Quick introduction to Multi Variate Analysis with ROOT: A short introduction to TMVA

Material from TMVA Workshop (Andreas Hoecker (CERN))

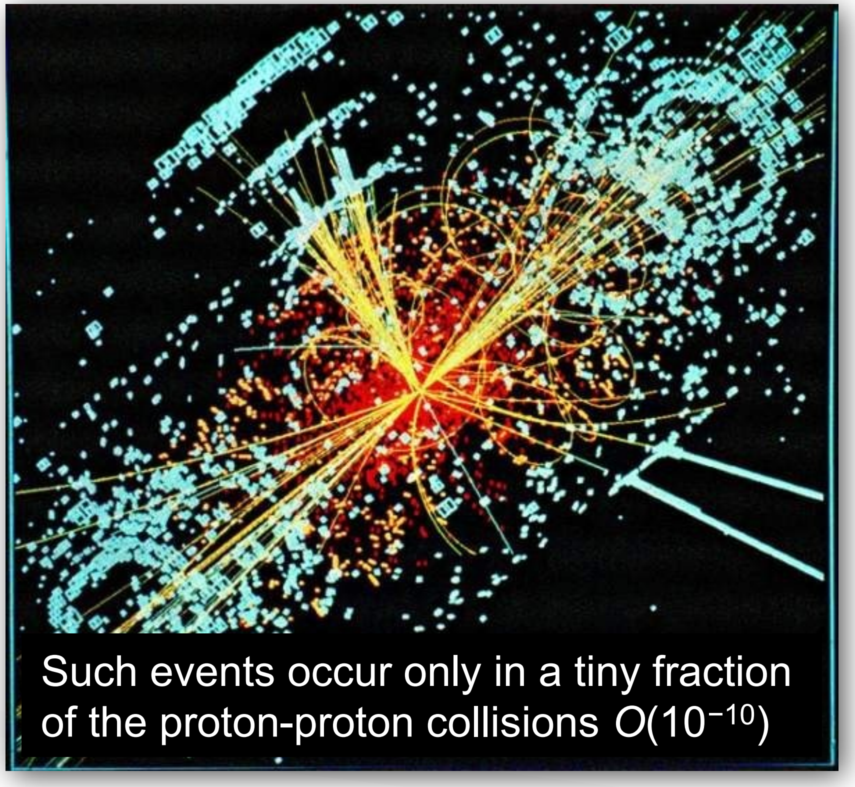
TMVA

- TMVA started in 2006 on the Sourceforge development platform
- 6 core developers, 21 contributors so far
- TMVA is written in C++, and relies on ROOT functionality
- Since ROOT 5.15 / TMVA v3.7.2 TMVA is part of ROOT, and developed directly in ROOT SVN
  - Continue to maintain primary *tmva-users* mailing list on Sourceforge
  - New TMVA versions also published as downloadable *tgz* files on Sourceforge
  - For bug reports, use ROOT Savannah

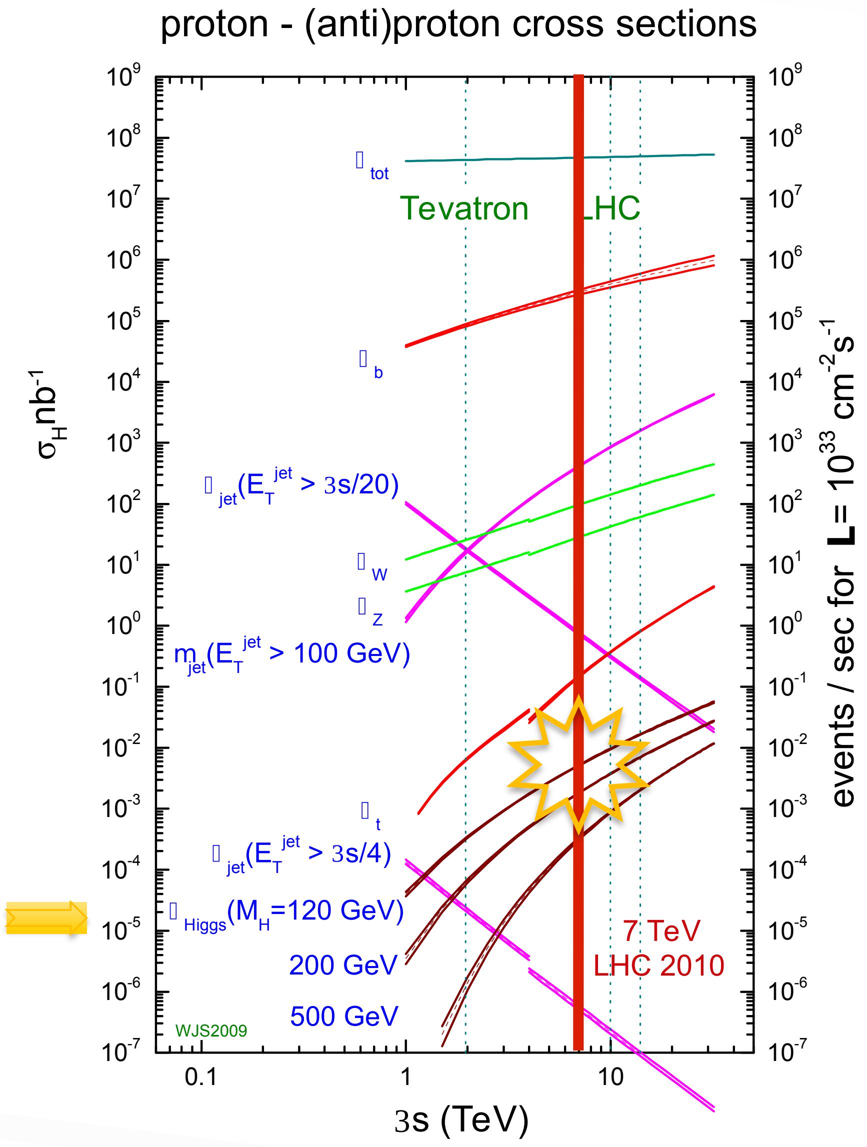
Simulated Higgs Event in CMS

Higgs event in an LHC proton–proton collision at high luminosity

(together with ~24 other inelastic events)



Such events occur only in a tiny fraction of the proton-proton collisions $O(10^{-10})$



Event Classification in HEP

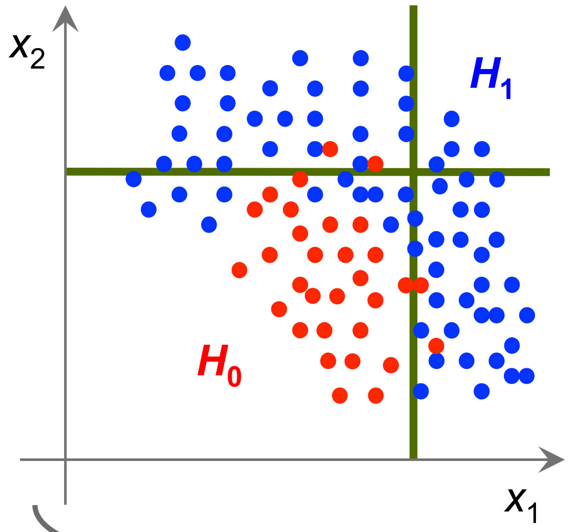
- Most HEP analyses require discrimination of signal from background:
  - Event level (Higgs searches, …)
  - Cone level (Tau-vs-jet reconstruction, …)
  - Track level (particle identification, …)
  - Object level (flavour tagging, …)
  - Parameter estimation (significance, mass, *CP* violation in *B* system, …)
- The multivariate input information used for this has various sources
  - Kinematic variables (masses, momenta, decay angles, …)
  - Event properties (jet/lepton multiplicity, sum of charges, …)
  - Event shape (sphericity, Fox-Wolfram moments, …)
  - Detector response (silicon hits, dE/dx, Cherenkov angle, shower profiles, muon hits, …)
- Traditionally few powerful input variables were combined. New methods allow to use up to 100 and more variables w/o loss of classification power

  e.g. MiniBooNE: NIMA 543 (2005), or D0 single top: Phys.Rev. D78, 012005 (2008)

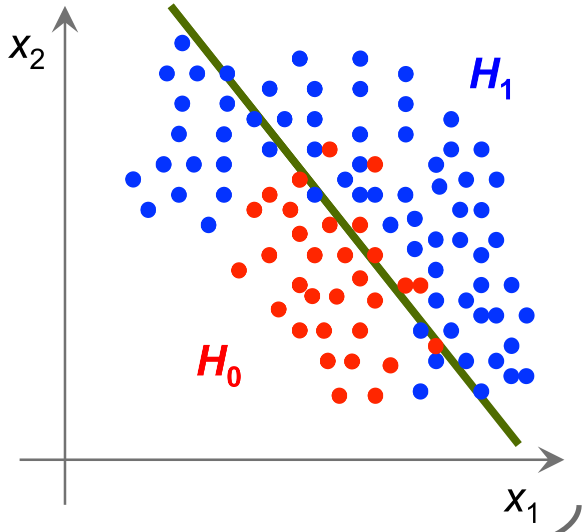
Event Classification

- Suppose data sample with two types of events: H_0, H_1
  - We have found discriminating input variables x_1, x_2, …
  - What decision boundary should we use to select events of type H_1 ?

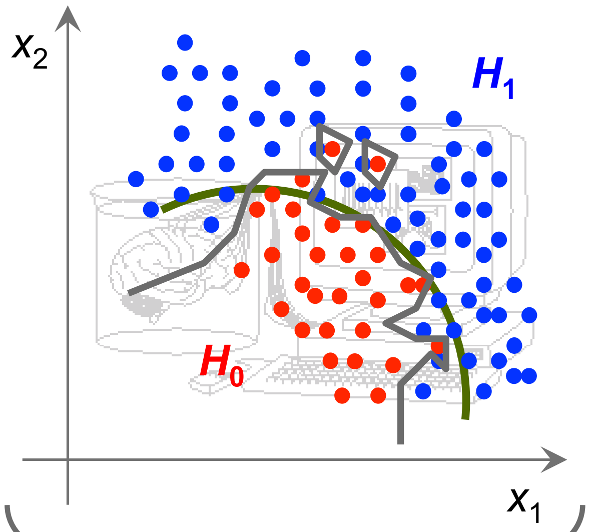
Rectangular cuts?



Linear boundary?



A nonlinear one?

