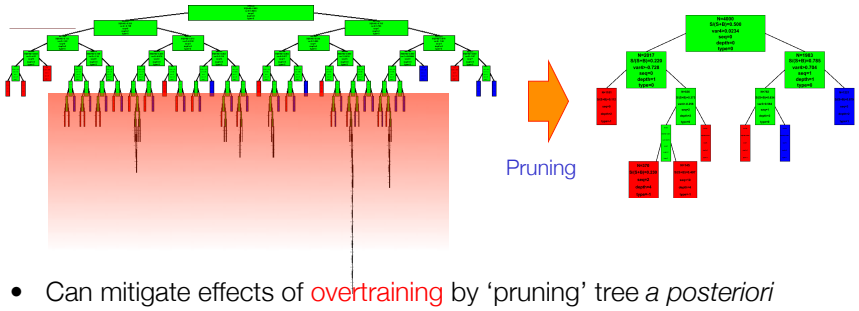
- Stop splitting when
  - not enough improvement (introduce a cutoff Δi)
  - not enough statistics in sample, or node is pure (signal or background)
- Example decision tree from learning process



Wouter Verkerke, NIKHEF

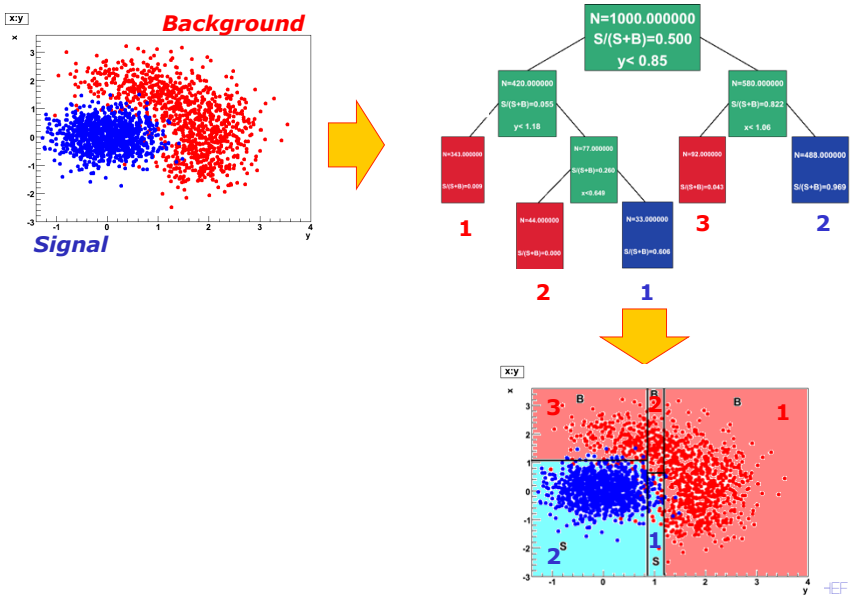
Machine learning with Decision Trees

- Given that analytical pdfs f(x|s) and f(x|b) are usually not available, splitting decisions are based on 'empirical impurity' rather than true 'impurity' → risk of overtraining exists



- Can mitigate effects of overtraining by 'pruning' tree *a posteriori*
  - **Expected error pruning** (prune weak splits that are consistent with original leaf within statistical error of training sample)
  - **Cost/Complexity pruning** (generally strategy to trade tree complexity against performance)

Concrete example of a trained Decision Tree



Boosted Decision trees

- Decision trees largely used with 'boosting strategy'
- **Boosting** = strategy to combine multiple weaker classifiers into a single strong classifier
- First provable boosting algorithm by Shapire (1990)
  - Train classifier *T1* on N events
  - Train *T2* on new N-sample, half of which misclassified by *T1*
  - Build *T3* on events where *T1* and *T2* disagree
  - Boosted classifier: MajorityVote(*T1*,*T2*,*T3*)
- Most used: AdaBoost = Adaptive Boosting (Freund & Shapire '96)
  - Learning procedure adjusts to training data to classify it better
  - Many variations on the same theme for actual implementation

AdaBoost

- Schematic view of *iterative* algorithm
  - **Train Decision Tree** on (weighted) signal and background training samples
  - **Calculate misclassification** rate for Tree K (initial tree has k=1)

$$\epsilon_k = \frac{\sum_{i=1}^{N} w_i^k \times \text{isMisclassified}_k(i)}{\sum_{i=1}^{N} w_i^k}$$

"Weighted average of **isMisclassified** over all training events"

  - **Calculate weight of tree K** in 'forest decision' $\alpha_k = \beta \times \ln((1-\epsilon_k)/\epsilon_k)$
  - **Increase weight of misclassified events** in Sample(k) to create Sample(k+1)

$$w_i^k \rightarrow w_i^{k+1} = w_i^k \times e^{\alpha_k}$$

- Boosted classifier is result is performance-weighted 'forest'

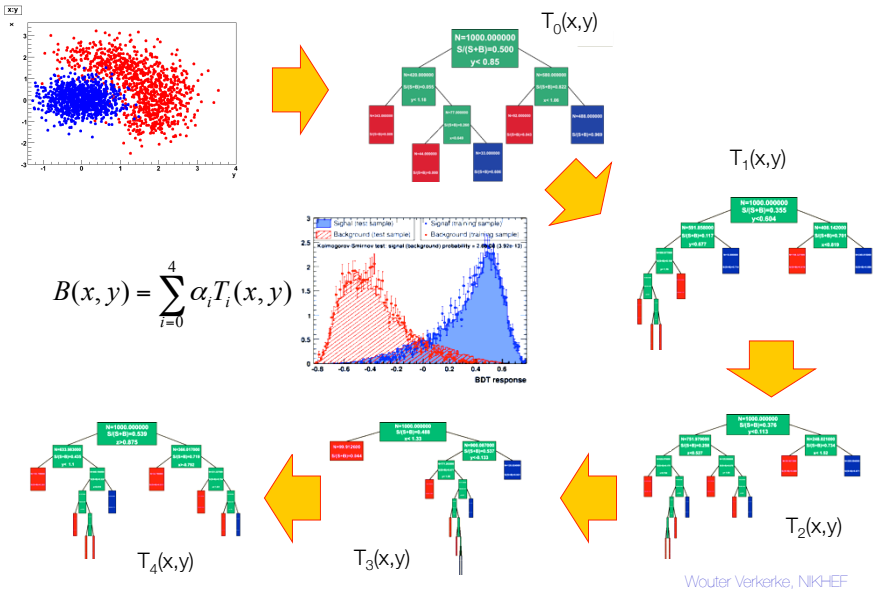
$$T(i) = \sum_{k=1}^{N_{\text{tree}}} \alpha_k T_k(i)$$

"Weighted average of **Trees** by their performance"

