e/π separation in EMC using TMVA

Malgorzata Gumberidze

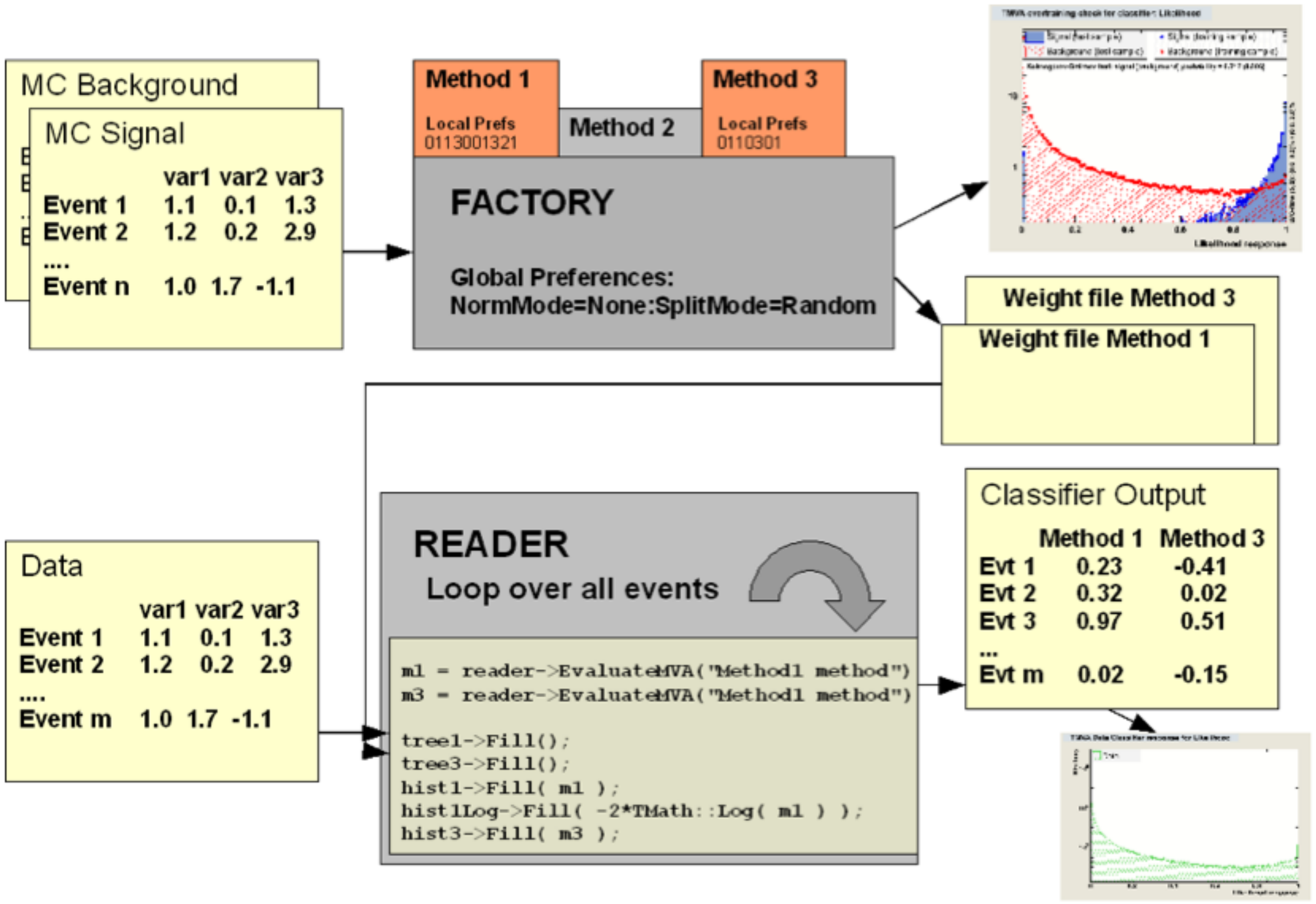
What it is TMVA

TMVA: The Toolkit for MultiVariate Analysis

- Have one common platform / interface for high-end multivariate 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

Basic steps of the analysis using TMVA



```
m1 = reader->EvaluateMVA("Method1 method")
m3 = reader->EvaluateMVA("Method1 method")

tree1->Fill();
tree3->Fill();
hist1->Fill( m1 );
hist1Log->Fill( -2*TMath::Log( m1 ) );
hist3->Fill( m3 );
```

MVA methods included in TMVA

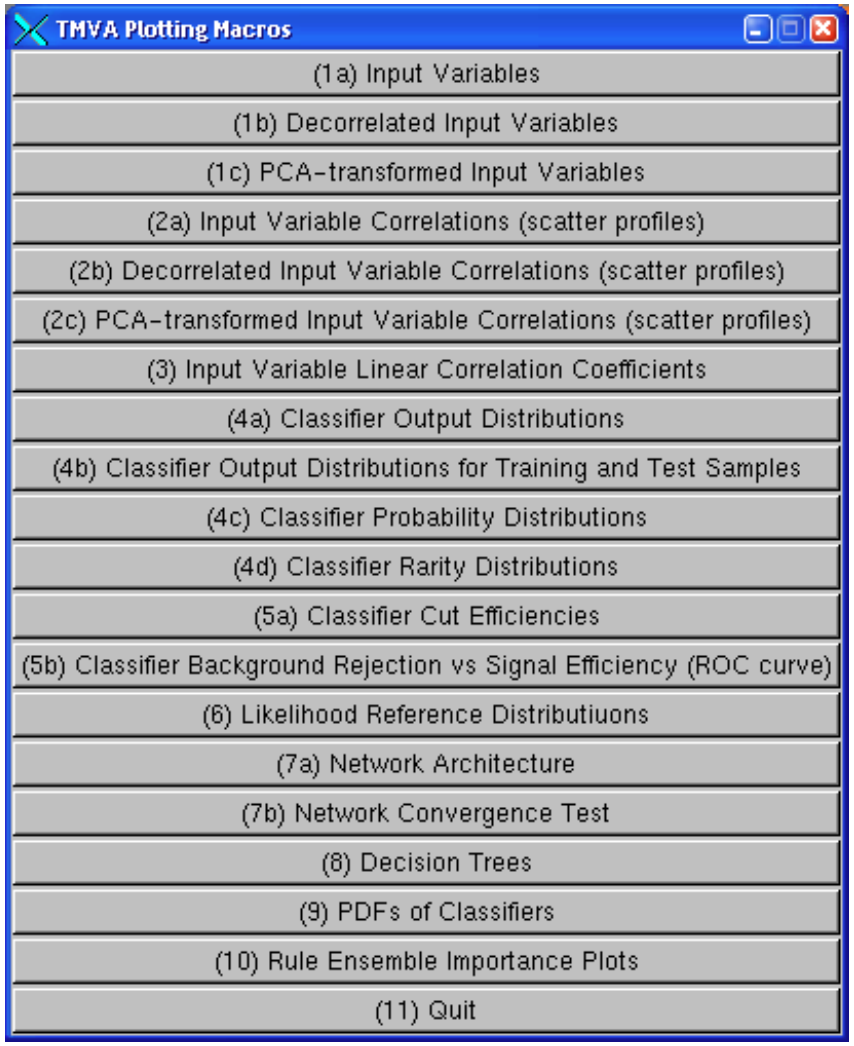
- Rectangular cut optimization
- Projection likelihood estimation
  - ➔studied independly by R. Kunne
  - ➔secondary focus of this work
- Multidimensional probability density estimation
  - Probobility density estimator range search (PDERS)
  - Multidimensional K-Nearest Neighbour (K-NN) ➔ studied previously by M. Babai
- Linear discriminant analysis
  - H-Matrix (χ^2) Estimator
  - Fisher Discriminant
  - Function Discriminant Analysis (FDA)
- Boosted/Bagged decision trees (BDT)
- Artificial neural networks (ANN)
  - Clermont-Ferrand neural network
  - ROOT neural network
  - Multilayer Perceptron (MLP) neural network
    - ➔used previously in so-call 'Babar framework'
    - ➔ main focus of this work
- Predictive learning via rule ensemble ()Rule-Fit
- Support Vector Machine (SVM)