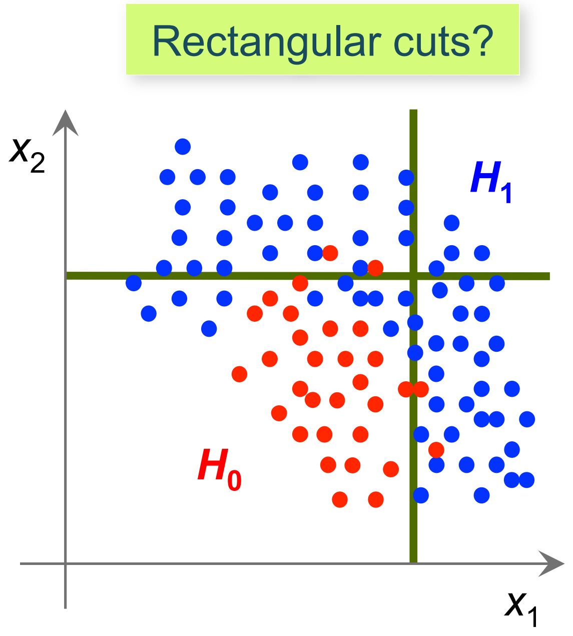


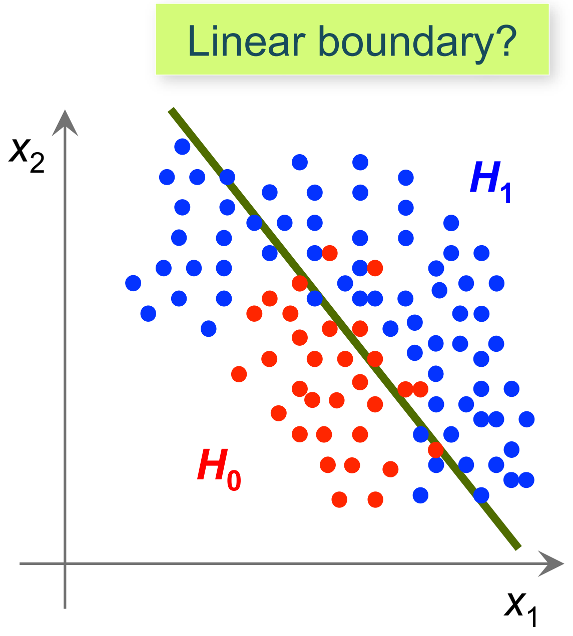
Low variance (stable), high bias methods

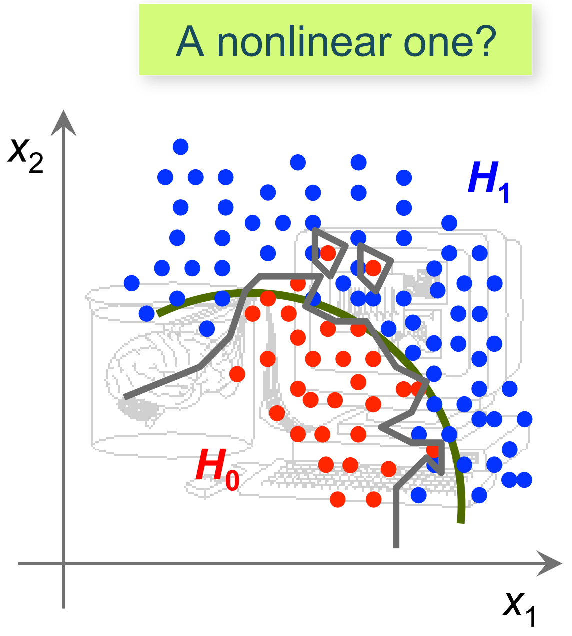
High variance, small bias methods

Event Classification

- Suppose data sample with two types of events: H_0, H_1
  - We have found discriminating input variables x_1, x_2, …
  - What decision boundary should we use to select events of type H_1 ?



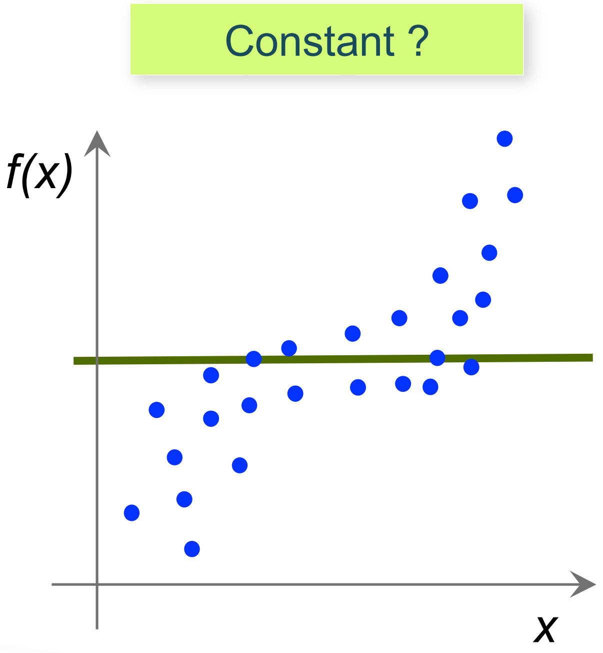


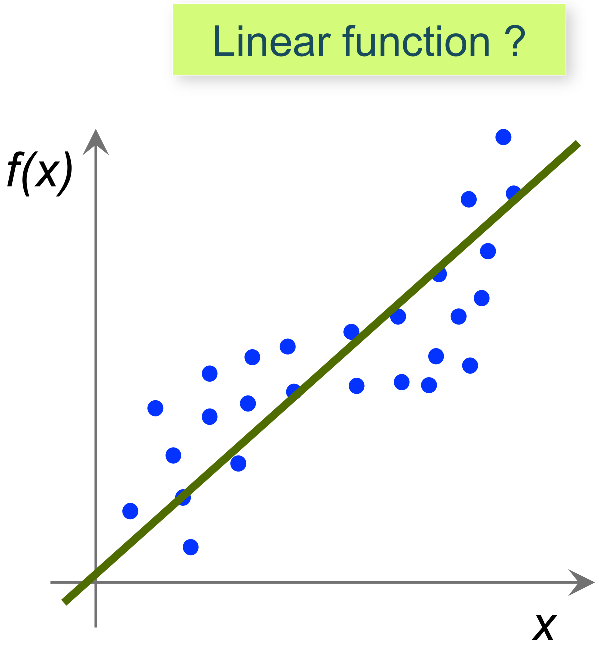


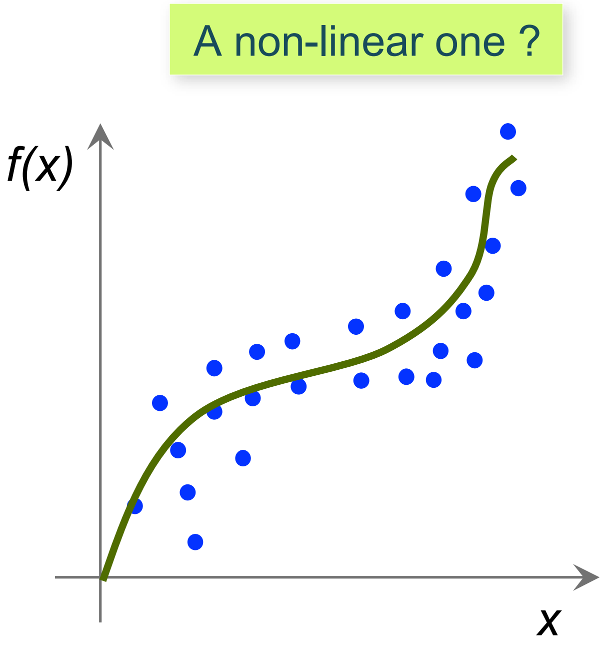
- How can we decide this in an optimal way ? → Let the machine learn it !

Parameter Regression

- How to estimate a "functional behaviour" from a set of measurements?
  - Energy deposit in a the calorimeter, distance between overlapping photons, …
  - Entry location of a particle in the calorimeter or on a silicon pad, …

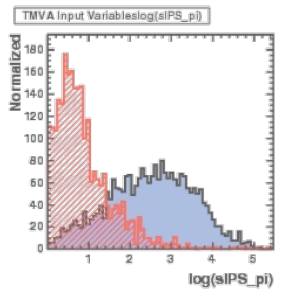


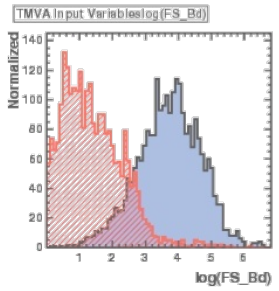


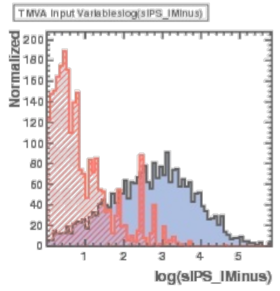


- Looks trivial? What if we have many input variables?

Multivariate Event Classification





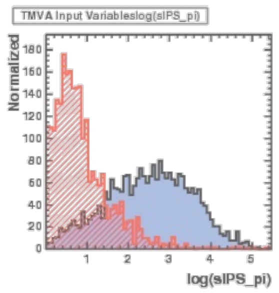


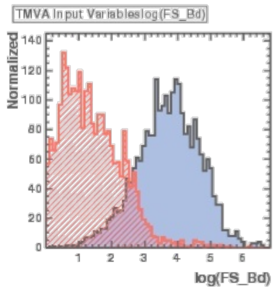
R^D

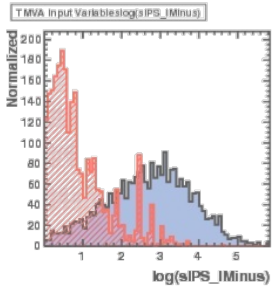
“feature space”

Each event, Signal or Background, has “*D*” measured variables.

Find a mapping from *D*-dimensional input-observable = ”feature” space to one dimensional output à ☾class labels







R^D

"feature space"

Each event, Signal or Background, has "*D*" measured variables.

Find a mapping from *D*-dimensional input-observable = "feature" space to one dimensional output à class labels

$y(x): R^n \rightarrow R$:

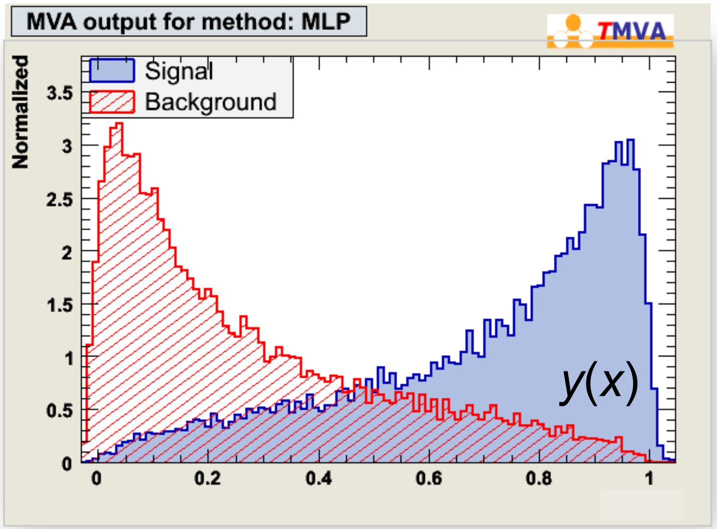
$$\xrightarrow{\quad\quad} \mathrm{R}$$

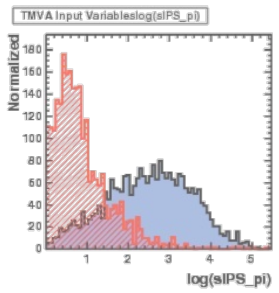
Most general form

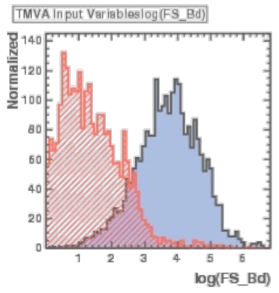
$y = y(\mathbf{x})$; $\mathbf{x}$ in $\mathbf{R}^D$