AdaBoost by example

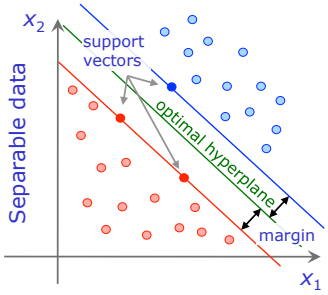
- So-so classifier (Error rate = 40%) $\alpha = \ln \frac{1-0.4}{0.4} = 0.4$
  - Misclassified events get their weight multiplied by exp(0.4)=1.5
  - Next tree will have to work a bit harder on these events
- Good classifier (Error rate = 5%) $\alpha = \ln \frac{1-0.05}{0.05} = 2.9$
  - Misclassified events get their weight multiplied by exp(2.9)=19 (!!)
  - Being failed by a good classifier means a big penalty: must be a difficult case
  - Next tree will have to pay much more attention to this event and try to get it right
- Note that boosting usually results in (strong) overtraining
  - Since with misclassification rate will ultimately go to zero

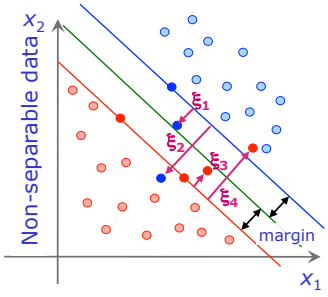
Example of Boosting



Support Vector Machines

- Find hyperplane that best separates signal from background
  - Best separation: maximum distance (margin) between closest events (*support*) to hyperplane
  - Linear decision boundary is defined by solution of a Langrangian
  - *Solution of Lagrangian only depends on inner product of support vectors*
- For non-separable data add misclassification cost
  - add *misclassification cost* parameter $C \cdot \Sigma_i \xi_i$ to minimization function





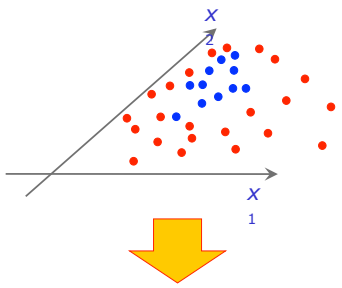
Wouter Verkerke, NIKHEF

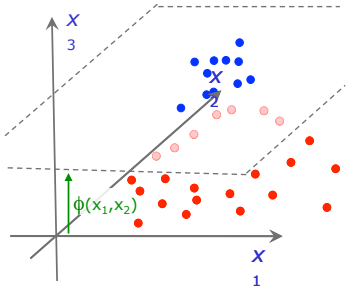
Support Vector Machines

- Non-linear cases
  - Transform variables into higher dimensional feature space

    (x,y) → (x,y,z=ϕ(x,y))

    where again a linear boundary (hyperplane) can separate the data
  - Explicit basis functions not required: use *Kernel Functions* to approximate scalar products between transformed vectors in the higher dimensional feature space
  - Choose Kernel and use the hyperplane using the linear techniques developed above



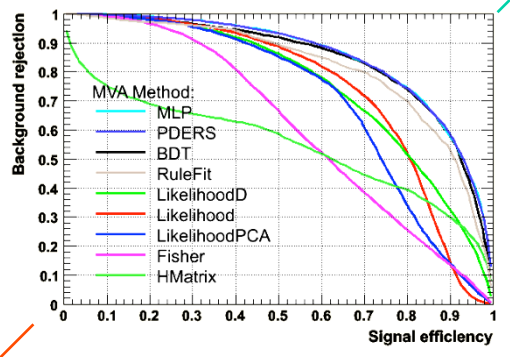


Wouter Verkerke, NIKHEF

Characterizing and comparing performance

- Performance of a test statistic characterized by ε(sig) vs ε(bkg) curve
  - Curve for theoretical maximum performance can be added if true S(x) and B(x) are known
  - Position on curve determines tradeoff between type-I and type-II errors

Good Performance



Bad Performance

What is *T*MVA

- ROOT: is the analysis framework used by most (HEP)-physicists
- Idea: rather than just implementing new MVA techniques and making them available in ROOT (*i.e.*, like TMultiLayerPercetron does):
  - Have one common platform / interface for all MVA classifiers
  - Have common data pre-processing capabilities
  - Train and test all classifiers on same data sample and evaluate consistently
  - Provide common analysis (ROOT scripts) and application framework
  - Provide access with and without ROOT, through macros, C++ executables or python

A. Hoecker: Multivariate Analysis with TMVA

Limitations of *T*MVA

- Development started beginning of 2006 – a mature but **not** a final package
- Known limitations / missing features
  - Performs classification only, and only in binary mode: *signal* versus *background*
  - Supervised learning only (no unsupervised "bump hunting")
  - Relatively stiff design – not easy to mix methods, not easy to setup categories
  - Cross-validation not yet generalised for use by all classifiers
  - Optimisation of classifier architectures still requires tuning "by hand"
- Work ongoing in most of these areas → see later in this talk

*T*MVA Content

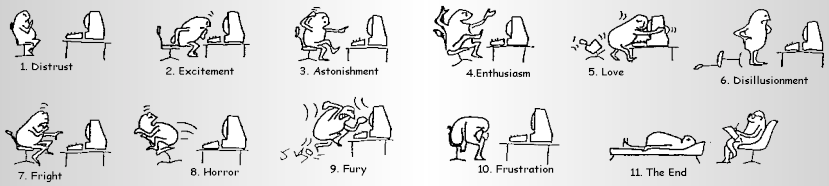
- Currently implemented classifiers
  - Rectangular cut optimisation
  - Projective and multidimensional likelihood estimator
  - k-Nearest Neighbor algorithm
  - Fisher and H-Matrix discriminants
  - Function discriminant
  - Artificial neural networks (3 *multilayer perceptron* impls)
  - Boosted/bagged decision trees
  - RuleFit
  - Support Vector Machine
- Currently implemented data preprocessing stages:
  - Decorrelation
  - Principal Value Decomposition
  - Transformation to uniform and Gaussian distributions

Using TMVA

A typical TMVA analysis consists of two main steps:

1. *Training phase*: training, testing and evaluation of classifiers using data samples with known signal and background composition
2. *Application phase*: using selected trained classifiers to classify unknown data samples