So single good classifier

Criteria		Classifiers								
		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		🙂	🙂	😐	🙂	🙂	☹️	☹️	☹️	☹️

MVA Evaluation Tool

- TMVA is not only a collection of classifiers, but an MVA framework
- After training, TMVA provides ROOT evaluation scripts (through GUI)



TMVA Plotting Macros

(1a) Input Variables
(1b) Decorrelated Input Variables
(1c) PCA–transformed Input Variables
(2a) Input Variable Correlations (scatter profiles)
(2b) Decorrelated Input Variable Correlations (scatter profiles)
(2c) PCA–transformed Input Variable Correlations (scatter profiles)
(3) Input Variable Linear Correlation Coefficients
(4a) Classifier Output Distributions
(4b) Classifier Output Distributions for Training and Test Samples
(4c) Classifier Probability Distributions
(4d) Classifier Rarity Distributions
(5a) Classifier Cut Efficiencies
(5b) Classifier Background Rejection vs Signal Efficiency (ROC curve)
(6) Likelihood Reference Distributiuons
(7a) Network Architecture
(7b) Network Convergence Test
(8) Decision Trees
(9) PDFs of Classifiers
(10) Rule Ensemble Importance Plots
(11) Quit

Plot all signal (S) and background (B) input variables with and without pre-processing

Correlation scatters and linear coefficients for S & B

Classifier outputs (S & B) for test and training samples (spot overtraining)

Classifier *Rarity* distribution

Classifier significance with optimal cuts

B rejection versus S efficiency

Classifier-specific plots:
- Likelihood reference distributions
- Classifier PDFs (for probability output and Rarity)
- Network architecture, weights and convergence
- Rule Fitting analysis plots

- Visualise decision trees

Using TMVA in PANDA

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

- ➢ **IN PROGRESS**

② *Application phase*: using selected trained classifiers to classify unknown data samples

- ➢ **IN NEXT STEP**

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

➢ **IN PROGRESS**

② *Application phase*: using selected trained classifiers to classify unknown data samples

➢ **IN NEXT STEP**

Input to the training:

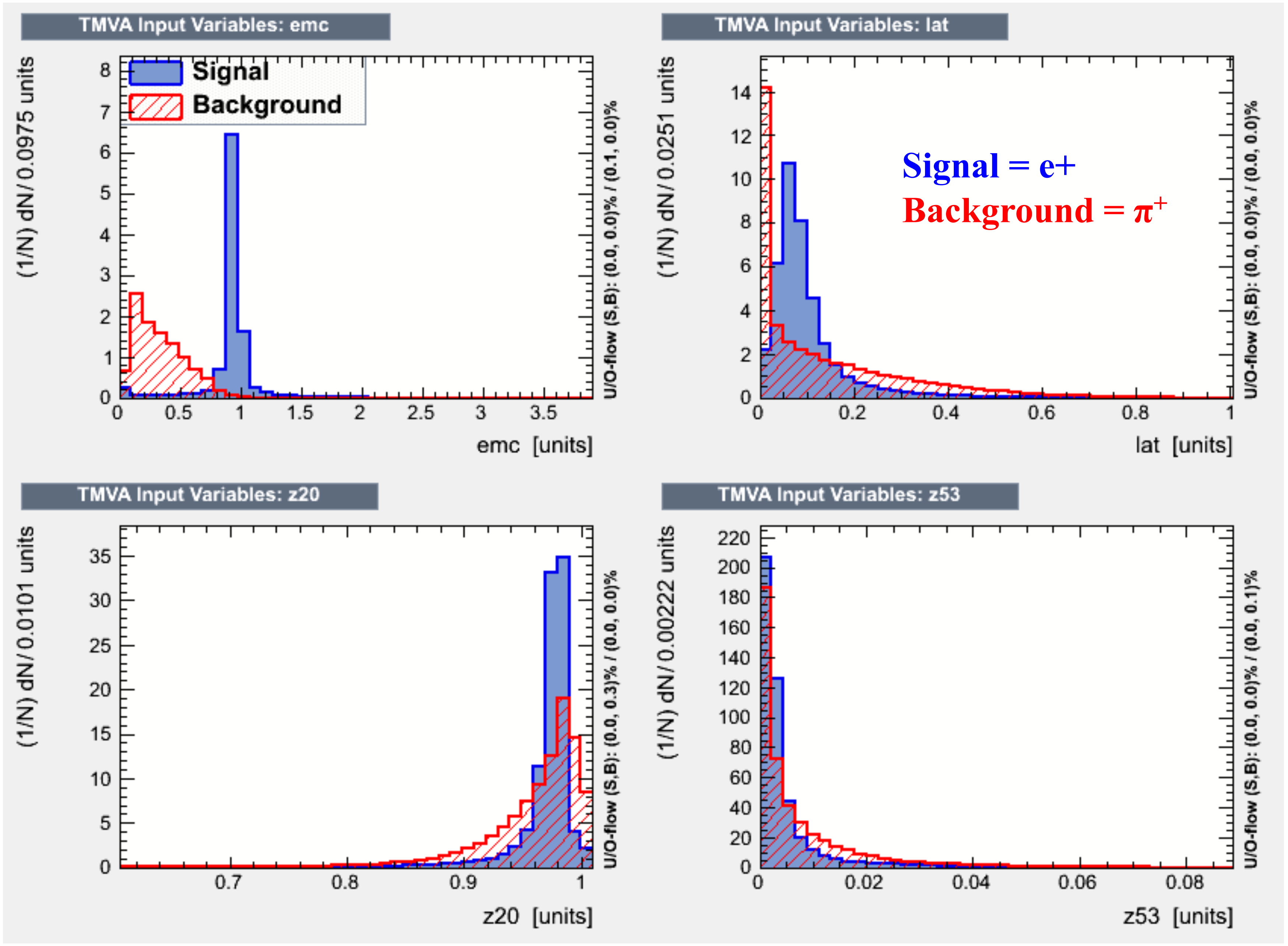
✓ single white (in p, θ, φ) particles: e and π, 10^6 events of each, p in range (0.2; 2 GeV/c)

✓ for each MC track the best reconstructed selected (based on the p value)

✓ all tracks has EMC index

✓ several variables tested to separation: E/p, z20, z53, lateral momenta, (p, θ, Npoints)