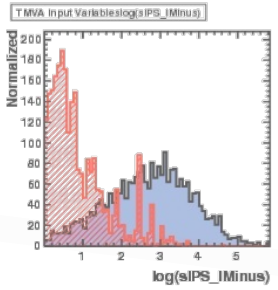
$\mathbf{x} = \{x_1,\ldots,x_D\}$: input variables

Plotting the resulting *y*(*x*) values:





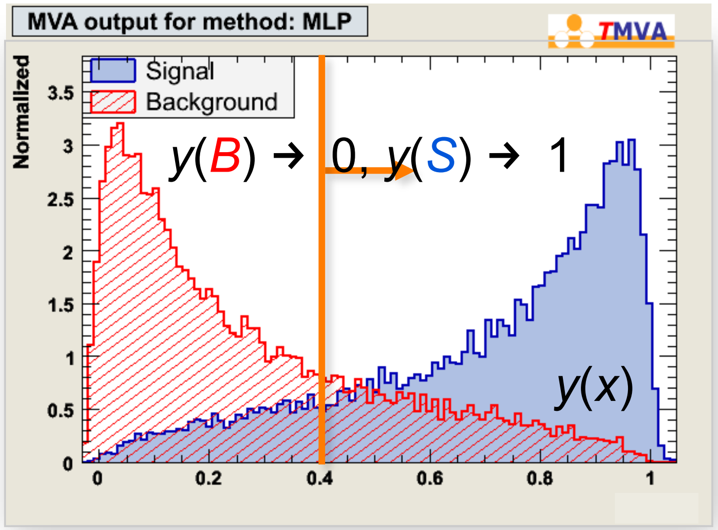




R^D

"feature space"

Each event, Signal or Background, has "*D*" measured variables.



$y(x)$: $R^n \rightarrow R$:

$\longrightarrow$ R

$y(x)$: "test statistic" in *D*-dimensional space of input variables

Distributions of $y(x)$: $\mathrm{PDF}_S(y)$ and $\mathrm{PDF}_B(y)$

Used to set the selection cut !

→ Efficiency and purity

$y(x)$: > cut: signal
= cut: decision boundary
< cut: background

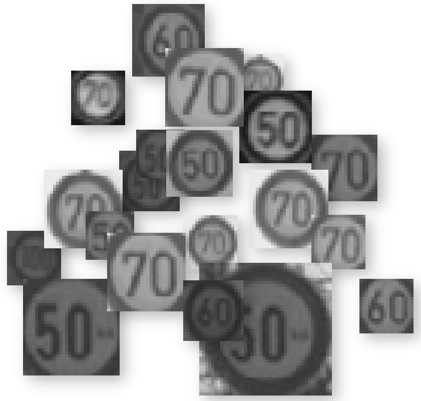
$y(x)$ = const: surface defining the decision boundary

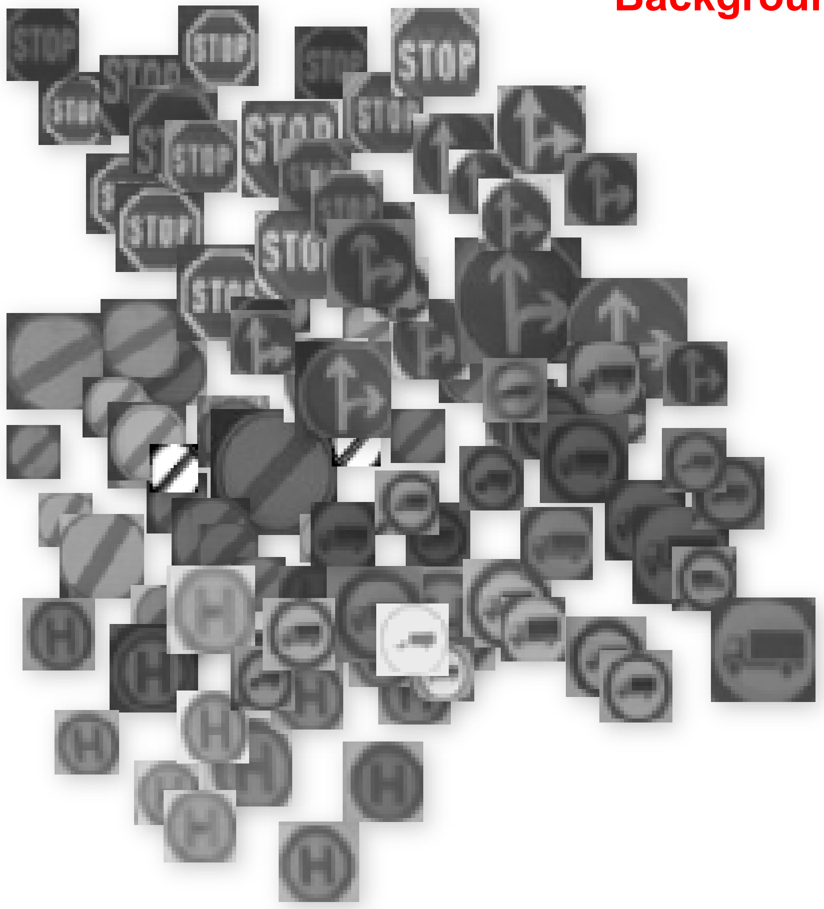
Overlap of $\mathrm{PDF}_S(y)$ and $\mathrm{PDF}_B(y)$ affects separation power, purity

Multi-Class Classification

Signal

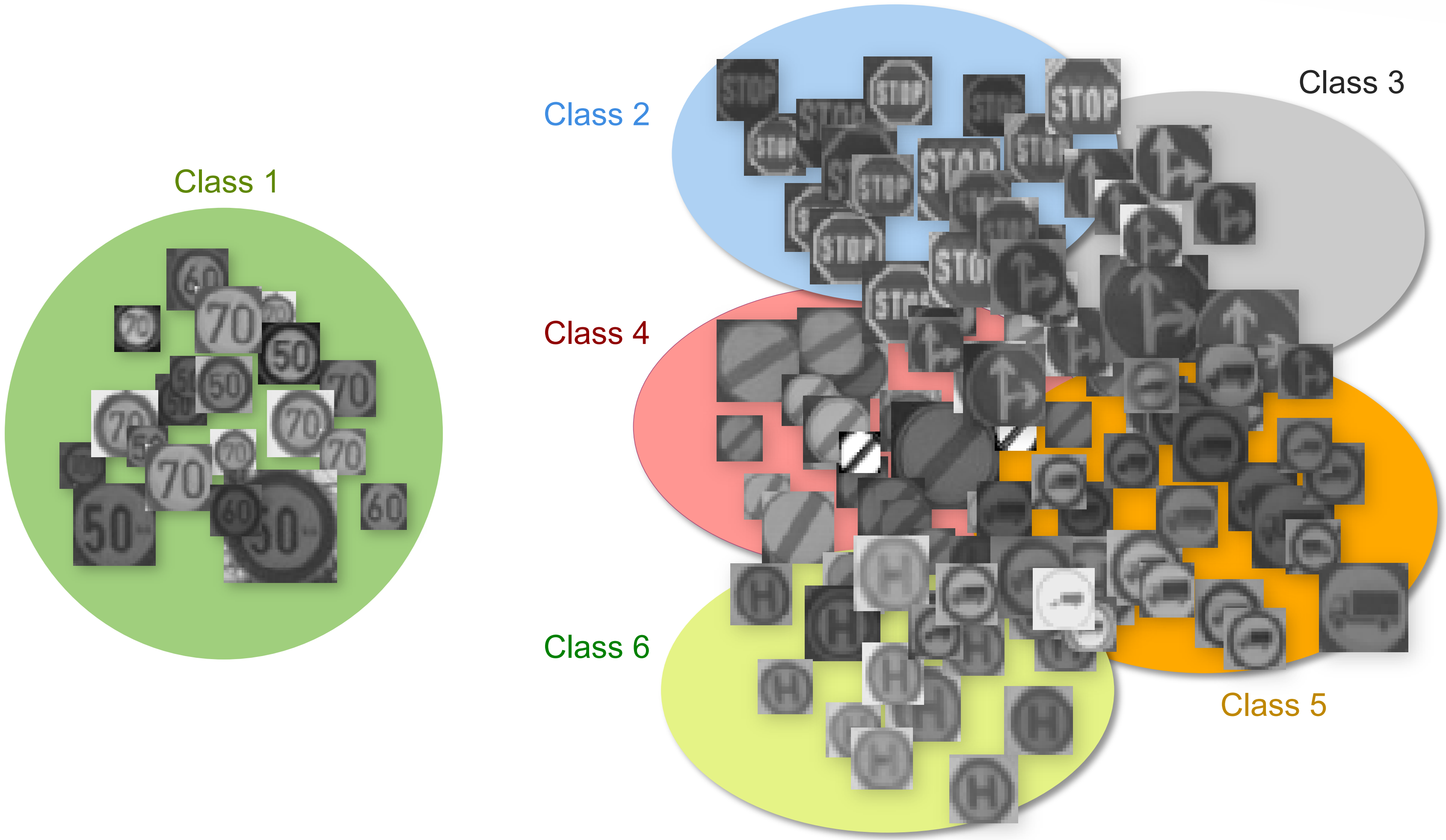
Background





Binary classification: two classes, “signal” and “background”

Multi-Class Classification



Multi-class classification – natural extension for many classifiers

Event Classification

P(Class=C|**x**) (or simply P(C|x)) : probability that the event class is of type C, given the measured observables $\mathbf{x} = \{x_1,\ldots,x_D\}$ à ￼y(**x**)

Probability density distribution according to the measurements **x** and the given mapping function

Prior probability to observe an event of "class C", *i.e.*, the relative abundance of "signal" versus "background"

$$P(\mathrm{Class} = C \mid y) = \frac{P(y \mid C) \cdot P(C)}{P(y)}$$

Posterior probability

Overall probability density to observe the actual measurement y(**x**), *i.e.*, $P(y) = \sum P(y \mid \mathrm{Class}) P(\mathrm{Class})$

Bayes Optimal Classification

$$P(\text{Class} = C \mid y) = \frac{P(y \mid C) P(C)}{P(y)}$$

$\mathbf{x} = \{x_1, \ldots, x_D\}$: measured observables
$y = y(\mathbf{x})$

AND

Minimum error in misclassification if C chosen such that it has maximum $P(C|\mathbf{y})$

→ To select S(ignal) over B(ackground), place decision on:

[Or any monotonic function of P(S|y) / P(B|y)]

$$\frac{P(S \mid y)}{P(B \mid y)} = \frac{P(y \mid S)}{P(y \mid B)} \cdot \frac{P(S)}{P(B)} > c$$

"c" determines efficiency and purity

Posterior odds ratio

Likelihood ratio as discriminating function y(x)

Prior odds ratio of choosing a signal event (relative probability of signal vs. bkg)

Any Decision Involves a Risk

Decide to treat an event as "Signal" or "Background"

Type-1 error:

Classify event as Class C even though it is not
(accept a hypothesis although it is not true)
(reject the null-hypothesis although it would have been the correct one)

→ loss of purity (in the selection of signal events)

Type-2 error:

Fail to identify an event from Class C as such
(reject a hypothesis although it would have been true)
(fail to reject the null-hypothesis/accept null hypothesis although it is false)

→ loss of efficiency (in selecting signal events)

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	J🏝	Type-2 error
Back-ground	Type-1 error	J🏝

"A": region where event is called signal

Significance α: Type-1 error rate:
(=p-value): α = background selection "efficiency"

$$\alpha = \int_A P(x \mid B)\,dx$$

should be small !