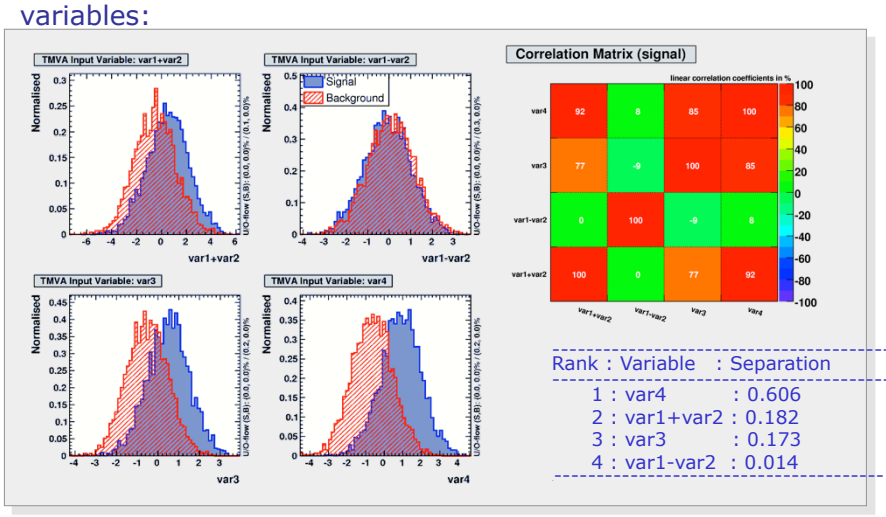
➧ Illustration of these steps with toy data samples



→ TMVA tutorial

A Toy Example (idealized)

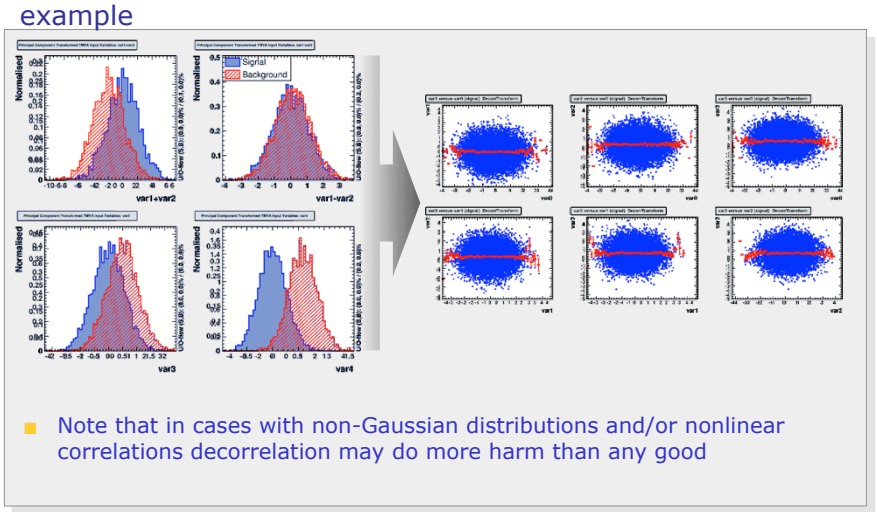
- Use data set with 4 linearly correlated Gaussian distributed variables:



Rank	Variable	Separation
1	var4	0.606
2	var1+var2	0.182
3	var3	0.173
4	var1-var2	0.014

Preprocessing the Input Variables

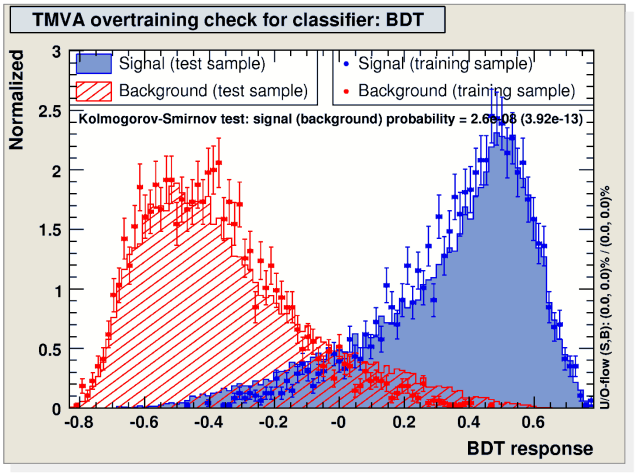
- Decorrelation of variables before training is useful for *this* example



  - Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

Evaluating the Classifier Training (II)

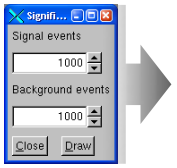
- Check for overtraining: classifier output for test *and* training samples …

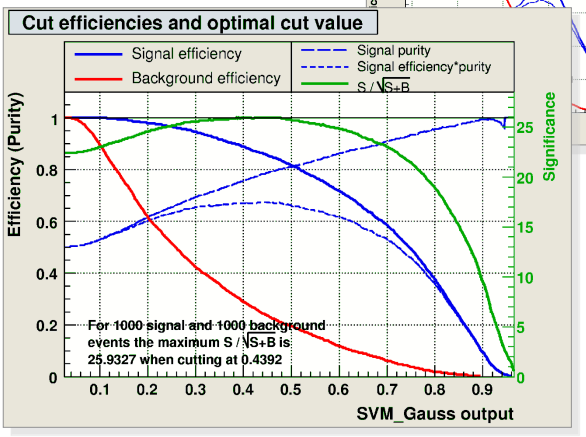


Evaluating the Classifier Training (V)

- Optimal cut for each classifiers …

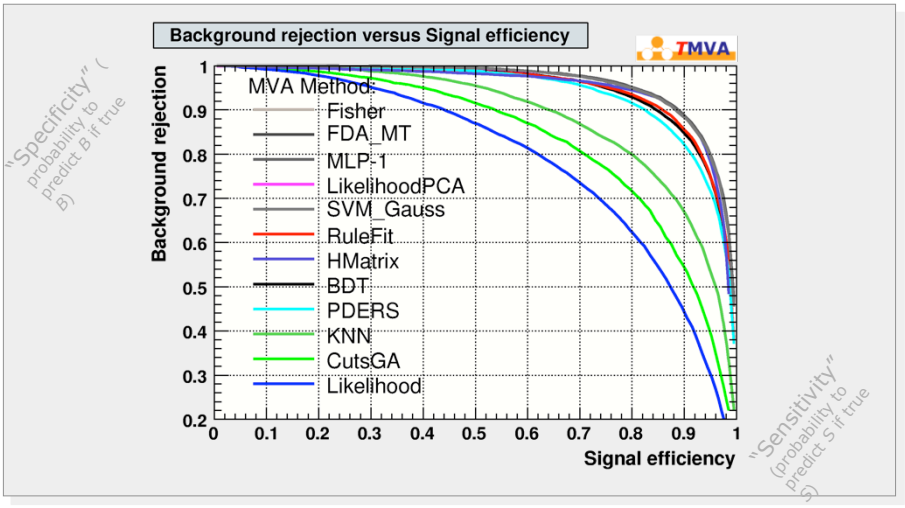
  Determine the optimal cut (working point) on a classifier output





Receiver Operating Characteristics (ROC) Curve

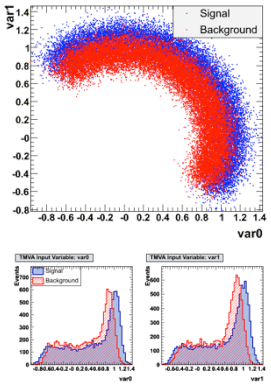
- Smooth background rejection versus signal efficiency curve: (from cut on classifier output)