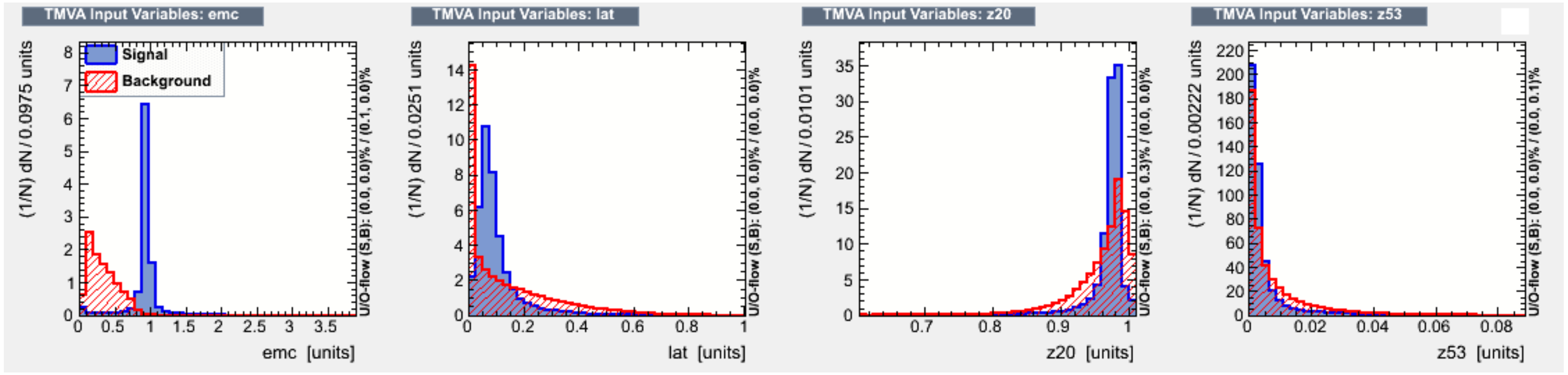
Multivariable event classification

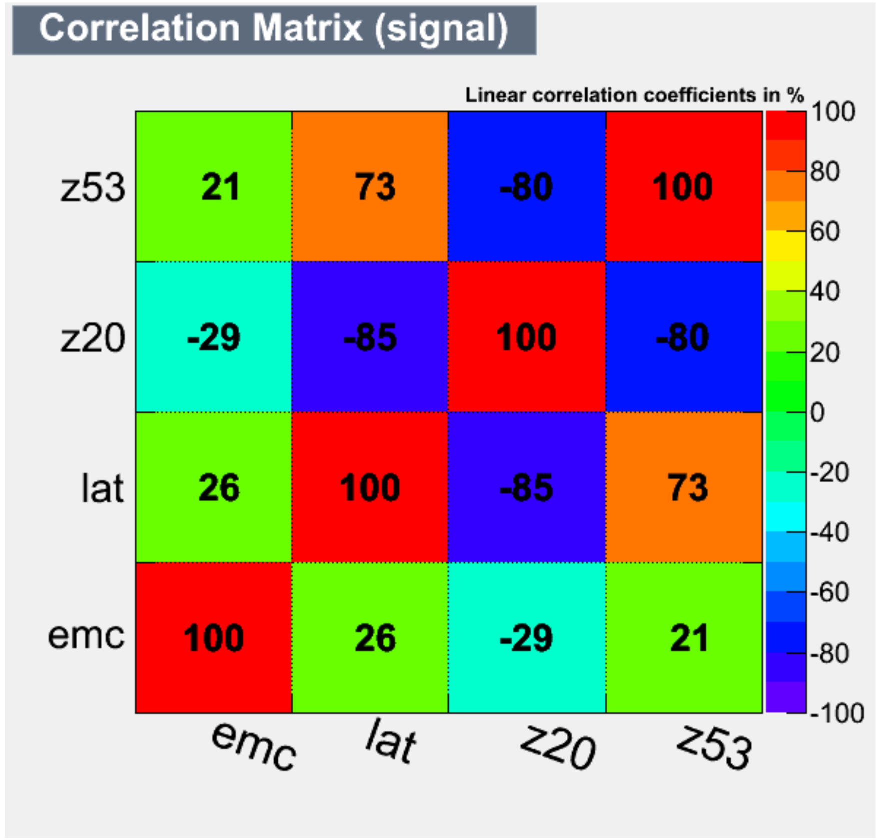


Signal = e+

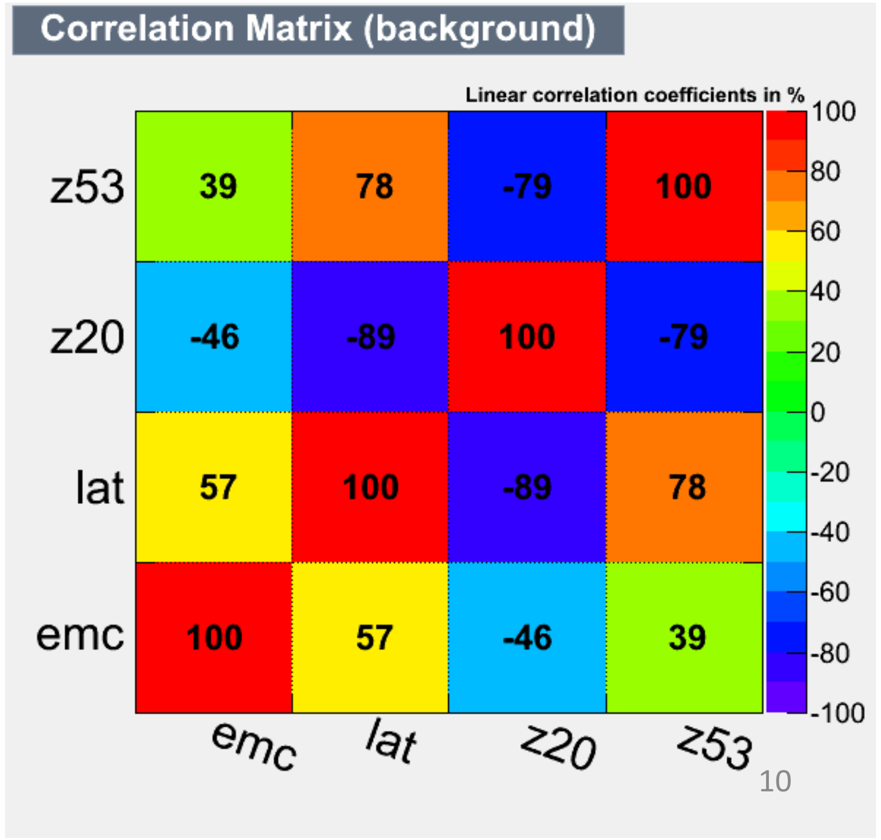
Background = π^+

Multivariable event classification





	emc	lat	z20	z53
z53	21	73	-80	100
z20	-29	-85	100	-80
lat	26	100	-85	73
emc	100	26	-29	21



	emc	lat	z20	z53
z53	39	78	-79	100
z20	-46	-89	100	-79
lat	57	100	-89	78
emc	100	57	-46	39

What should be the architecture of the MLP?

For a multilayer perceptron a single hidden layer is sufficient to approximate a given continuous correlation function to any precision, provided that a sufficiently large number of neurons is used in the hidden layer.

If the available computing power and the size of the training data sample suffice, one can increase the number of nodes in the hidden layer until the optimal performance is reached.

It is likely that the same performance can be achieved with a network of more than one hidden layer and a potentially much smaller total number of hidden nodes. This would lead to a shorter training time and a more robust network.