Size β: Type-2 error rate:
Power: 1- β = signal selection efficiency

$$\beta = \int_{!A} P(x \mid S)\,dx$$

should be small !

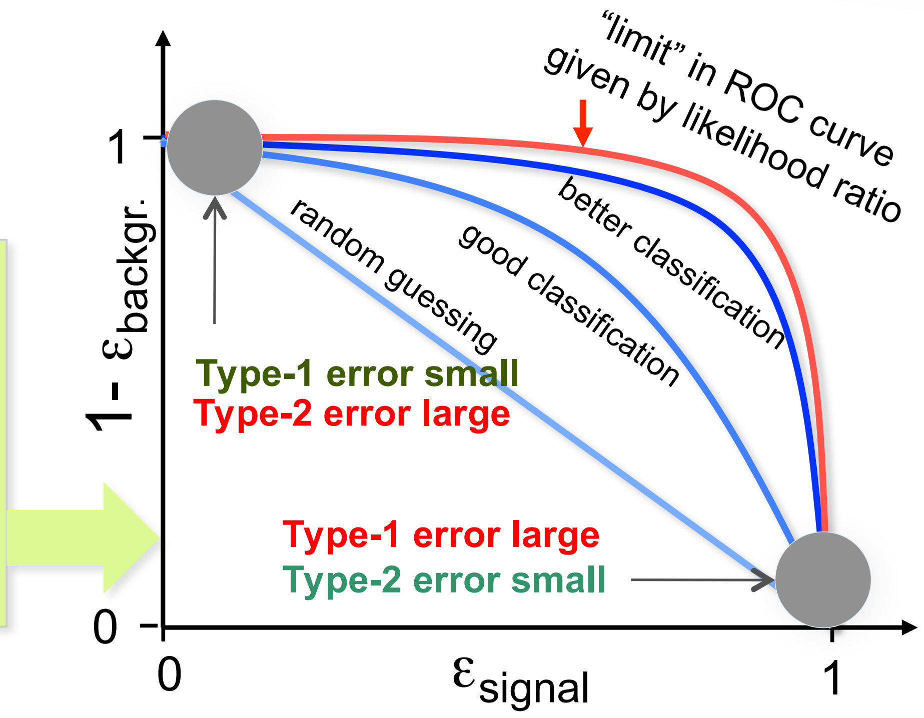
Neyman-Pearson Lemma

Likelihood Ratio : $y(x) = \dfrac{P(x|S)}{P(x|B)}$

Neyman-Peason:

The Likelihood ratio used as "selection criterion" y(x) gives for each selection efficiency the best possible background rejection.

i.e. it maximises the area under the "*Receiver Operation Characteristics*" (ROC) curve



→ Varying y(x) > "cut" moves the working point (efficiency and purity) along the ROC curve

- How to choose "cut"? → need to know prior probabilities (*S*, *B* abundances)
  - Measurement of signal cross section: maximum of $S/\sqrt{(S+B)}$ or equiv. $\sqrt{(\epsilon \cdot p)}$
  - Discovery of a signal : maximum of $S/\sqrt{(B)}$
  - Precision measurement: high purity (p)
  - Trigger selection: high efficiency (ϵ)*(sometimes high background rejection)*

Realistic Event Classification

Unfortunately, the true probability densities functions are typically unknown:
à Neyman-Pearson's lemma doesn't really help us…

Use MC simulation, or more generally: set of known (already classified) "events"

Use these "training" events to:

- Try to estimate the functional form of P(x|C) from which the likelihood ratio can be obtained
  e.g. D-dimensional histogram, Kernel densitiy estimators, MC-based matrix-element methods, …

- Find a "discrimination function" y(x) and corresponding decision boundary (i.e. hyperplane* in the "feature space": y(x) = const) that optimally separates signal from background
  e.g. Linear Discriminator, Neural Networks, Boosted Decision, …

→ Supervised (machine) learning

* Hyperplane in the strict sense goes through the origin. Here is meant an "affine set" to be precise.

Realistic Event Classification

Of course, there is no magic in here. We still need to:

- Choose the discriminating variables
- Choose the class of models (linear, non-linear, flexible or less flexible)
- Tune the "learning parameters" → bias vs. variance trade off
- Check the generalisation properties (avoid overtraining)
- Consider trade off between statistical and systematic uncertainties

from background

e.g. Linear Discriminator, Neural Networks, …

Multivariate Analysis Methods in *T*MVA

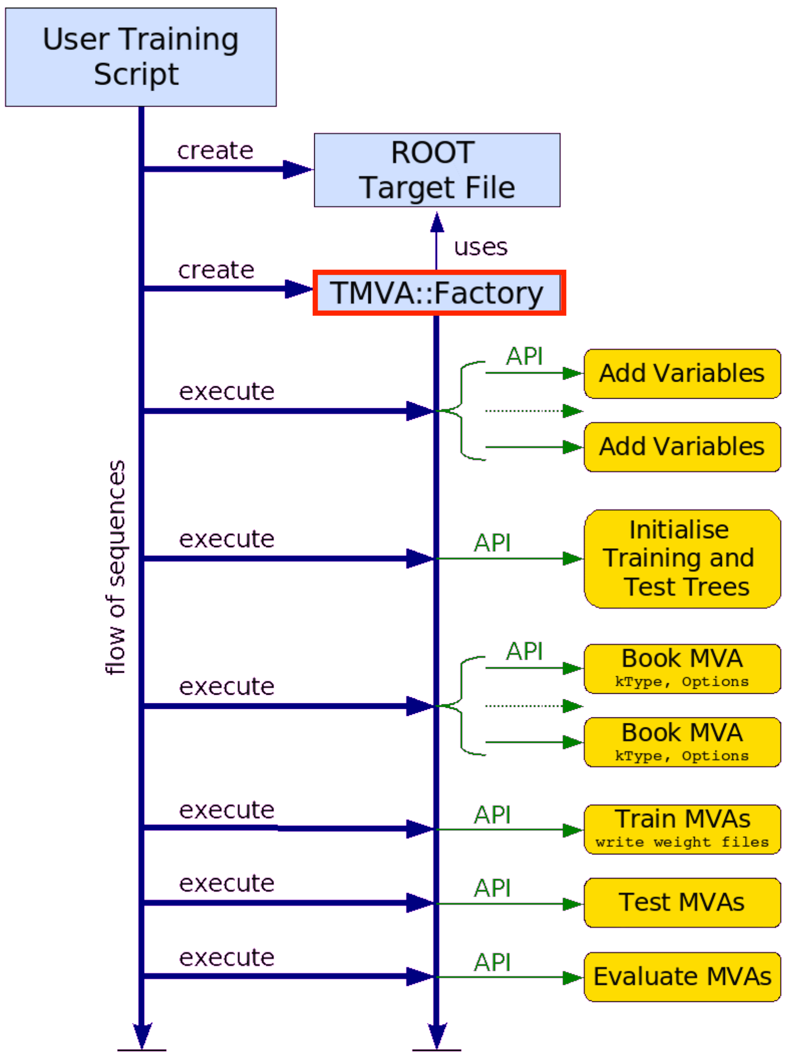
- Examples for classifiers and regression methods
  - Rectangular cut optimisation
  - Projective and multidimensional likelihood estimator
  - k-Nearest Neighbor algorithm
  - Fisher and H-Matrix discriminants
  - Function discriminants
  - Artificial neural networks
  - Boosted decision trees
  - RuleFit
  - Support Vector Machine
- Preprocessing methods:
  - Decorrelation, Principal Value Decomposition, Gaussianisation
- Examples for synthesis methods:
  - Boosting, Categorisation(valid for all methods, and their combinations)

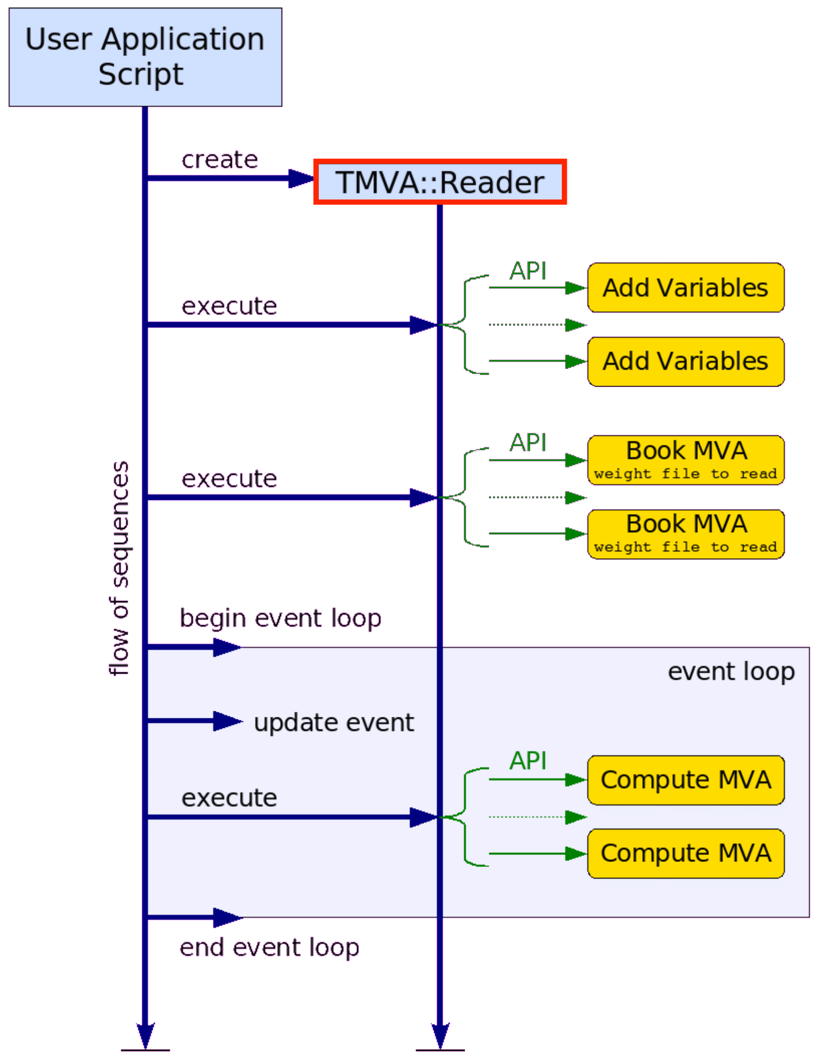
Using *T*MVA

A typical *T*MVA analysis consists of two main steps:

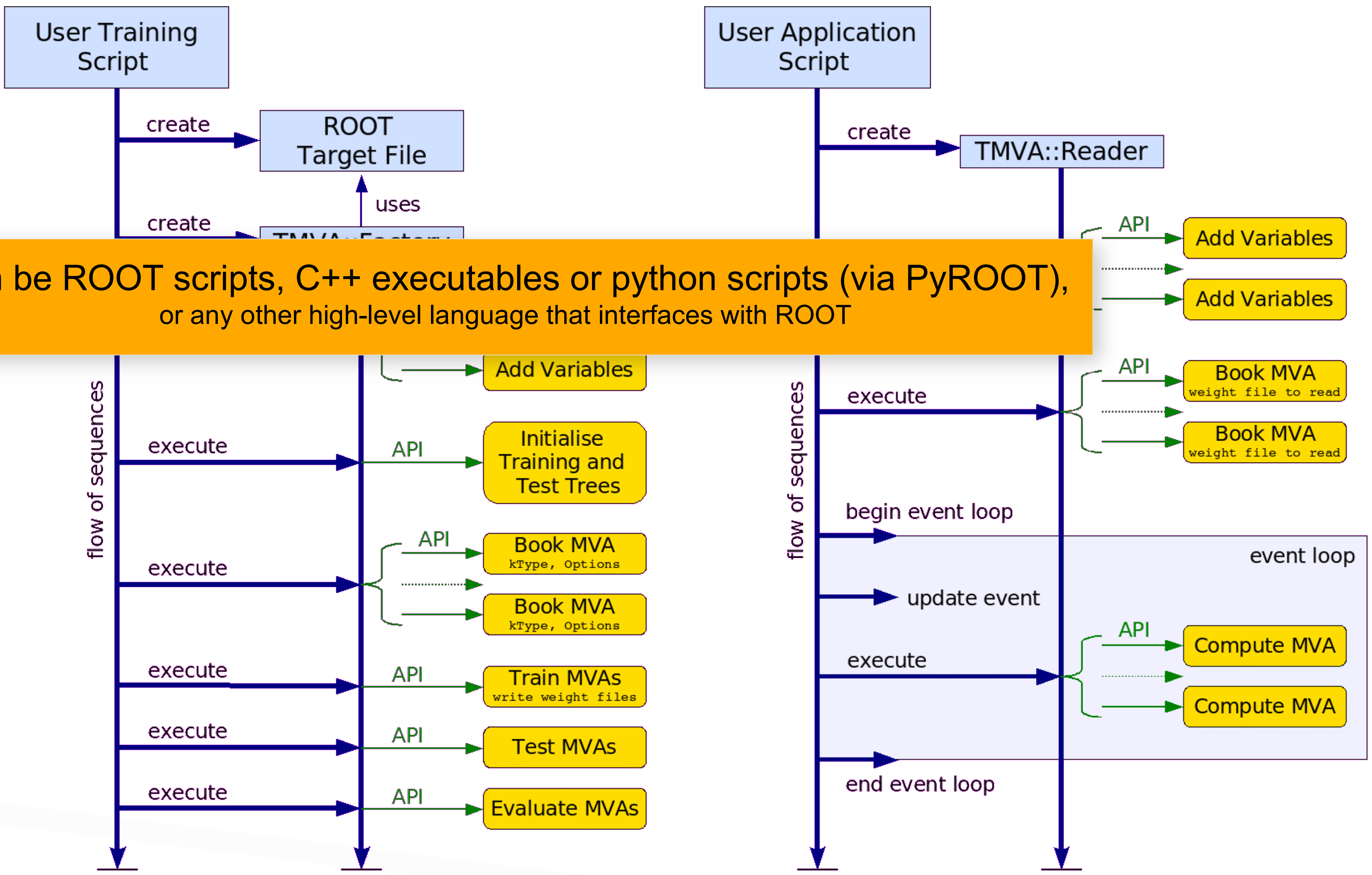
1. ***Training phase***: training, testing and evaluation of classifiers using data samples with known signal and background composition
2. ***Application phase***: using selected trained classifiers to classify unknown data samples

Code Flow for Training and Application



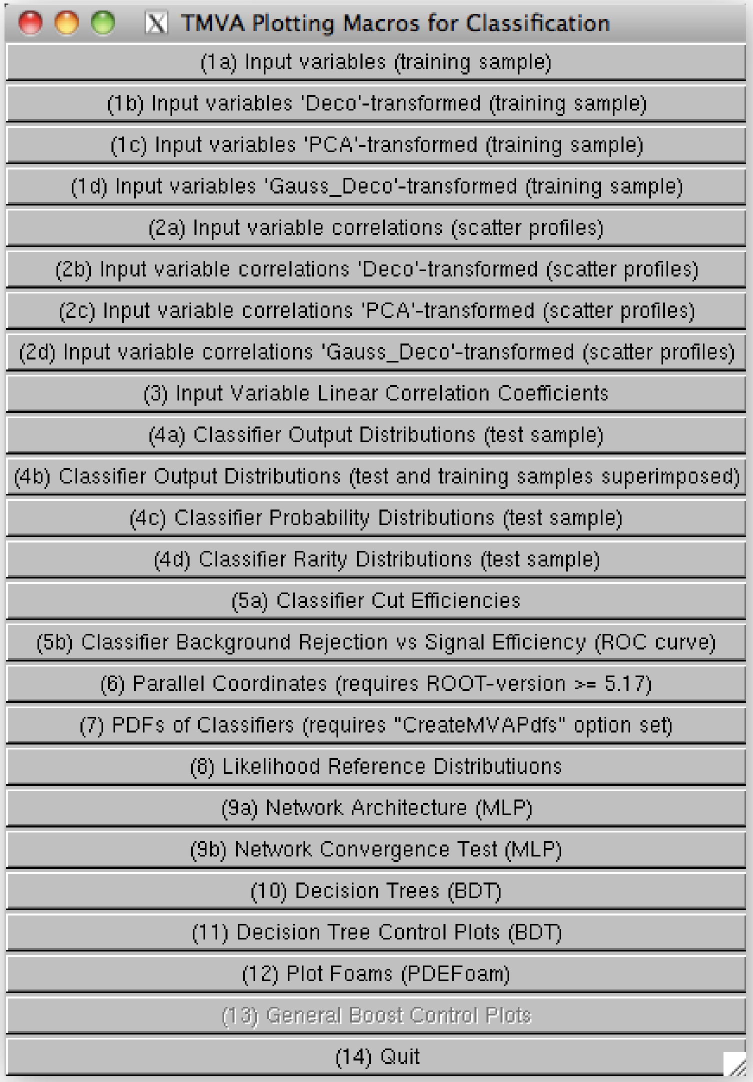


Code Flow for Training and Application



Can be ROOT scripts, C++ executables or python scripts (via PyROOT), or any other high-level language that interfaces with ROOT

Strong Methods need Strong Evaluation

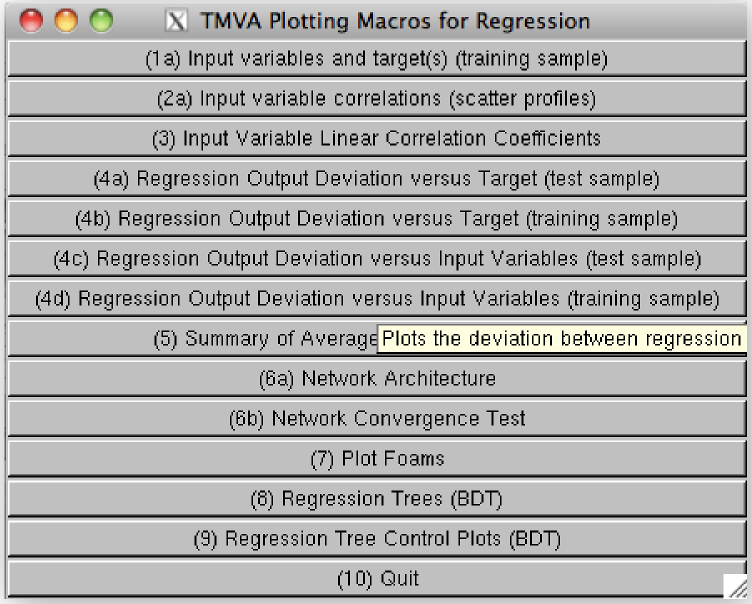


TMVA Plotting Macros for Classification

(1a) Input variables (training sample)
(1b) Input variables 'Deco'-transformed (training sample)
(1c) Input variables 'PCA'-transformed (training sample)
(1d) Input variables 'Gauss_Deco'-transformed (training sample)
(2a) Input variable correlations (scatter profiles)
(2b) Input variable correlations 'Deco'-transformed (scatter profiles)
(2c) Input variable correlations 'PCA'-transformed (scatter profiles)
(2d) Input variable correlations 'Gauss_Deco'-transformed (scatter profiles)
(3) Input Variable Linear Correlation Coefficients
(4a) Classifier Output Distributions (test sample)
(4b) Classifier Output Distributions (test and training samples superimposed)
(4c) Classifier Probability Distributions (test sample)
(4d) Classifier Rarity Distributions (test sample)
(5a) Classifier Cut Efficiencies
(5b) Classifier Background Rejection vs Signal Efficiency (ROC curve)
(6) Parallel Coordinates (requires ROOT-version >= 5.17)
(7) PDFs of Classifiers (requires "CreateMVAPdfs" option set)
(8) Likelihood Reference Distributiuons
(9a) Network Architecture (MLP)
(9b) Network Convergence Test (MLP)
(10) Decision Trees (BDT)
(11) Decision Tree Control Plots (BDT)
(12) Plot Foams (PDEFoam)
(13) General Boost Control Plots
(14) Quit

A lot of evaluation information is already provided in the logging output of the training