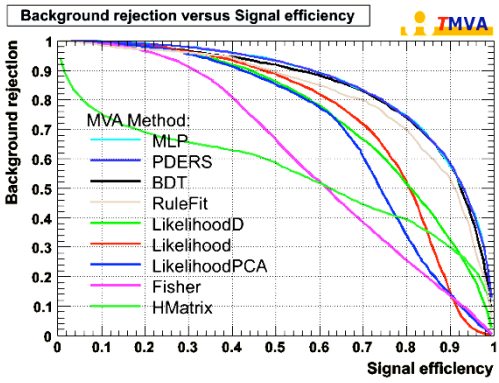


Example: Circular Correlation

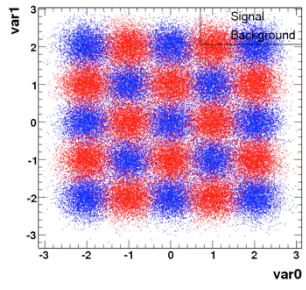
- Illustrate the behavior of linear and nonlinear classifiers

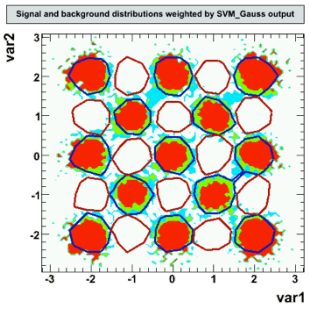
Circular correlations
(same for signal and background)



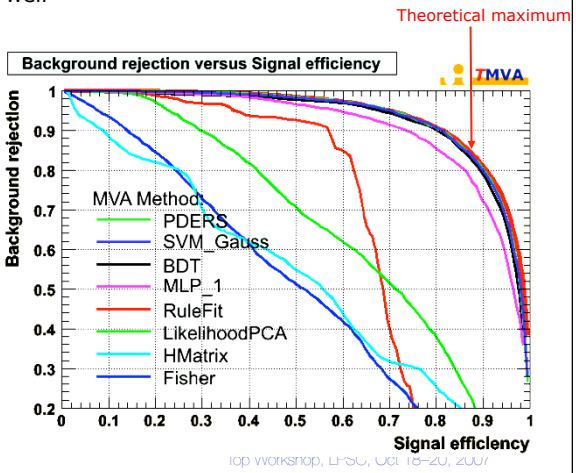


The "Schachbrett" Toy





- Performance achieved without parameter tuning: PDERS and BDT best "out of the box" classifiers
- After specific tuning, also SVM und MLP perform well

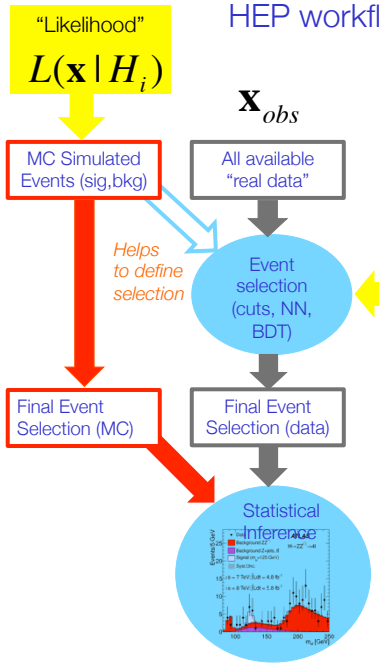


Summary of the Classifiers and their Properties

Criteria		Cuts	Likeli-hood	PDERS / k-NN	H-Matrix	Fisher	MLP	BDT	RuleFit	SVM
Perfor-mance	no / linear correlations	😐	🙂	🙂	😐	🙂	🙂	😐	🙂	🙂
	nonlinear correlations	😐	☹	🙂	☹	☹	🙂	🙂	😐	🙂
Speed	Training	☹	🙂	🙂	🙂	🙂	😐	☹	😐	☹
	Response	🙂	🙂	☹/😐	🙂	🙂	🙂	😐	😐	😐
Robust -ness	Overtraining	🙂	😐	😐	🙂	🙂	☹	☹	😐	😐
	Weak input variables	🙂	🙂	☹	🙂	🙂	😐	😐	😐	😐
Curse of dimensionality		☹	🙂	☹	🙂	🙂	😐	🙂	😐	😐
Transparency		🙂	🙂	😐	🙂	🙂	☹	☹	☹	☹

The properties of the Function discriminant (FDA) depend on the chosen function

HEP workflow versus statistical concepts



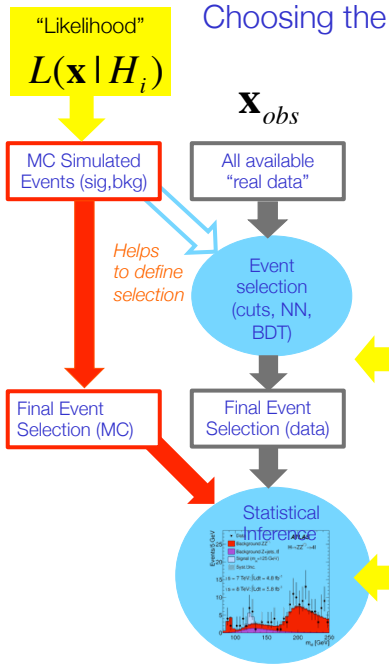
"Likelihood Ratio"

$$\lambda(\mathbf{x}) \equiv \frac{L(\mathbf{x} \mid H_{s+b})}{L(\mathbf{x} \mid H_b)} > \alpha$$

Or **approximation of optimal test statistic** with a parametric form from **machine learning**

Remaining question:
What value of α represents 'optimal cut'?

Choosing the optimal cut on the test statistic



Note that in the limit of an optimal test statistic, and when subsequent using LR hypothesis test, *the cut on a has no influence on the statistical inference!*

→ *Purely operational decision (ntuple-sizes etc…)*

"Likelihood Ratio"

$$\lambda(\mathbf{x}) \equiv \frac{L(\mathbf{x}\,|\,H_{s+b})}{L(\mathbf{x}\,|\,H_b)} > \alpha$$

"p-value from Likelihood Ratio test statistic"

$$p_0(\mathbf{x}\,|\,H_i) = \int_{\lambda_{obs}}^{\infty} f(\lambda\,|\,H_i)$$

Choosing the optimal cut on the test statistic

- But reality is usually more complex:
  - Test statistics are usually not optimal,
  - Ingredients to test statistics, i.e. the event selection, are usually not perfectly known (systematic uncertainties)
- *In the subsequent statistical test phase we can account for (systematic) uncertainties in signal and background models in a detailed way*. In the event selection phase we cannot
  - Pragmatically considerations in design of event selection criteria: Ability to estimate level of background from the selected data & Small sensitivity of signal acceptance to selection criteria used
- Result is that Likelihood Ratio used for event selection and final hypothesis test are different ($\lambda_{selection} \neq \lambda_{hypotest}$) ➔ Cut on $\lambda_{selection}$ *will* influence statistical test with $\lambda_{hypotest}$
- To be able decide on optimal cut on $\lambda_{selection}$ you need a figure merit that approximates behavior of statistical test using $\lambda_{hypotest}$