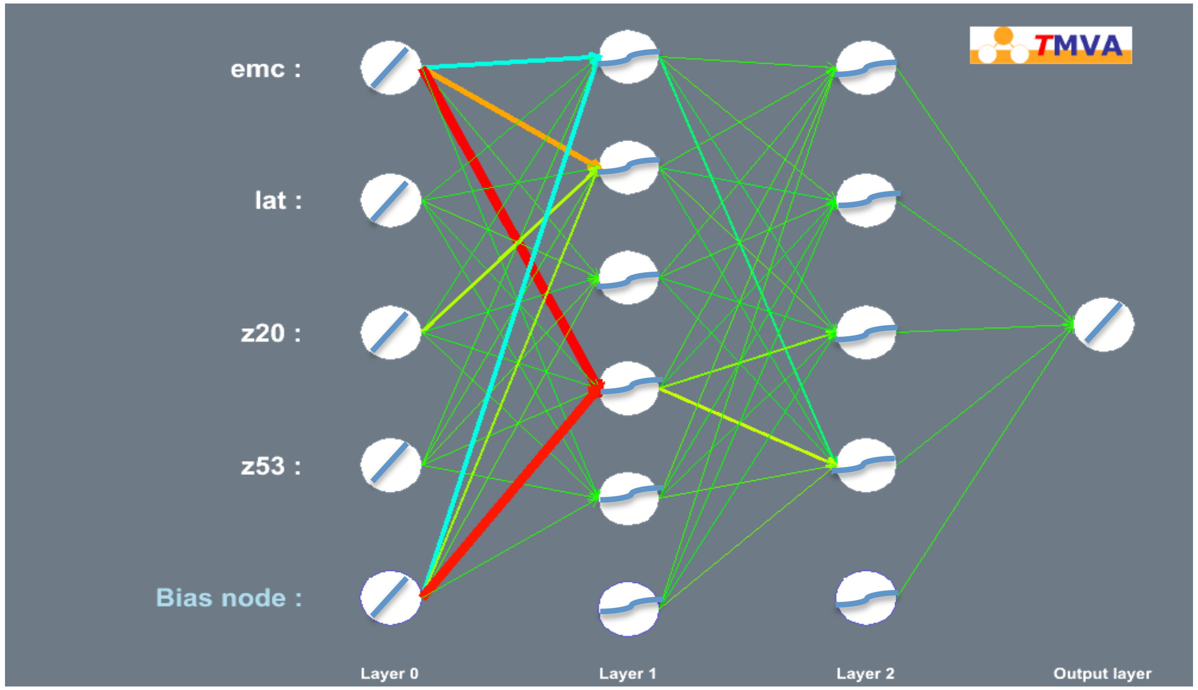
MLP architecture

$$ {}^{(k,l)} = \left(w_0 + \sum_{i=1}^{n} x_i w_i^{(k,l)} \right) * \alpha $$

w_0 - bias node

α – node activation function

$$\alpha : x \rightarrow \begin{cases} x & \textit{Linear}, \\ \dfrac{1}{1+e^{-kx}} & \textit{Sigmoid}, \\ \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}} & \textit{Tanh}, \\ e^{-x^2/2} & \textit{Radial}. \end{cases}$$



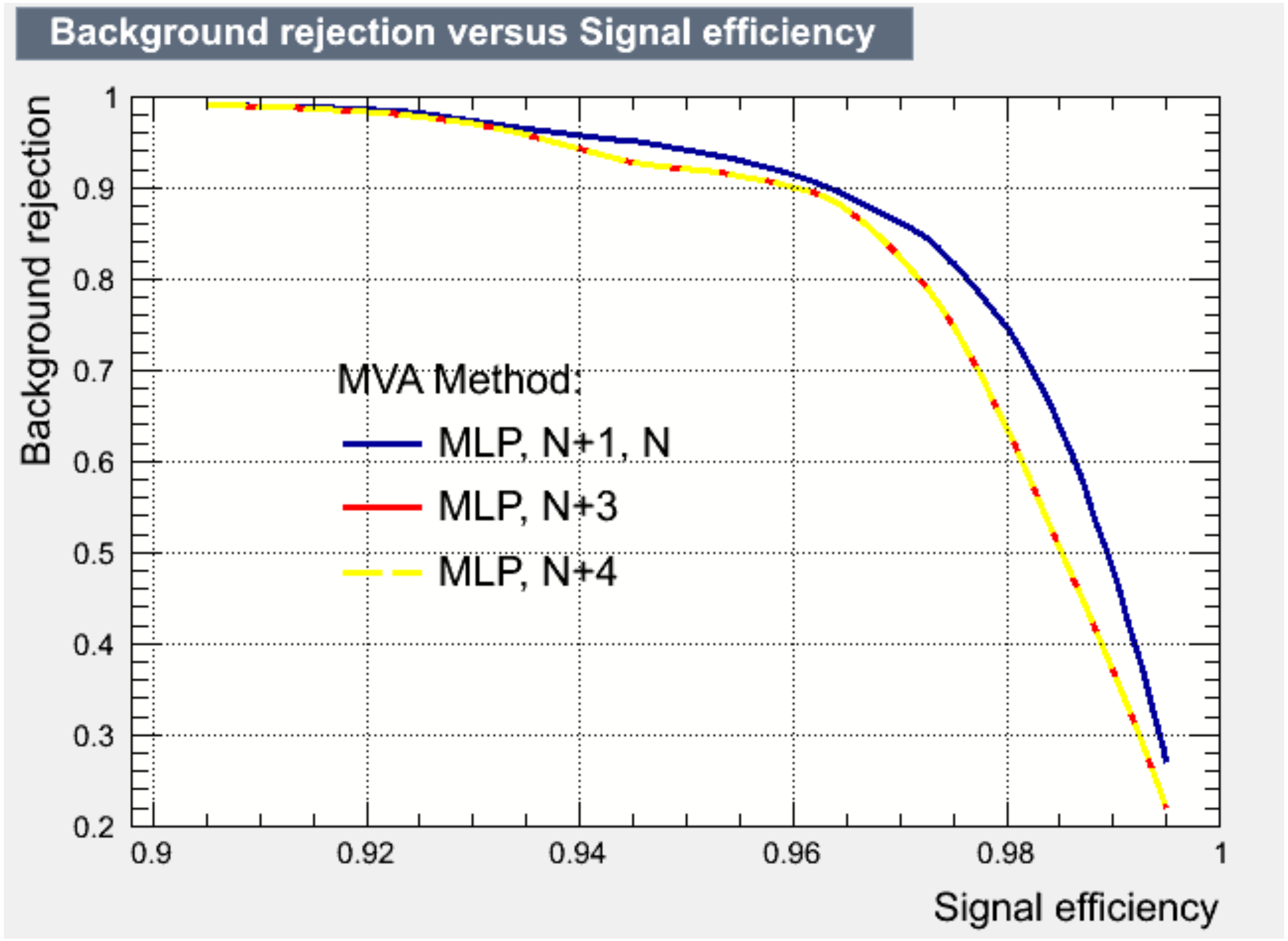
- ❖ input layer 0 (N_{var} + 1)
- ❖ 2 hidden Layers 1 and 2
- ❖ Output Layer 3
- ❖ Arrows reflect the coupling strength (dark blue: strong-negative, red: strong positive)

- ❖ Connection only in one direction -> feed forward network (no backward loops)
- ❖ Connection weighted and during the learning process the set of weights is searched which minimizes a given error function
- ❖ Classifiers output is calculated by the node response "activation" function.
- ❖ Nodes (neurons) in hidden layers represent the "activation functions" whose arguments are linear combinations of input variables -> non-linear response to the input

focus on MLP method

ROC: Relative Operating Characteristic

The resulting relations between background rejection versus signal efficiency can be displayed using **ROC curve.**