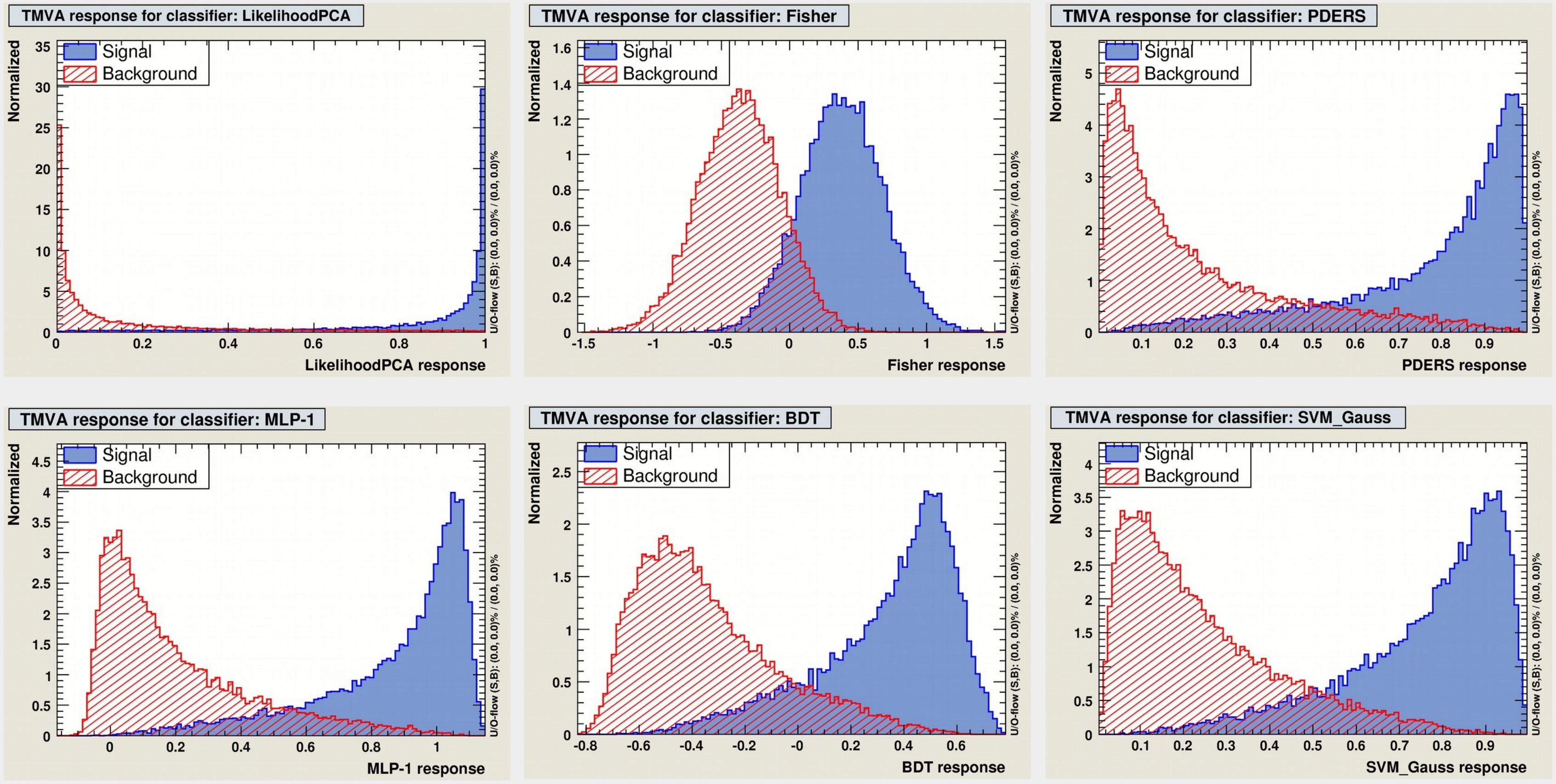
Simple GUIs provide access to evaluation plots and tools for single and multi-class classification and regression



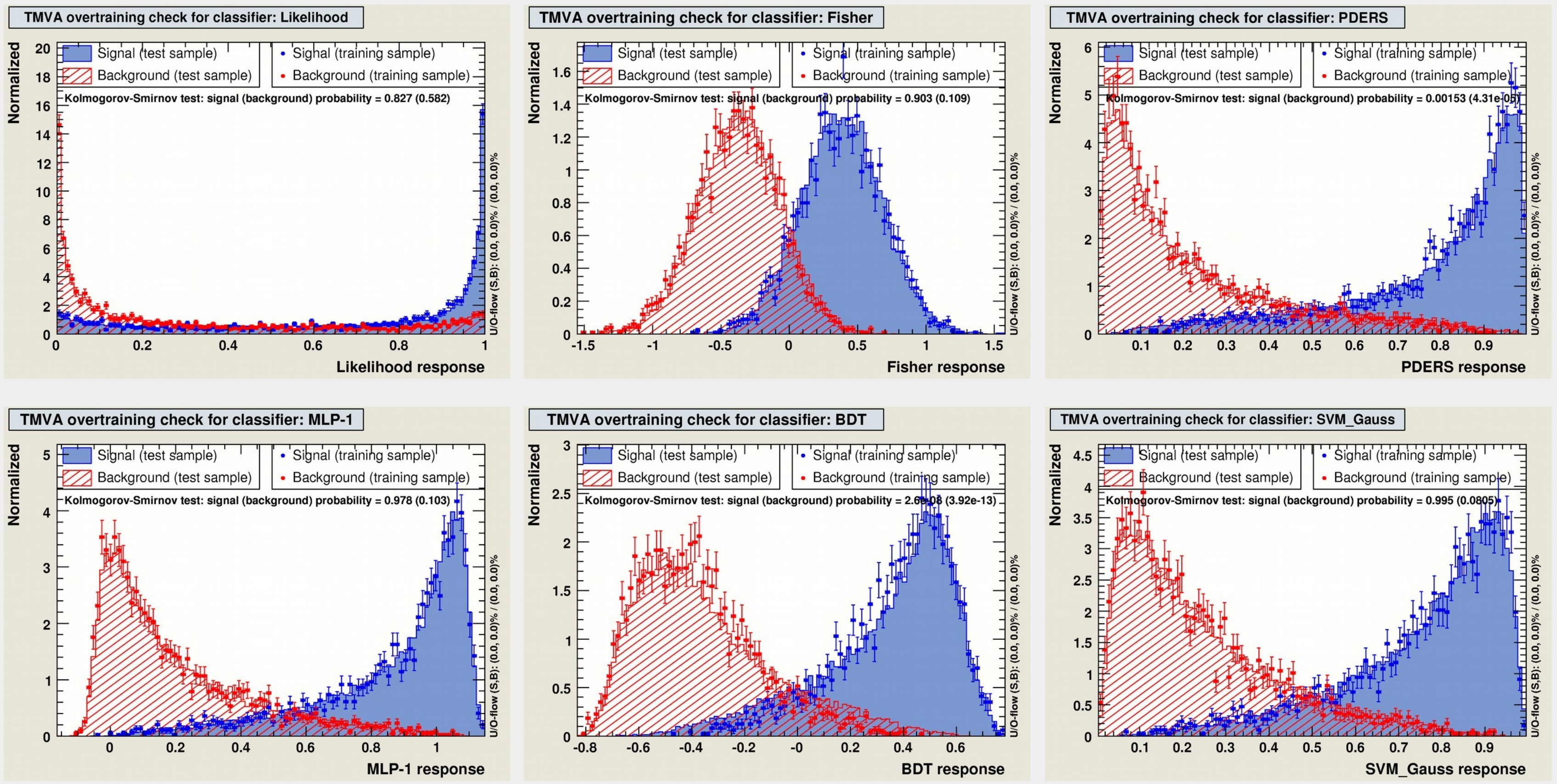
TMVA Plotting Macros for Regression

(1a) Input variables and target(s) (training sample)
(2a) Input variable correlations (scatter profiles)
(3) Input Variable Linear Correlation Coefficients
(4a) Regression Output Deviation versus Target (test sample)
(4b) Regression Output Deviation versus Target (training sample)
(4c) Regression Output Deviation versus Input Variables (test sample)
(4d) Regression Output Deviation versus Input Variables (training sample)
(5) Summary of Average Plots the deviation between regression
(6a) Network Architecture
(6b) Network Convergence Test
(7) Plot Foams
(8) Regression Trees (BDT)
(9) Regression Tree Control Plots (BDT)
(10) Quit

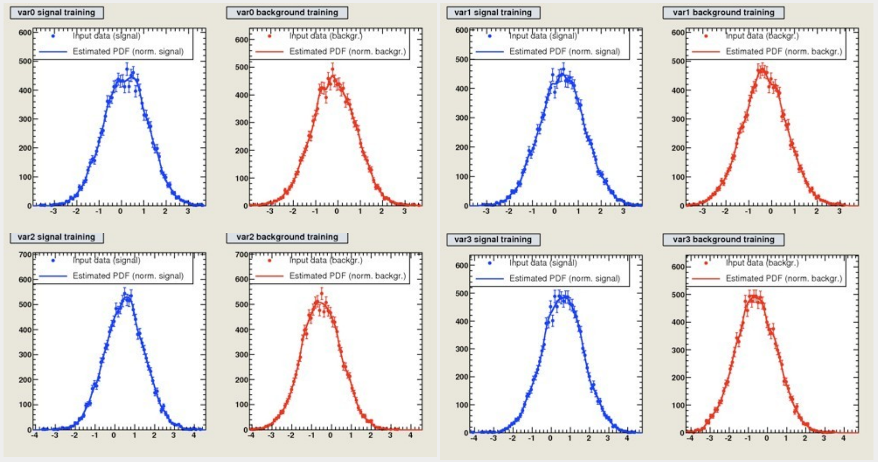
Involved Methods need Thorough Evaluation

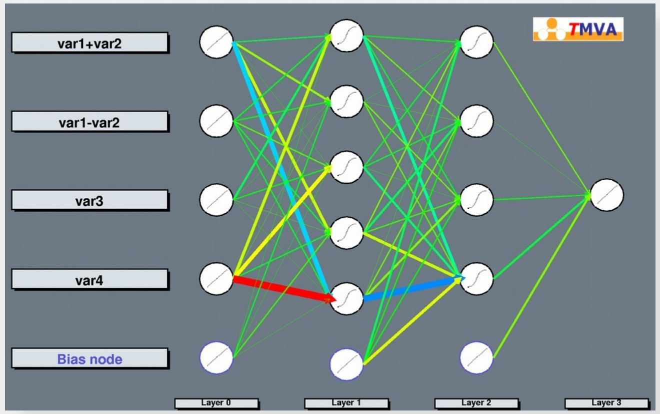


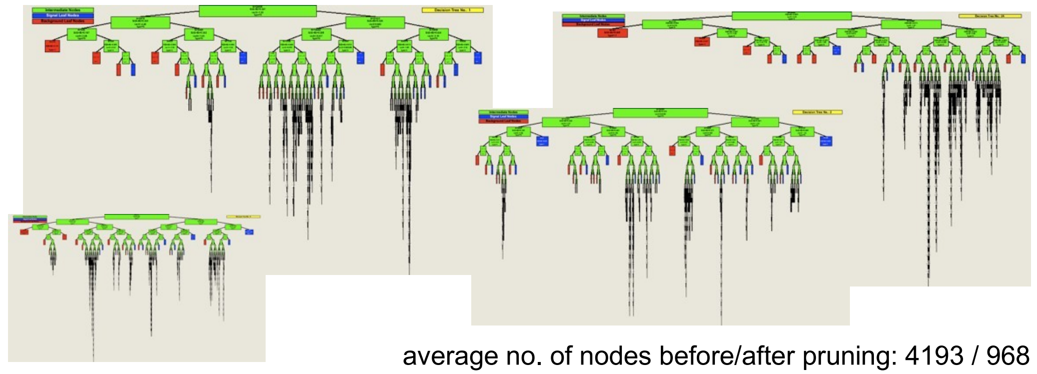
Involved Methods need Thorough Evaluation



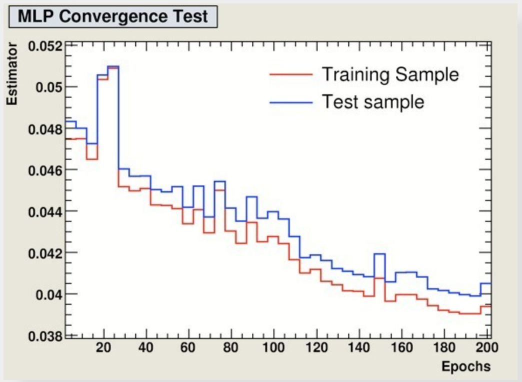
Involved Methods need Thorough Evaluation







average no. of nodes before/after pruning: 4193 / 968



Practical Tips and Tricks for TMVA Users

From Eckhard von Toerne University of Bonn

A Closer Look
at
Input Data

General data properties

- Variables may be statistically (un-)correlated
- Signal and/or Background may cover full volume, partial volume, or are only found on hypersurfaces.
- Variables may have spikes, steps, tails, poles
- One or many connected regions
- Number of variables

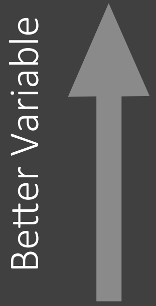
  → beware of **"curse of dimensionality"**

How to …
choose input variables

Evaluating the Classifiers

(taken from TMVA output…)

Input Variable Ranking



```
--- Fisher         : Ranking result (top variable is best ranked)
--- Fisher         : ---------------------------------------------
--- Fisher         : Rank : Variable   : Discr. power
--- Fisher         : ---------------------------------------------
--- Fisher         :    1 : var4       : 2.175e-01
--- Fisher         :    2 : var3       : 1.718e-01
--- Fisher         :    3 : var1       : 9.549e-02
--- Fisher         :    4 : var2       : 2.841e-02
--- Fisher         : ---------------------------------------------
```

- How discriminating is a variable ?