Wouter Verkerke, NIKHEF

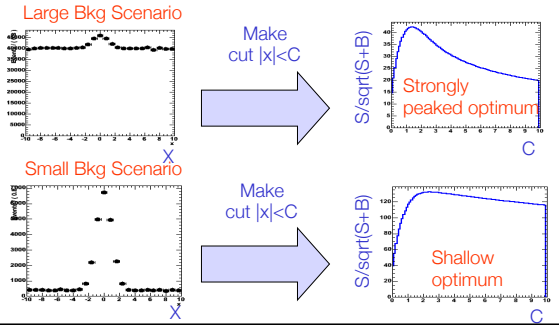
Traditional approximate Figures of Merit

- Traditional choices for Figure of Merit

Note that position of optimum depends on a priori knowledge of signal cross section

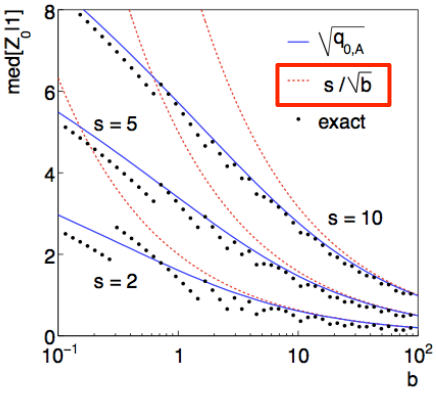
$$F(\alpha) = \frac{S(\alpha)}{\sqrt{B(\alpha)}} \quad \text{'discovery'} \qquad F(\alpha) = \frac{S(\alpha)}{\sqrt{S(\alpha)+B(\alpha)}} \quad \text{'measurement'}$$

  - Choice of FOM for cut optimization requires assumption on subsequent statistical analysis strategy. These traditional FOMs quantify signal significance for a counting experiment with an known level of expected background, and not e.g. 'strongest upper limit', no accounting for systematic uncertainties



Validity of approximations in Figures of Merit

- Note that approximations made in 'traditional' figure of merit are not always good (even for a counting experiment!)
- E.g. for 'discovery FOM' s/√b
  illustration of approximation for s=2,5,10 and b in range [0.01-100]
  shows significant deviations of s/√b from actual significance at low b



Improved discovery F.O.M ("Asimov Z") suggested for situations where s<<b is not true

$$\sqrt{q_{0,A}} = \sqrt{2\left((s+b)\ln(1+s/b) - s\right)} \,.$$
$$= \frac{s}{\sqrt{b}}\left(1 + \mathcal{O}(s/b)\right) \,.$$

Final comments on event selection

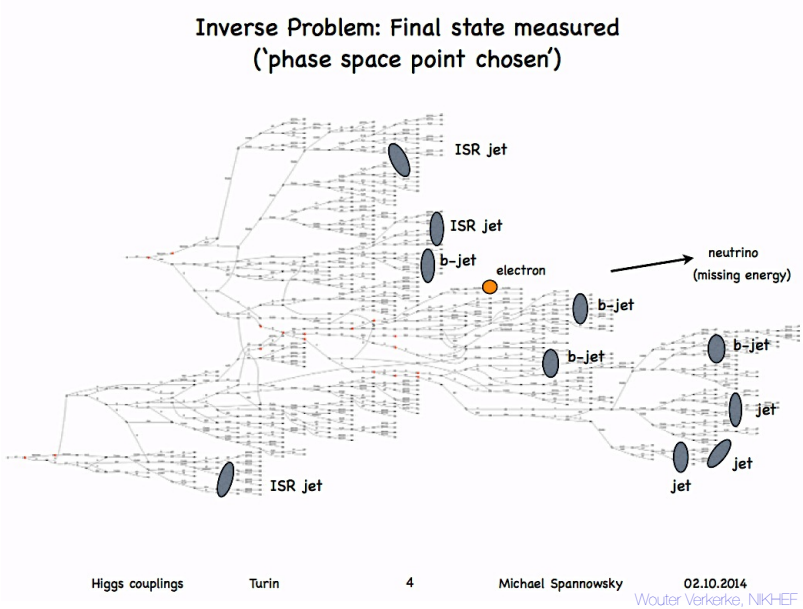
- Main issue with event selection is usually, sensitivity of selection criteria to systematic uncertainties
- What you'd like to avoid is your BDT/NN that is trained to get a small statistical uncertainty has a large sensitivity to a systematic uncertainties
- No easy way to incorporate effect of systematic uncertainties in training process

  → Can insert some knowledge of systematic uncertainties included in figure of merit when deciding where to cut in BDT/NN, but proper calculation usually requires much more information that signal and background event counts and is time consuming
- Use your physics intuition…

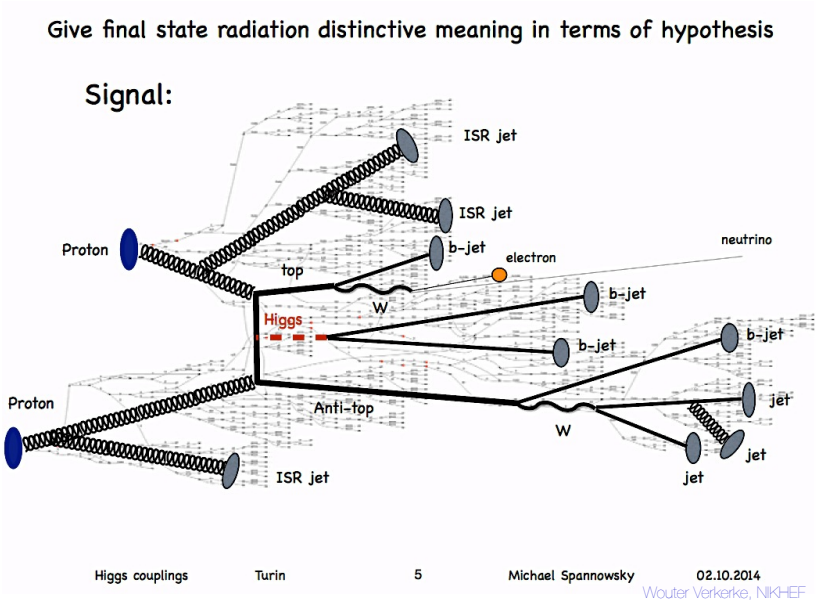
Alternatives to Machine Learning, back to NP optimal discrimant…