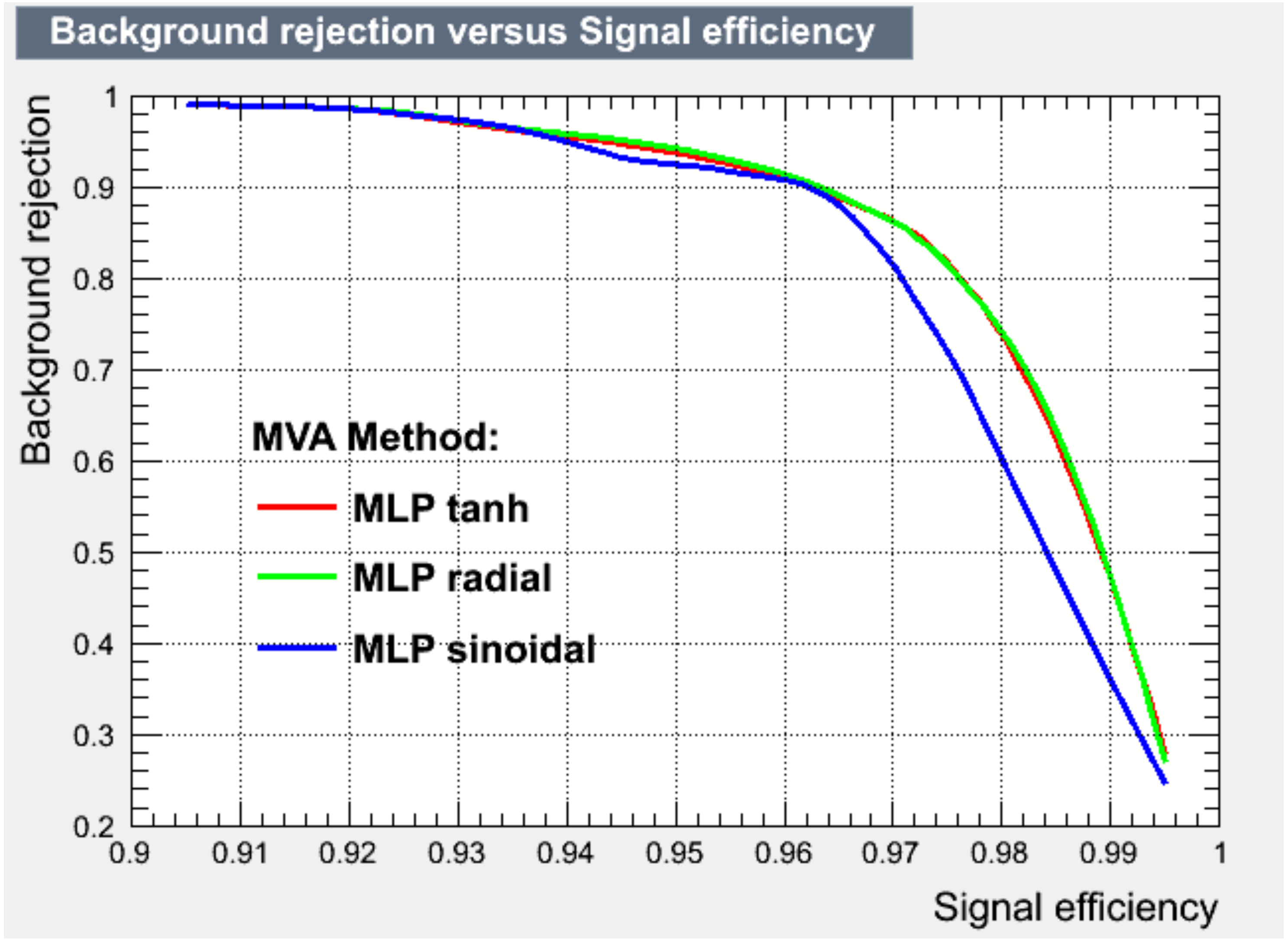


2 hidden layer: N+1, N
1 hidden layer: N+3
1 hidden layer N+4

N – number of input variables

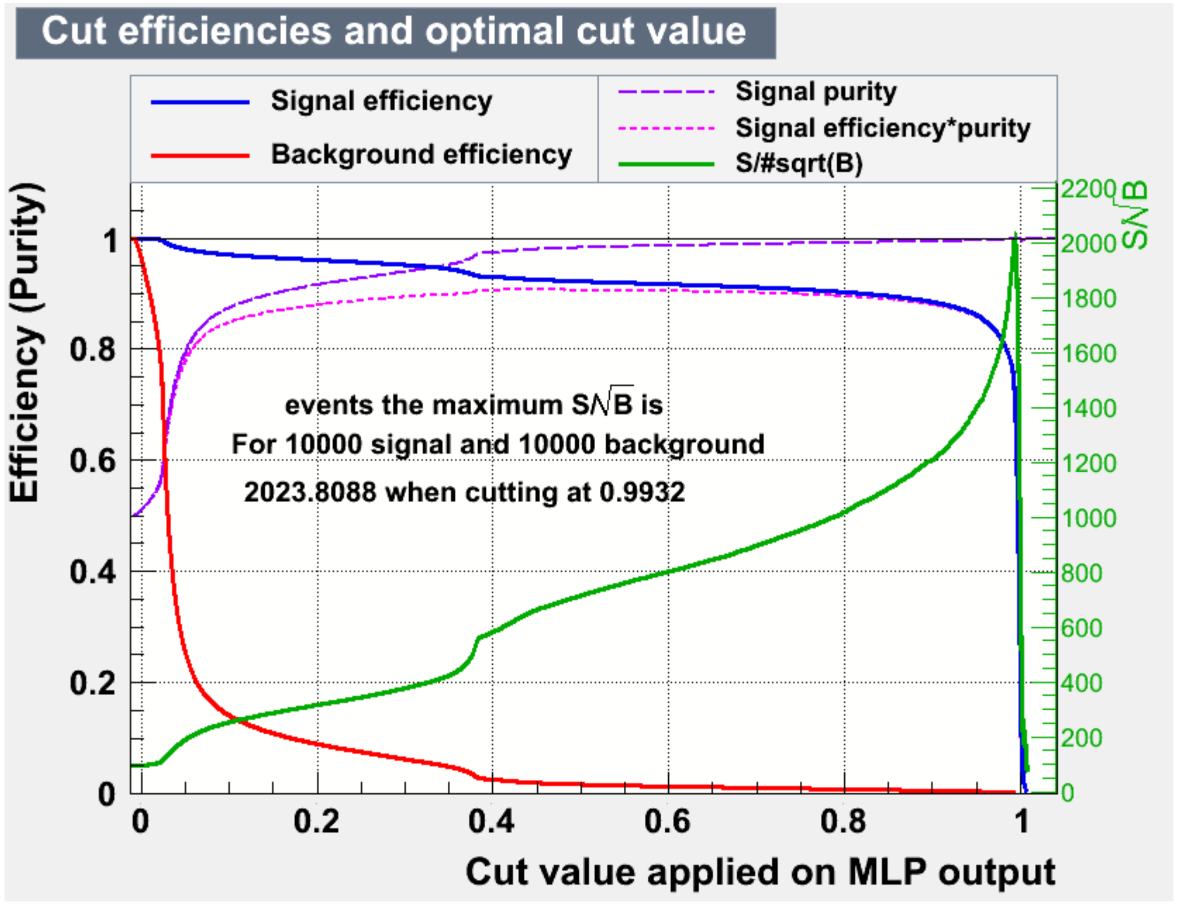
Small differences in the performance in case of 2 hidden layers and 1 hidden layer: for further performance 2 hidden layers has been chosen.

MLP: neuron activation function



for further simulations: tanh has been used (default value in TMVA)

MLP: Cut Efficiency



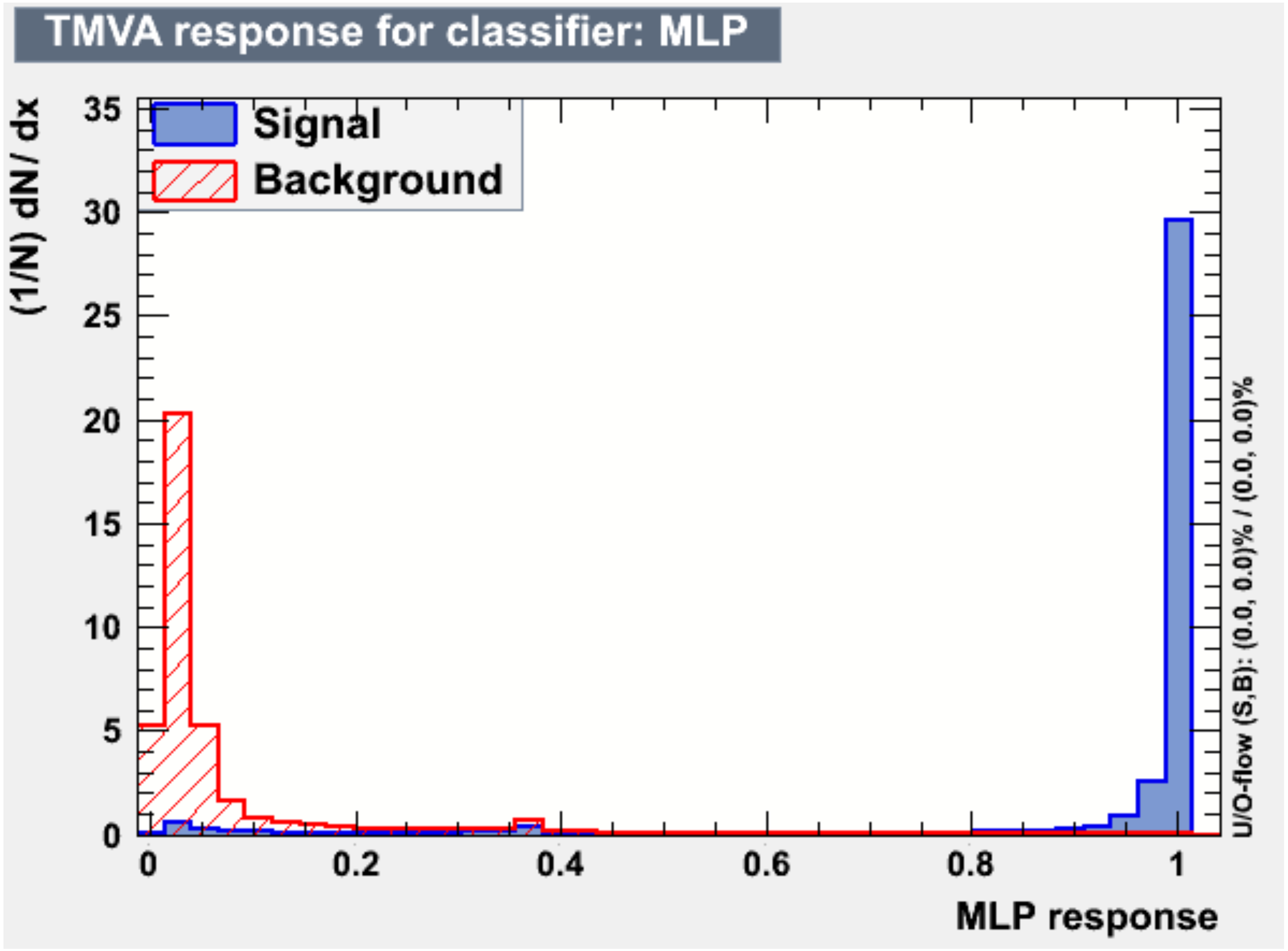
Working point: Optimal cut on a classifier output (=optimal point on ROC curve) depends on the problem:

- ❖Cross section measurement:
  - ✓maximum of S/√(S+B)
- ❖ Signal Search:
  - ✓maximum of S/√(B)
- ❖ Precision measurement:
  - ✓high purity
- ❖ Trigger selection:
  - ✓ high efficiency

❖ To deduct the optimal cut value on the classifier, one looks to the *efficiency* and *purity* plots. The maximum of signal *purity*efficiency* indicates the cut value.

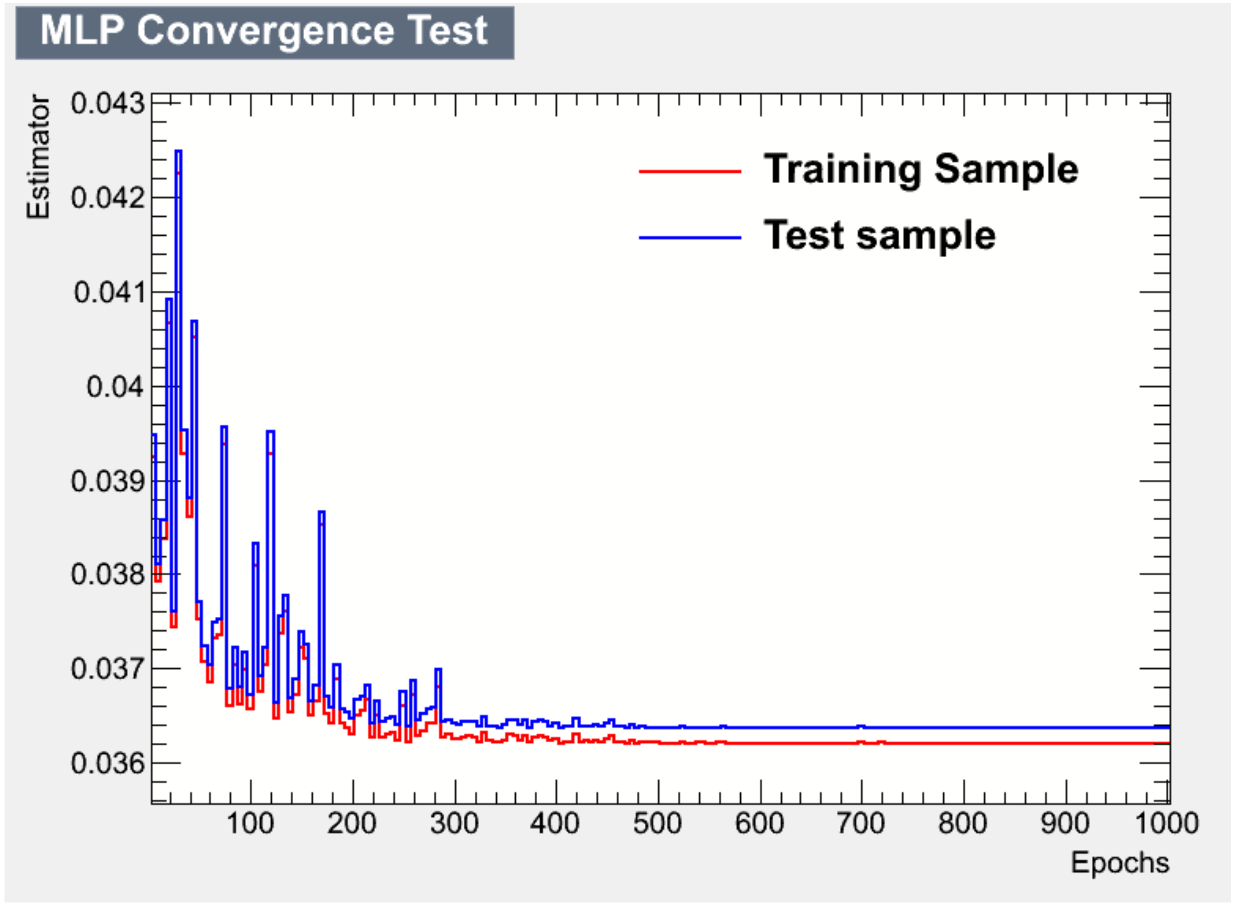
❖ Those curves were calculated during evaluation of the classifiers using a statistically independent signal/background sample.