Classifier correlation and overlap

```
--- Factory        : Inter-MVA overlap matrix (signal):
--- Factory        : ------------------------------
--- Factory        :              Likelihood  Fisher
--- Factory        : Likelihood:      +1.000  +0.667
--- Factory        :     Fisher:      +0.667  +1.000
--- Factory        : ------------------------------
```

- Do classifiers select the same events as signal and background ? If not, there is something to gain !

How to …
choose the multivariate method

Basis of our choice

How large is the training sample and how many variables contain useful information?

Number of parameter that define the method needs to be smaller than data size.

For most classifiers the number of employed „parameters“ may be chosen by user:

Examples:

How large are correlations among variables

How conservative is the E.B.?

Choice of MVA methods

- Number of „parameters“ is limited due to small data sample

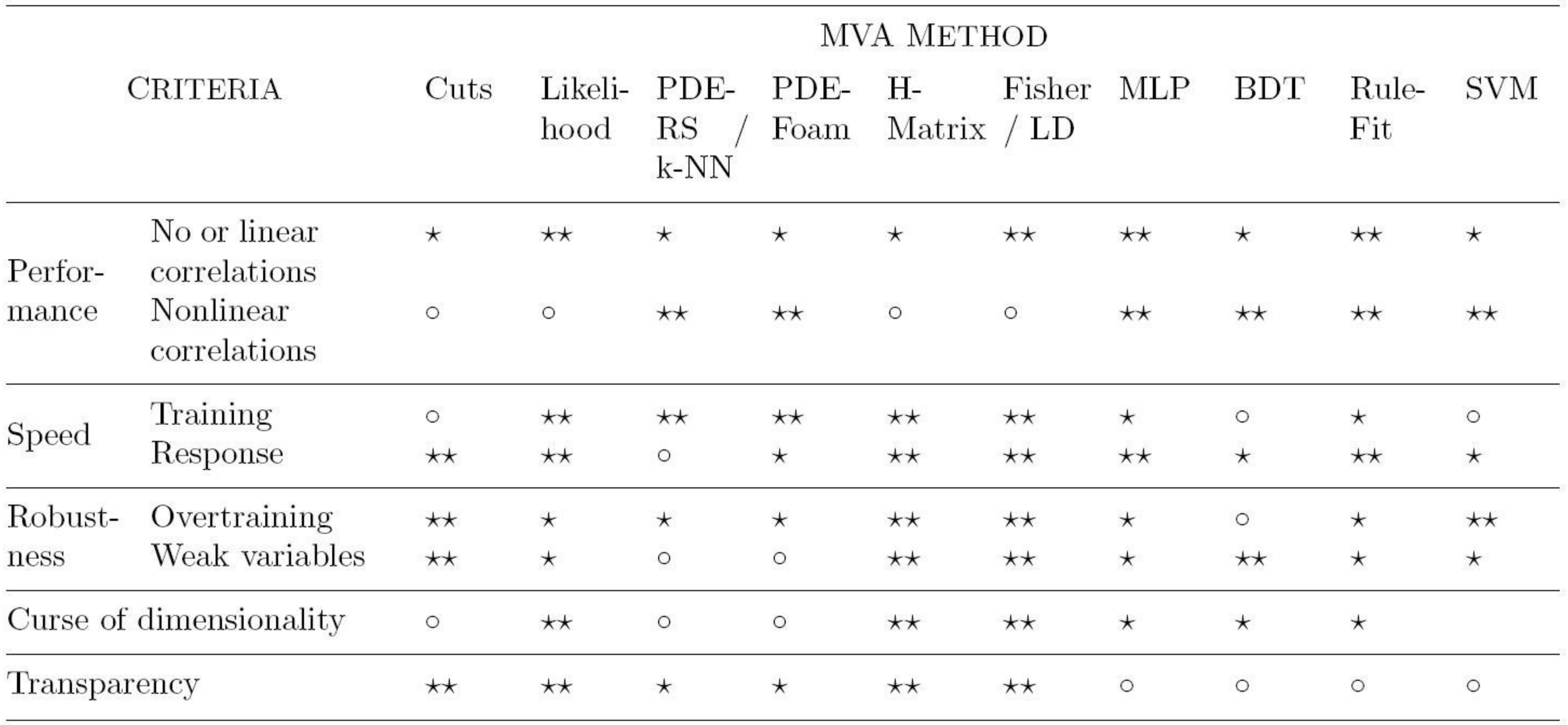
→ Use Linear classifier or FDA, small BDT (small MLP)

- Variables are uncorrelated (or only linear corrs) → likelihood
- I just want something simple → Cuts, LD, Fisher
- Methods that usually work out of the box, even for complex problems → BDT, MLP, SVM

List of acronyms:

BDT = boosted decision tree, see manual page 103
ANN = articifical neural network
MLP = multi-layer perceptron, a specific form of ANN, also the name of our flagship ANN, manual p. 92
FDA = functional discriminant analysis, see manual p. 87
LD = linear discriminant , manual p. 85
SVM = support vector machine, manual p. 98 , SVM currently available only for classification
Cuts = like in “cut selection“, manual p. 56
Fisher = Ronald A. Fisher, classifier similar to LD, manual p. 83

Summary



CRITERIA		MVA METHOD									
		Cuts	Likeli-hood	PDE-RS / k-NN	PDE-Foam	H-Matrix	Fisher / LD	MLP	BDT	Rule-Fit	SVM
Perfor-mance	No or linear correlations	⋆	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
	Nonlinear correlations	○	○	⋆⋆	⋆⋆	○	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆
Speed	Training	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆	○	⋆	○
	Response	⋆⋆	⋆⋆	○	⋆	⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
Robust-ness	Overtraining	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	○	⋆	⋆⋆
	Weak variables	⋆⋆	⋆	○	○	⋆⋆	⋆⋆	⋆	⋆⋆	⋆	⋆
Curse of dimensionality		○	⋆⋆	○	○	⋆⋆	⋆⋆	⋆	⋆	⋆	
Transparency		⋆⋆	⋆⋆	⋆	⋆	⋆⋆	⋆⋆	○	○	○	○

Table 6: Assessment of MVA method properties. The symbols stand for the attributes "good" (⋆⋆), "fair" (⋆) and "bad" (○). "Curse of dimensionality" refers to the "burden" of required increase in training statistics and processing time when adding more input variables. See also comments in the text. The FDA method is not listed here since its properties depend on the chosen function.

From the TMVA manual, chapter 10.

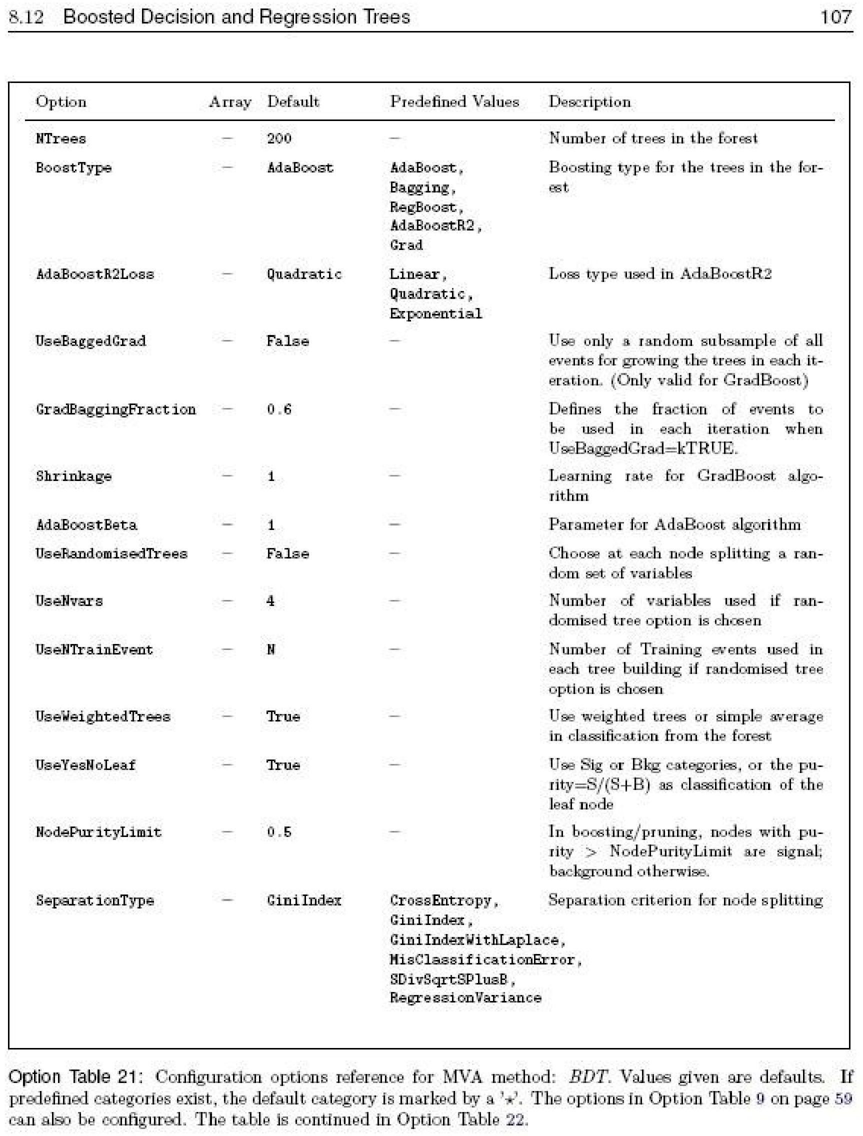
Customizing the method via the option string

BDT option table (from manual)

- Method booking

factory->BookMethod(
TMVA::Types::kBDT, “myBDT",
“**BoostType=Grad:SeparationType=GiniIndex:Ntrees=500**“);

- Read description of method in the manual.
- Choose the number of defining parameters according to data size and number of variables.