- Machine learning or Bayesian learning approach doesn't use detailed physics knowledge of signal and background processes that is available inside simulation to achieve separation
  - Only through final distribution S(x) and B(x) that is implicitly provided through MC simulation samples
- Another approach is to exploit full information of physics simulation process better to construct S(x) and B(x) and construct (optimal) NP discriminant from these → **Matrix Element Methods**
- Idea is to inject knowledge of the hard physics processes as encoded in physics simulation directly into discriminant and approximate effects of detector reconstruction through so-called 'transfer functions'
  - At level of hard physics simulation, calculation of probability model for truth-level quantities still tractable (although still relatively expensive)
  - Add effects of parton showers and detector resolution a posteriori with transfer functions

The Matrix Element Method



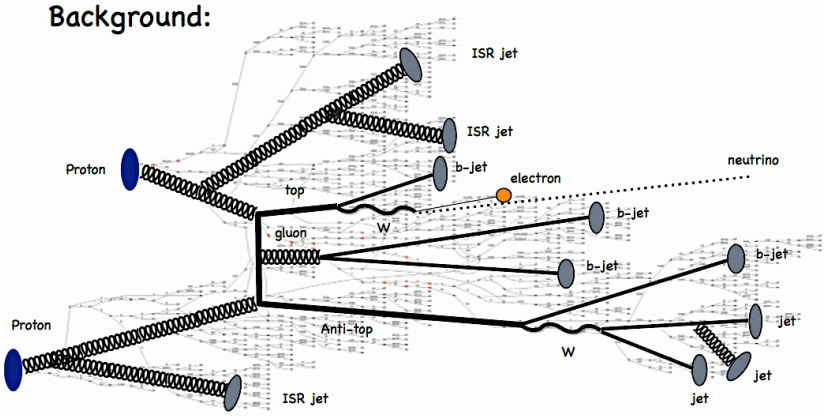
The Matrix Element Method



The Matrix Element Method

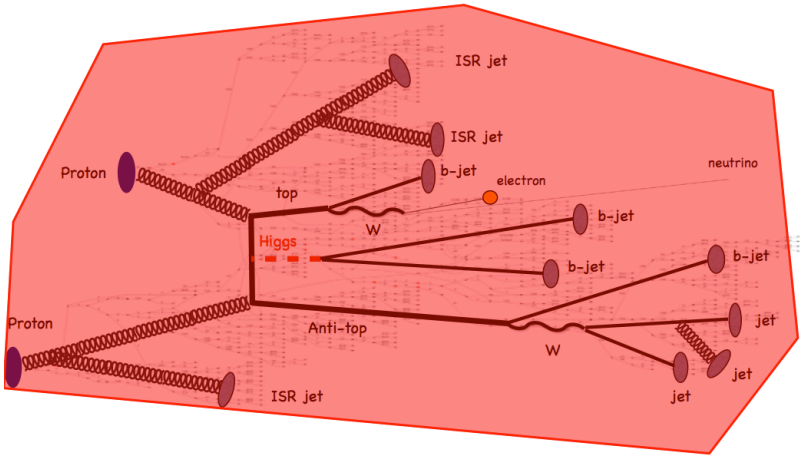
Give final state radiation distinctive meaning in terms of hypothesis

Background:



The Matrix Element Method

Ideally one would like to use all radiation related to hard process to discriminate signal from background



The Matrix Element Method

The matrix element method in a nutshell:

Y = parton-level final state
X = reconstruction-level final state

Given a theoretical assumption α, attach a weight $P(\mathbf{x}, \alpha)$ to each experimental event **x** quantifying the validity of the theoretical assumption α for this event.

$$P(\mathbf{x}, \alpha) = \frac{1}{\sigma} \int d\phi(\mathbf{y}) |M_\alpha|^2(\mathbf{y}) W(\mathbf{x}, \mathbf{y})$$

P(x,α) = L(x|H$_α$)
α=S,B

$|M_\alpha|^2$ is squared matrix element = S(y) or B(y) from theory (=calculable!)

$W(\mathbf{x}, \mathbf{y})$ is the resolution or transfer function

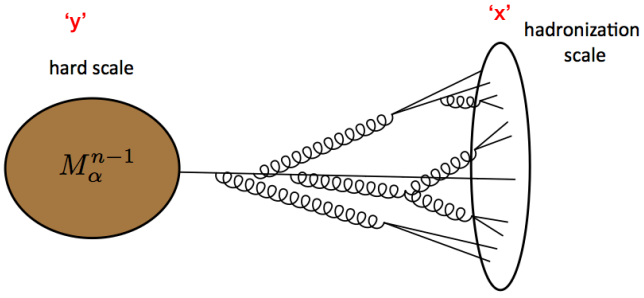
$d\phi(\mathbf{y})$ is the parton-level phase-space measure

The value of the weight $P(\mathbf{x}, \alpha)$ is the probability to observe the experimental event **x** in the theoretical frame α

λ_{MEM}=P(x,S)/p(x,B)

The Matrix Element Method

Purpose of the transfer function is to match jets to partons



Probability density function: $\int d\mathbf{y}\, W(\mathbf{x}, \mathbf{y}) = 1$

The Matrix Element Method

W(x|y) = p.d.f for observable quantities x, given parton-level theory observables y

The form of the transfer function:

resolution in

$$W(\mathbf{x},\mathbf{y}) \approx \Pi_i \frac{1}{\sqrt{2\pi}\sigma_{E,i}} e^{-\frac{(E_i^{rec}-E_i^{gen})^2}{2\sigma_{E,i}^2}}$$ Energy

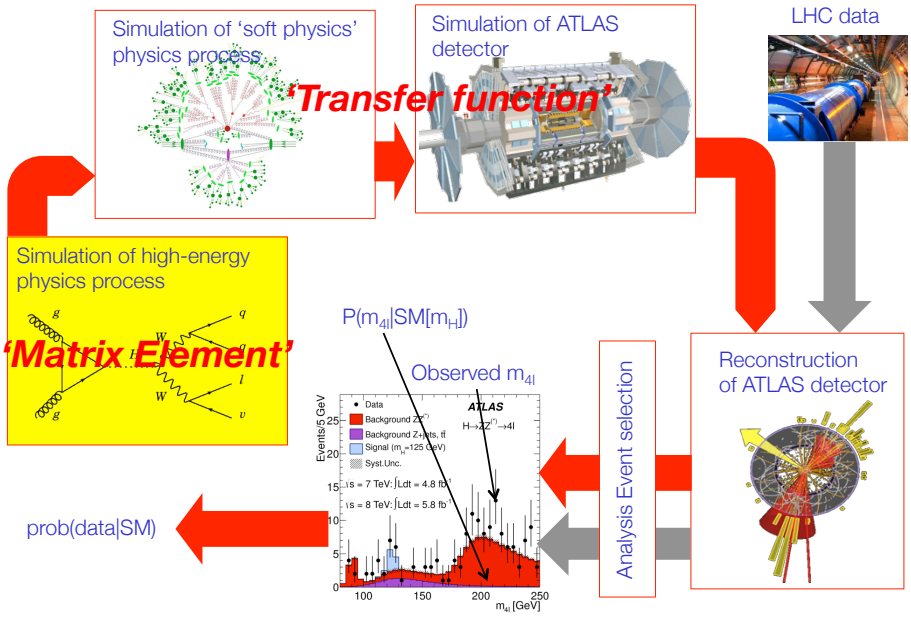
$$\times \frac{1}{\sqrt{2\pi}\sigma_{\phi,i}} e^{-\frac{(\phi_i^{rec}-\phi_i^{gen})^2}{2\sigma_{\phi,i}^2}}$$ azimuthal angle