Discriminator distributions



Convergence test

- Compare performance between training and test sample to detect overtraining

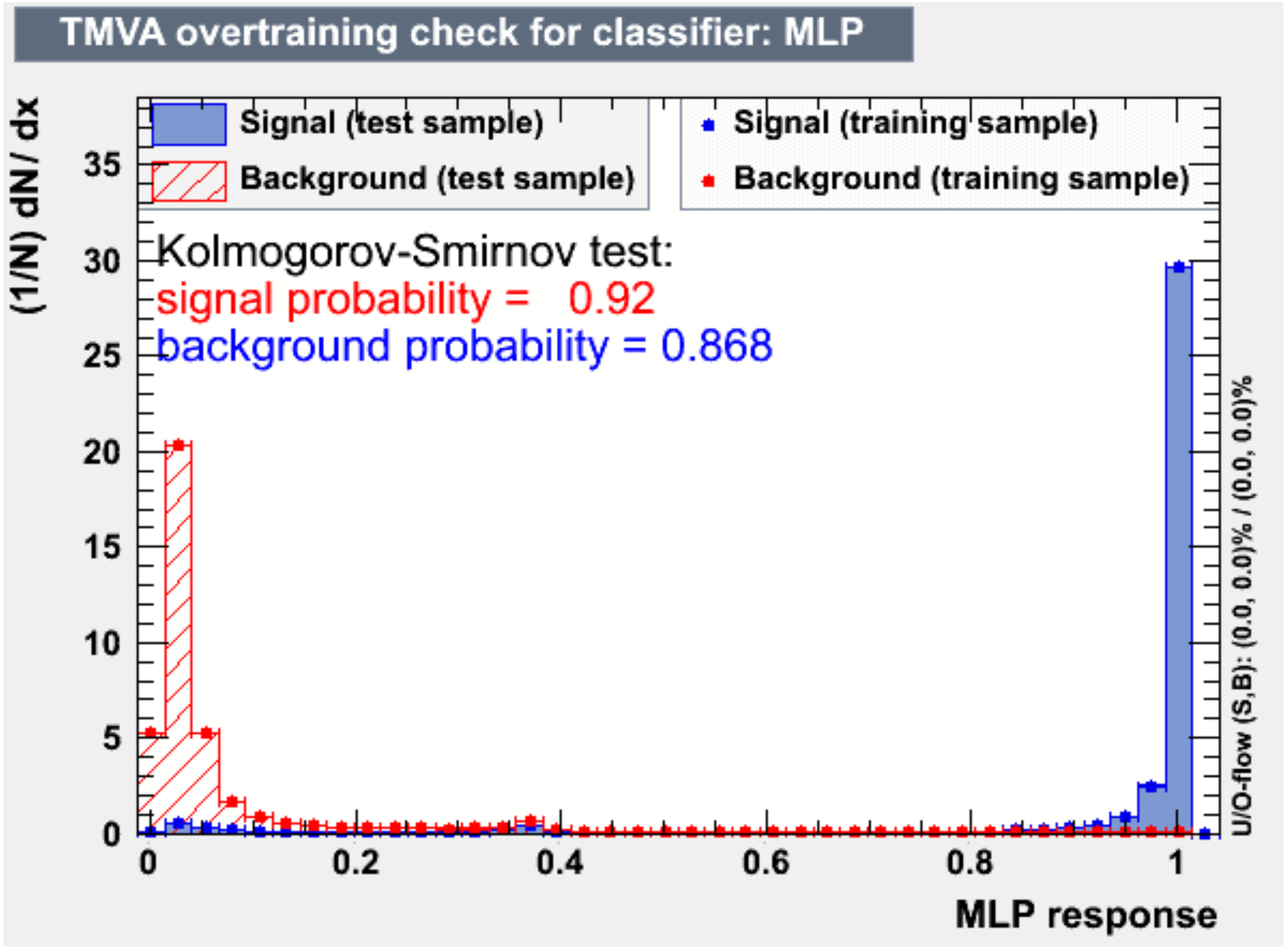


Compares the error estimator for the training and testing samples. If overtraining occurred, the estimator for the training sample should monotonously decrease, while the estimator of the testing sample should show a minimum after which it increases.

The set of weights and therefore the value of the error function has converged after a number of appreciatively 800 training cycles. The processing time rises with the number of training cycles. Hence it is reasonable not to do more training cycles than necessary.

4 variables, 1000 cycles (epchos) = 2h50min

Overtraining ….



no problem with the overtraining …

Summary

- ready simulation for the momentum range: 0.2 – 5 GeV/c
- additional variables added into the simulation in order to perform test using: E1, E9, E25 or ratios E1/E9, E9/E25, also other Zernik momenta: Z00, Z20, Z31, Z33 (variables used in old framework)

➢ to be presented next week

- comparison with Ronald method (using the same variables)
  - efficiency vs. suppression
- compare with other methods in TMVA

➢ to be ready in 2 weeks

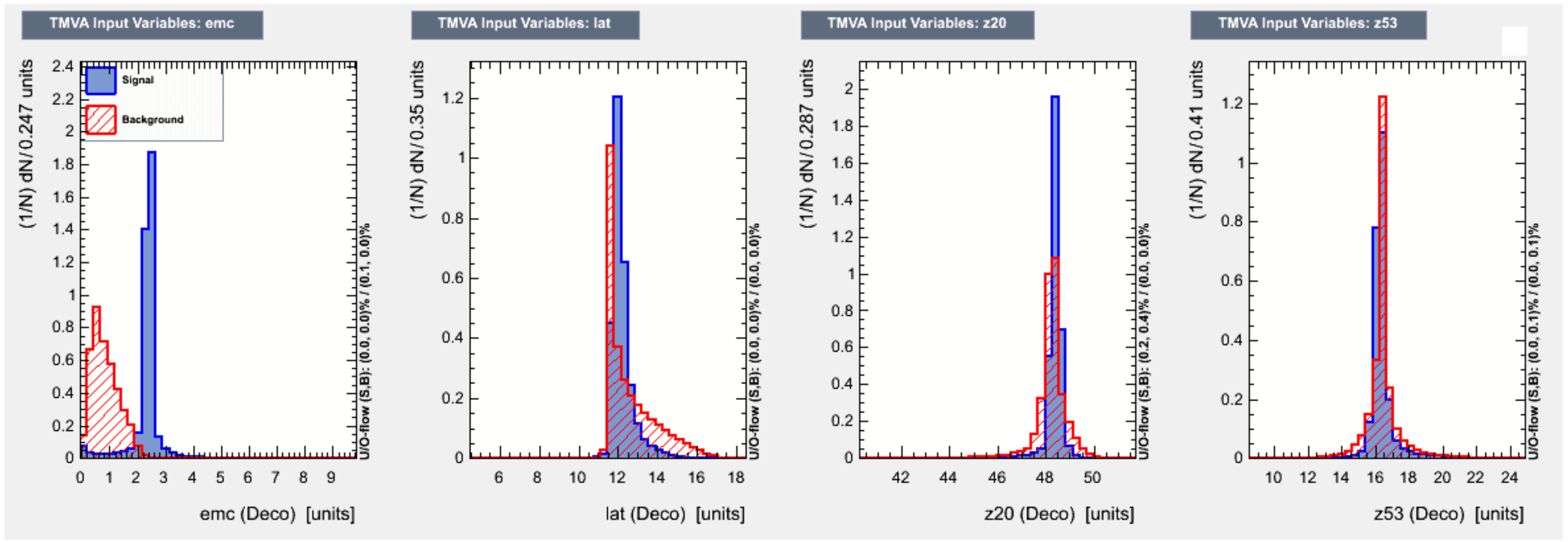
backup slides

Preprocesing of the input variables

TMVA provides decorrelation tools and advances methods that are more capable of dealing with correlated variables. Two methods are available for the decoralation of the variables:

determine ***square-root*** C' of the covariant matrix C,

- ✓ i.e., C = C' C'
- ✓ transformation from original (x) in de-correlated variable space (x') by x' = C' -1X



Attention. This is is able to eliminate only linear correlations!!

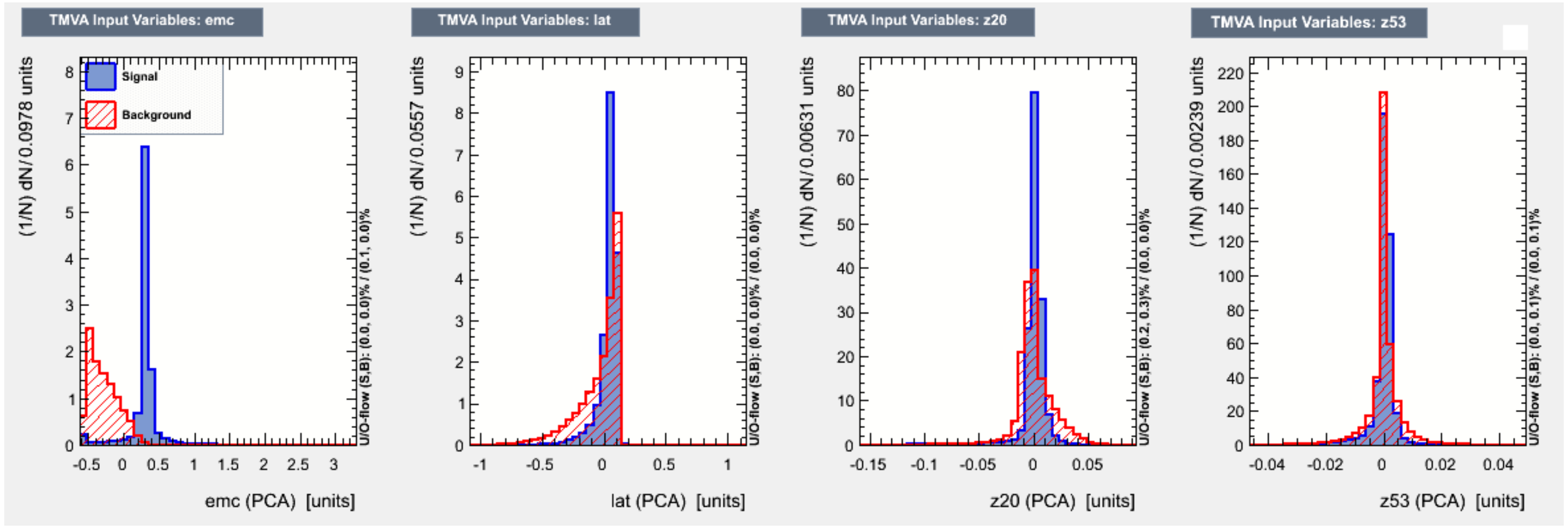
Preprocesing of the input variables

Principal Componant Analysis (PCA) is typically used to:

✓ reduce dimensionality of a problem

✓ find the most dominant features in your distribution by transforming

- The eigenvectors of the covariance matrix with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the data set. Along these axis the variance is largest
  - ✓ sort the eigenvectors according to their eigenvalues
- Dataset is transformed in variable space along these eigenvectors
  - ✓ Along the "first" dimension the data show the largest "features", the smallest features are found in the "last" dimension.



Matrix of eigenvectors V obey the relation: $C \cdot V = D \cdot V$ ➔ **PCA eliminates correlations!**

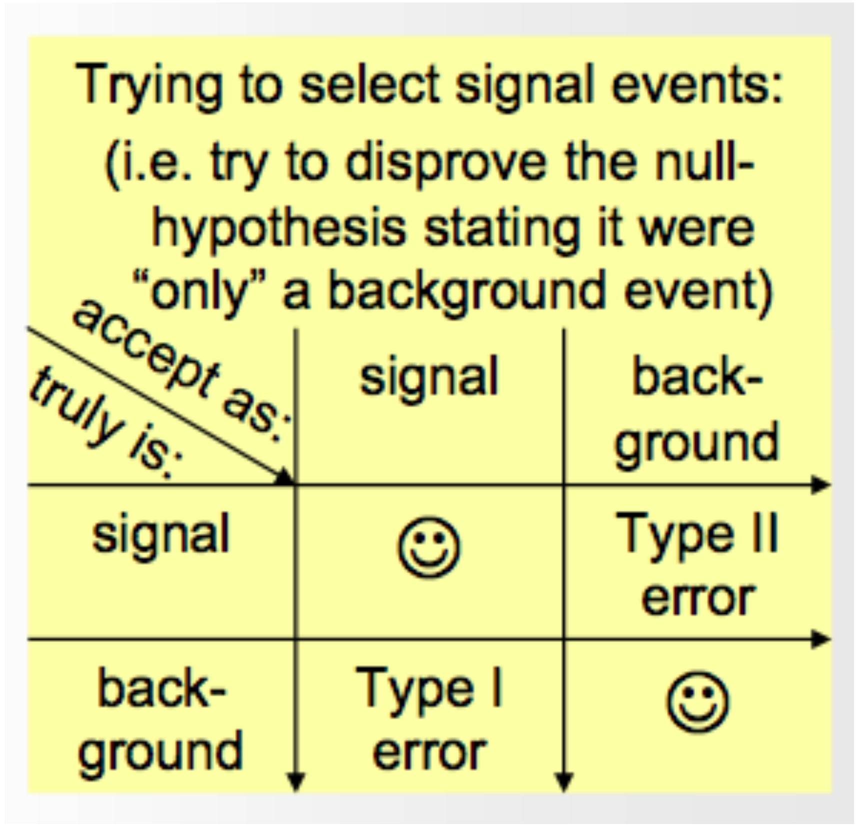
correlation matrix

diagonalised square root of C

Evaluation: any decision involves a certain risk

- ➢ decide to treat an event as "Signal" or "Background"

- ❖ False positive (type 1 error) = ε_{backgr}:
  - classify background event as signal
  - → loss of purity in the selection of signal evt
- ❖ False negative (type 2 error) = $1-\varepsilon_{signal}$:
  - fail to identify a signal event as such
  - → loss of efficiency in the selection of signal evt
  - True positive: ε_{signal}

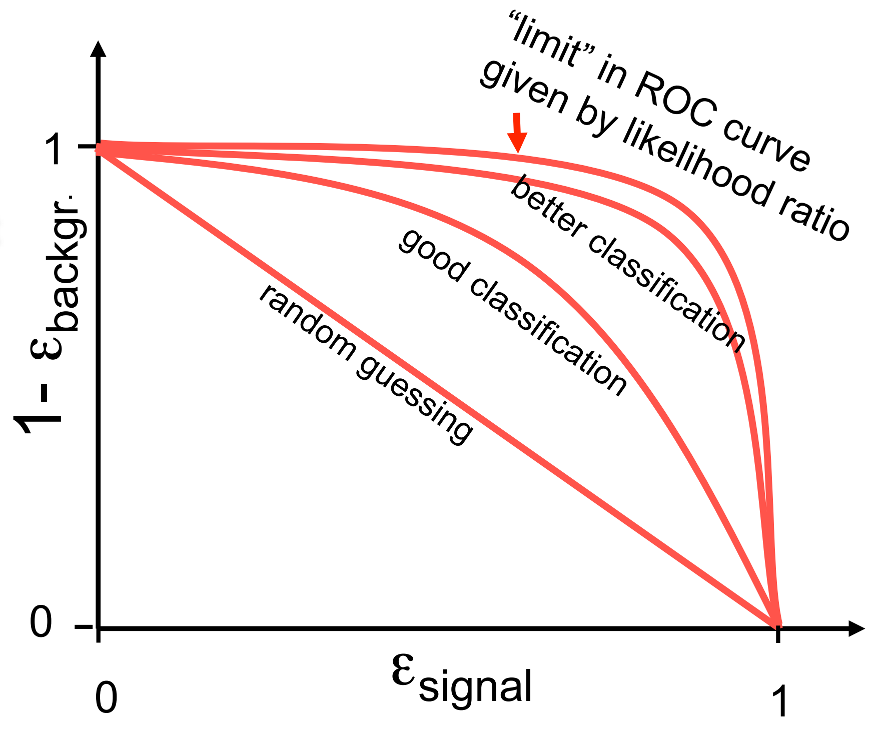


Trying to select signal events:
(i.e. try to disprove the null-hypothesis stating it were "only" a background event)

accept as: / truly is:	signal	back-ground
signal	☺	Type II error
back-ground	Type I error	☺

Evaluation: any decision involves a certain risk

- ROC curve describes performance of a binary classifier by plotting the false positive vs. the true positive fraction
- False positive (type 1 error) = $\varepsilon_{\text{backgr}}$:
  - classify background event as signal
  - → loss of purity in the selection of signal evt
- False negative (type 2 error) = $1-\varepsilon_{\text{signal}}$:
  - fail to identify a signal event as such
  - → loss of efficiency in the signal selection
  - True positive: $\varepsilon_{\text{signal}}$
- Likelihood ratio test $y(x) = \dfrac{P(S \mid x)}{P(B \mid x)}$



- The Likelihood ratio used as "selection criterion" y(x) gives for each selection efficiency the best possible background rejection (Neyman-Pearson)
  - It maximizes the area under the ROC-curve (PDE classifiers)

Optimal cut for each classifier

Working Point: Optimal cut on a classifier output (=optimal point on ROC curve) depends on the problem

❖Cross section measurement:	maximum of $S/\sqrt{(S+B)}$
❖Search:	maximum of $S/\sqrt{(B)}$
❖Precision measurement:	high purity
❖Trigger selection:	high efficiency

Parameters to be tune ….