8.12 Boosted Decision and Regression Trees 107

Option	Array	Default	Predefined Values	Description
NTrees	–	200	–	Number of trees in the forest
BoostType	–	AdaBoost	AdaBoost, Bagging, RegBoost, AdaBoostR2, Grad	Boosting type for the trees in the forest
AdaBoostR2Loss	–	Quadratic	Linear, Quadratic, Exponential	Loss type used in AdaBoostR2
UseBaggedGrad	–	False	–	Use only a random subsample of all events for growing the trees in each iteration. (Only valid for GradBoost)
GradBaggingFraction	–	0.6	–	Defines the fraction of events to be used in each iteration when UseBaggedGrad=kTRUE.
Shrinkage	–	1	–	Learning rate for GradBoost algorithm
AdaBoostBeta	–	1	–	Parameter for AdaBoost algorithm
UseRandomisedTrees	–	False	–	Choose at each node splitting a random set of variables
UseNvars	–	4	–	Number of variables used if randomised tree option is chosen
UseNTrainEvent	–	N	–	Number of Training events used in each tree building if randomised tree option is chosen
UseWeightedTrees	–	True	–	Use weighted trees or simple average in classification from the forest
UseYesNoLeaf	–	True	–	Use Sig or Bkg categories, or the purity=S/(S+B) as classification of the leaf node
NodePurityLimit	–	0.5	–	In boosting/pruning, nodes with purity > NodePurityLimit are signal; background otherwise.
SeparationType	–	GiniIndex	CrossEntropy, GiniIndex, GiniIndexWithLaplace, MisClassificationError, SDivSqrtSPlusB, RegressionVariance	Separation criterion for node splitting

Option Table 21: Configuration options reference for MVA method: *BDT*. Values given are defaults. If predefined categories exist, the default category is marked by a '⋆'. The options in Option Table 9 on page 59 can also be configured. The table is continued in Option Table 22.

How to obtain signal and background samples for training

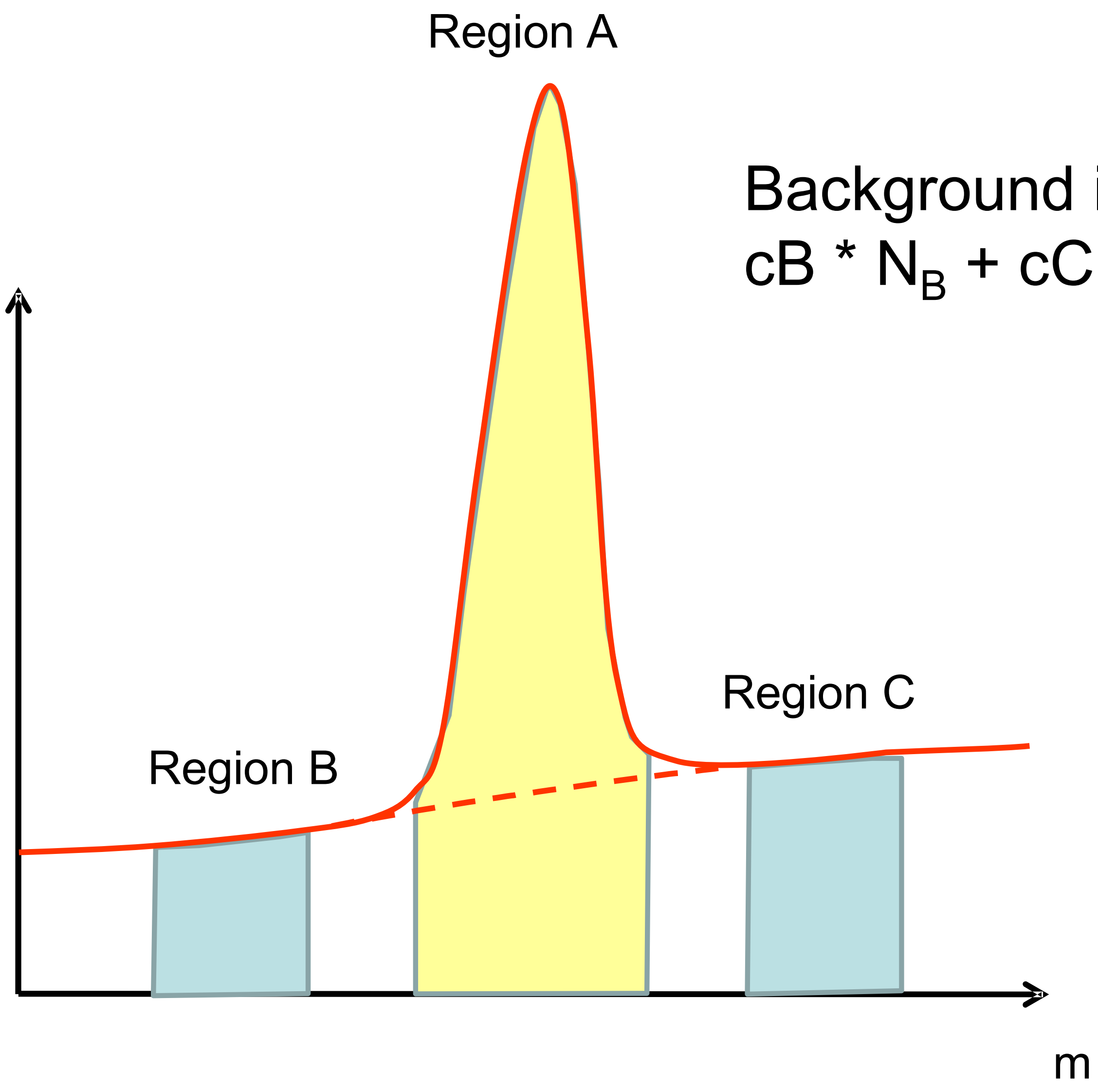
Signal and background samples for training

- What works for a counting analysis usually works for a MVA too.
- Examples:
  - Monte Carlo
  - Sidebands (also ABCD method)
  - Event Crossing

$\}$ works with data

Example Analysis sideband method with TMVA

Sideband method with TMVA



Background in Region A = cB * N_B + cC * N_C

A complete TMVA training/testing session

```
void TMVAnalysis( )
{
  TFile* outputFile = TFile::Open( "TMVA.root", "RECREATE" );

  TMVA::Factory *factory = new TMVA::Factory( "MVAnalysis", outputFile,"!V");

  TFile *input = TFile::Open("tmva_example.root");

  factory->AddVariable("var1+var2", 'F');
  factory->AddVariable("var1-var2", 'F');   //factory->AddTarget("tarval", 'F');

  TTree* dataTree = (TTree*) input->Get("TreeS");
  double coeffA = 1.0, coeffB = 0.34 coeffC = …; //set coefficients
  factory->AddTree (dataTree, “Signal”,            1.,         “m> signalLow && m<signalHigh”);  // Region A
  factory->AddTree (dataTree, “Background”, weightB, “m> bg1Low && m<bg1High”); // Region B
  factory->AddTree (dataTree, “Background”, weightC, “m> bg2Low && m<bg2High”); // Region C

  factory->PrepareTrainingAndTestTree( "", "",  "NormMode=None");

  factory->BookMethod( TMVA::Types::kMLP, "MLP",
  "!V:NCycles=200:HiddenLayers=N+1,N:TestRate=5" );

  factory->TrainAllMethods();
  factory->TestAllMethods();
  factory->EvaluateAllMethods();
  outputFile->Close();
  delete factory;
}
```

Create Factory

Add variables/ targets

Initialize Trees

Book MVA methods

Train, test and evaluate

How to ...
employ trained classifiers

Using the Reader (recommended)

```
#include "TMVA/Reader.h„
…
 TMVA::Reader* reader = new TMVA::Reader( “Verbose“ );  // “Silent“ to turn off log-outp.
```

Create Reader

```
reader->AddVariable(“var1“, &var1);   // add variables in same order as in training, pass all vars as floats
reader->AddVariable(“var2“, &var2);
reader->BookMVA(“BDT method“, „weights/weightfilename.xml“);
```

Add variables, book method

```
//Enter loop over all events
//Fill variables var1 and var2 with current values
float mvavalue =reader->EvaluateMVA( “BDT method“,);
```

Obtain MVA value for one event

Alternatively, pass all variables to reader as a vector of floats

```
Std::vector<float> vec(2);
TMVA::Reader* reader = new TMVA::Reader( “Verbose“ );  // “Silent“ to turn off log-outp.
reader->BookMVA(“BDT method“, „weights/weightfilename.xml“);

//Enter loop over all events
//Fill variables vector with current values
vec[0]=…;
vec[1]= …;
float mvavalue =reader->EvaluateMVA( vec, “BDT method“);
```

Important: pass all variables to Reader as floats!

Using the test tree (Q&D hack)

training and evaluation yields output root file with the results of the training and test

For a quick (and “dirty“) analysis the user might use the test tree TMVA.root:TestTree

Contents of the tree:

```
root [2] TestTree->Print()
******************************************************
*Tree    :TestTree : TestTree
*Entries :      165 : Total =          16578 bytes
******************************************************
*Br    0 :classID : classID/I                     ( ID=0 signal, ID=1, background)
*Br    1 :className : className/C                 ( className “Signal“ or “Background“  )
*Br    2 :var0      : var0/F                      ( the list of input variables)
*Br    3 :var1      : var1/F                           …..
*Br   10 :weight : weight/F                       (the training weight, this is the original weight *
                                                      renormalization factor)
*Br   11 :LD1     : LD1/F                         (The MVA output of the method named LD1 )
```

+ additional quantities (“spectators“) defined via factory->AddSpectator

If you want to use tree entry “weight“ as a lumi-weighted MC weight, either pass weight on as a spectator or trun off weight-renormalisation by setting „“NormEvents=None“ in the training session, using factory->PrepareTrainingAndTestTree("", "","NormMode=None");

Esercitazione 14

- Copy the TMVA tutorial cp-r $ROOTSYS/tutorials/tmva mytmva
- Go to mytmva directory and open the file TMVAClassification.C
- Download the
  - TMVA.tree.REAL2011.210.root
  - TMVA.treeSignalMC.root
  - Inside each root file there are some tree. We are interested on these tree:
    - TMVA.treeSignalMC.root : “TreeSignal”
    - TMVA.tree.REAL2011.210.root : “TreeLSB”
- Select the variables you want to select (start from tdcaTracks, tCosPointingAngle, tMomHe, tMomPi, tDecayLenght, tDecayLenght; add tMass as a spectator variable)
- Switch on only Likelihood, MLP and BDT.
- Set to 0.1 the signal weight: Double_t signalWeight = 0.1;
- Run the MVA → A TMVA.root should be there
- Now let’s have a look a the different histograms stored in TMVA.root using TMVACrossValidation.C
- Alternatively inside a root session you can type:
  - TMVA::TMVAGui("TMVA.root")
- And the GUI should pop up.
- Questions
  - Which method gives the best performance?
  - Play with the weight of training events, how does it effect the training?
  - Finally, switch on other methods (not all), which one is the best?

Esercitazione 14

- Choose a value of the discrimination “variable”

Second part :

Application:

- What we want do now is apply the “training” to the “real” data. This can be done using the TMVAClassification.C macro
- In the reader set the same variables you used for the training → Inside TMVA.root those are the only stored variables
- Loop on the “signal+background” tree to see how the invariant mass stored in the tree changes by changing the “discrimination variable” → i.e. store the IM in a histogram
- Plot the histogram in the same canvas to see which is the “best” selection