$$\times \frac{1}{\sqrt{2\pi}\sigma_{y,i}} e^{-\frac{(y_i^{rec}-y_i^{gen})^2}{2\sigma_{y,i}^2}}$$ rapidity

Complex, high-dimensional gaussian distribution!

Transfer function introduces new peaks on top of propagators

An overview of HEP data analysis procedures



The Matrix Element Method

Subtleties of the convolution $|M(y)|^2 \times W(y,x)$

1) $|M(y)|^2$

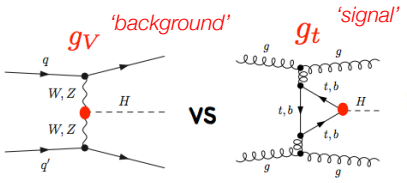
- Can be calculated at different order in pert. series (LO, NLO)
- Final state multiplicity fixed (exclusive process)
- Some kinematic configurations induce large logs (need resummation)

2) $W(y,x)$

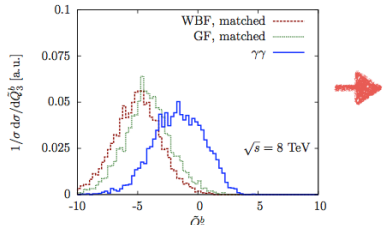
- Number of final state objects limited to exclusive process
- Integration very time consuming -> limits final state multiplicity
- Transfer function fit dependent (input from experiment)

The Matrix Element Method

Example g_V 'background' vs g_t 'signal' in $H \to \gamma\gamma$



Higgs reconstructed, but no transfer function for jets:

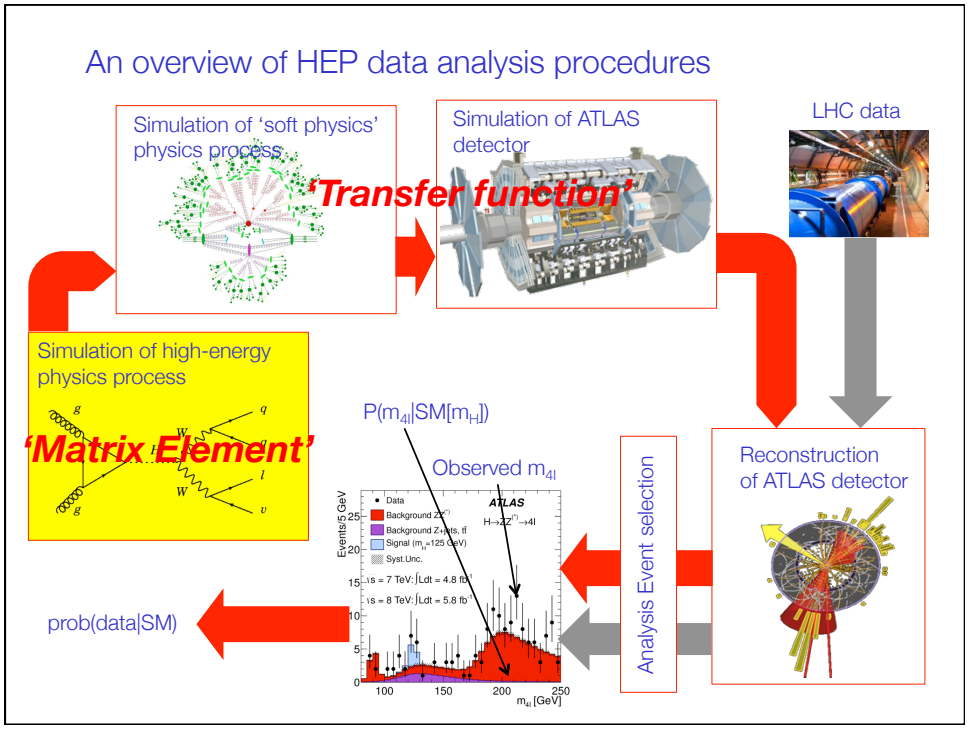


$S/B \nearrow 100\%$

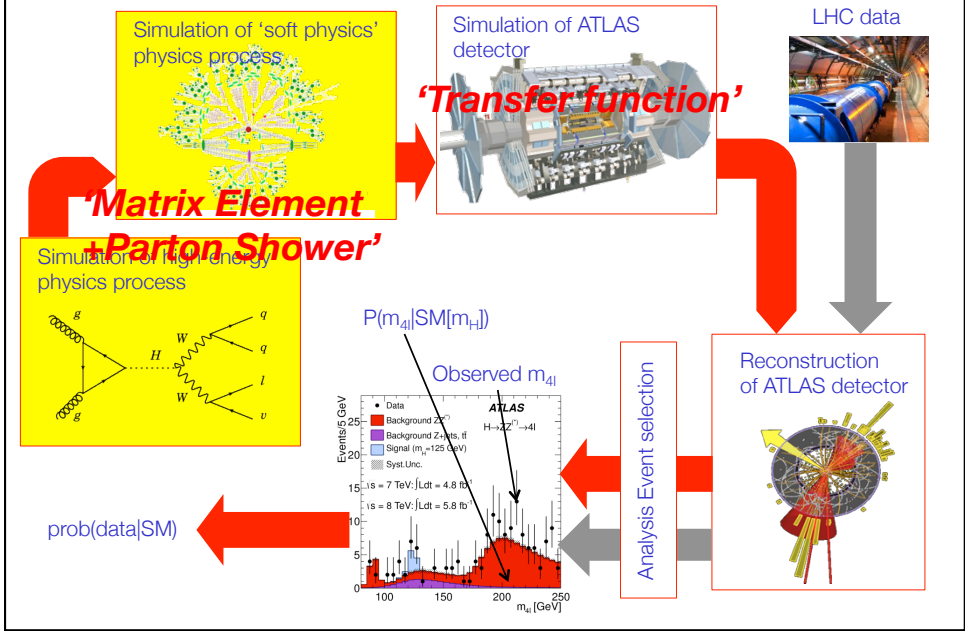
$$\tilde{Q}_n^b(p_1^\gamma, p_2^\gamma, \{p_n^j\}) = -\log\left[\frac{\{|\mathcal{M}^{\mathrm{WBF}}(pp \to (h \to \gamma\gamma)j^n)|^2 + |\mathcal{M}^{\mathrm{GF}}(pp \to (h \to \gamma\gamma)j^n)|^2\}}{|\mathcal{M}^{2\gamma}(pp \to \gamma\gamma j^n)|^2}\right]$$

Likelihood-Ratio based on Matrix Elements

An overview of HEP data analysis procedures



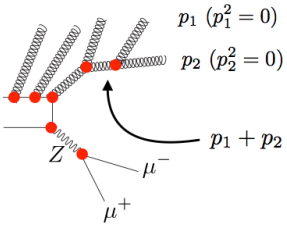
An overview of HEP data analysis procedures



The Matrix Element Method

Remove limitation of final state objects on $|M(y)|^2$

Shower approximation for matrix element, i.e. **shower deconstruction:**



partons from the hard interaction emit other partons (gluons and quarks)

These emissions are enhanced if they are collinear and/or soft with respect to the emitting parton

Probability enhanced in soft and collinear region due to ~ $1/(p_1 + p_2)^2$

- If $p_1 \to 0$, then $1/(p_1 + p_2)^2 \to \infty$
- If $p_2 \to 0$, then $1/(p_1 + p_2)^2 \to \infty$
- If $p_2 \to \lambda p_1$, then $1/(p_1 + p_2)^2 \to \infty$

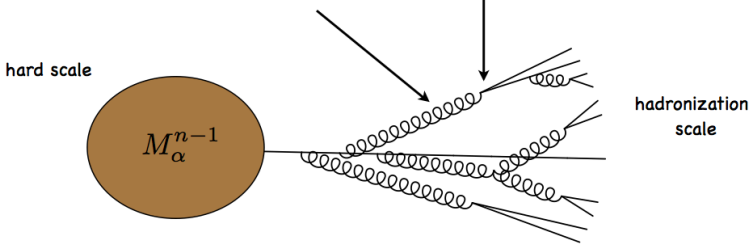
The Matrix Element Method

Factorization of emissions in soft/collinear limit

and Sudakov factors allow semiclassical approximation of quantum process:

propagator-lines = Sudakov factors

vertices = Splitting functions



Can calculate weight for shower history iteratively

Can use smaller objects and more objects (more information)

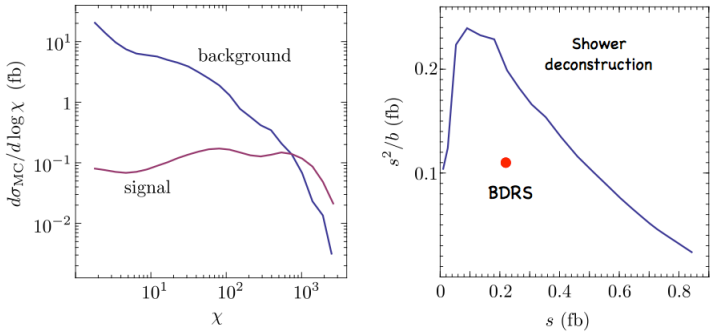
The Matrix Element Method

Likelihood-Ratio based on Matrix Elements with Parton Shower deconstruction

Results for Higgs boson:

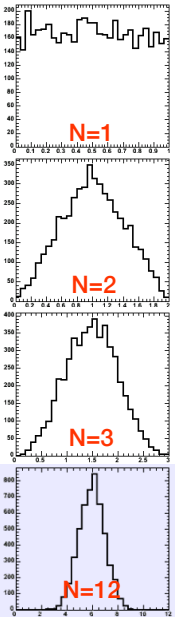
$$\chi(\{p,t\}_N) = \frac{P(\{p,t\}_N|\mathrm{S})}{P(\{p,t\}_N|\mathrm{B})}$$

[Soper, MS PRD 84 (2011)]



imperfect b-tagging (60%,2%) no b-tag required

Ex 1 - Demonstration of Central Limit Theorem



← 5000 numbers taken at random from a uniform distribution between [0,1].

- Mean = $^1/_2$, Variance = $^1/_{12}$

← 5000 numbers, each the sum of 2 random numbers, i.e. $X = x_1 + x_2$.

- Triangular shape

← Same for 3 numbers, $X = x_1 + x_2 + x_3$

← Same for 12 numbers, overlaid curve is exact Gaussian distribution

Important: tails of distribution converge very slowly CLT often *not* applicable for '5 sigma' discoveries

Ex 1 - Implications of non-Gaussian tails in distributions

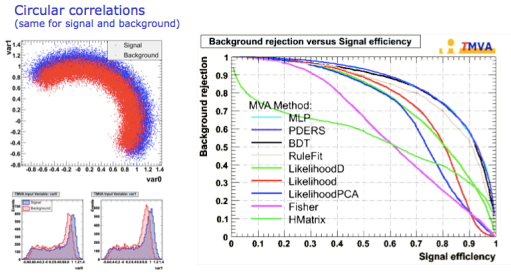
- If (tails of) distributions are not Gaussian, familiar mapping between 'standard deviations' and probabilities does not apply
  - If you have done your exercise correctly you'll see the following results for the Nsum=20 run with Nexp=10.000.000 for 1,2,3,4,5 sigma

```
n = 3198780 frac = 0.319879  +/- 0.00017 Gauss = 0.317311
n = 450384  frac = 0.0450384 +/- 6.7e-05 Gauss = 0.0455003
n = 22954   frac = 0.0022954 +/- 1.5e-05 Gauss = 0.0026998
n = 329     frac = 3.29e-05 +/- 1.8e-06  Gauss = 6.33425e-05
```

- Non-Gaussian tails can lead to significant deviations in probabilistic interpretation when Gaussian distribution is erroneously assumed
  - Probability '4 Gaussian sigma' fluctuation = 6.3 10^{-5}
  - Probability '4 standard deviation' fluctuation = 3.3 10^{-5} (=3.8 Gaussian sigma)
- For large significances, explicit calculation using actual distribution is need (more on this in the afternoon)

Exercises

- If you have not finished the exercises of yesterday (Ex1, Ex2), please leave them for now [solutions are now also provided for your convenience `ex1sol.C` `ex2sol.C` at www.nikhef.nl]
- Now: Practice machine-learned test statistics with TMVA (Ex 4)



- Easy-to-use setup (to generate toy samples for S, B, and to run TMVA training provided
- Note: Last-minute bug discovered in ROOT 5.34.21 for MacOS!
  - If your TMVA session on Mac crashes, please apply provided fix (see exercises for details). Recompilation time less than 30 seconds…