Progress on Multivariate Data Analysis with *T*MVA

Practical Tips and Tricks for TMVA Users

Eckhard von Toerne
University of Bonn

universitätbonn

Contents

- A closer look at input data
- How to ...
  - choose input variables
  - choose a suitable MVA method
- Analysis setup for training
- How to employ trained classifiers
- Information regarding tutorial

  Website: http://www.uni-bonn.de/~etoerne/tmva/

universitätbonn

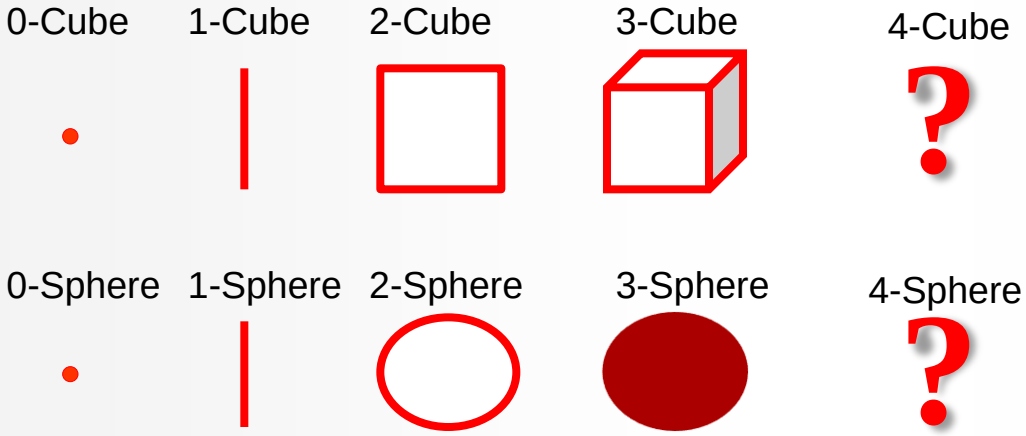
A Closer Look at Input Data

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“**

Curse of dimensionality: N-cube vs. N-Sphere

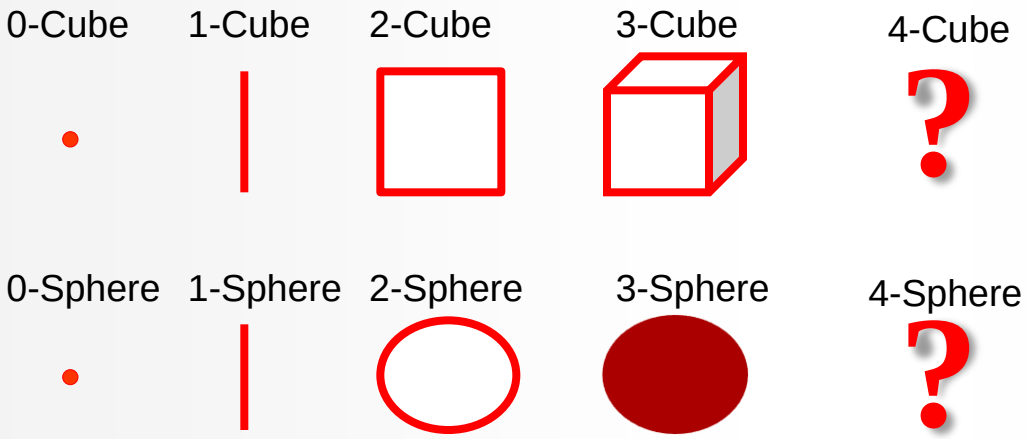


Volume of N-cube and N-sphere

- V(N-cube) $= R^N$ (R=side length)
- V(N-Sphere) $= c_N * R^N$ (R=radius)
- $c_2=\pi$, $c_3=4/3 * \pi$, $c_4=4.95$, $c_5=5.21$

So, what is $\lim_{N\to inf} c_N$?

Curse of dimensionality: N-cube vs. N-Sphere

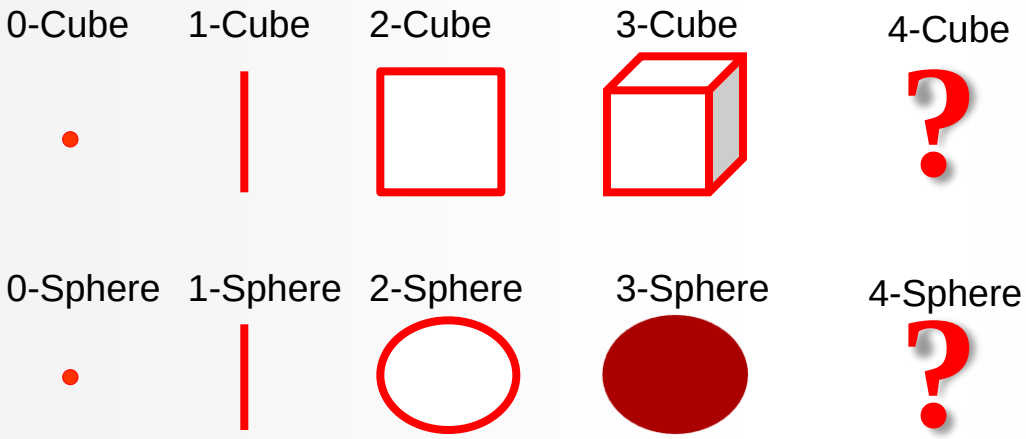


Volume of N-cube and N-sphere

- V(N-cube) $= R^N$ (R=side length)
- V(N-Sphere) $= c_N * R^N$ (R=radius)
- $c_2=\pi$, $c_3=4/3 * \pi$, $c_4=4.95$, $c_5=5.21$, $c_6 = 5.10$, $c_7 = 4.65$, $c_8=3.96$

So, what is $\lim_{N\to inf} c_N$?

Curse of dimensionality: N-cube vs. N-Sphere



Volume of N-cube and N-sphere

- V(N-cube) $= R^N$ (R=side length)
- V(N-Sphere) $= c_N * R^N$ (R=radius)
- $c_2=\pi$, $c_3=4/3 * \pi$, $c_4=4.95$, $c_5=5.21$, $c_6 = 5.10$, $c_7 = 4.65$, $c_8=3.96$

So, what is $\lim_{N\to inf} c_N = 0$

Most of the Volume of the N-sphere is in its outermost shell:

$dV \sim r^{N-1} dr$

universitätbonn

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 !

universitätbonn

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

- 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.

BDT option table (from manual)

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.

universitätbonn

How to obtain signal and background samples for training

Signal and background samples for training

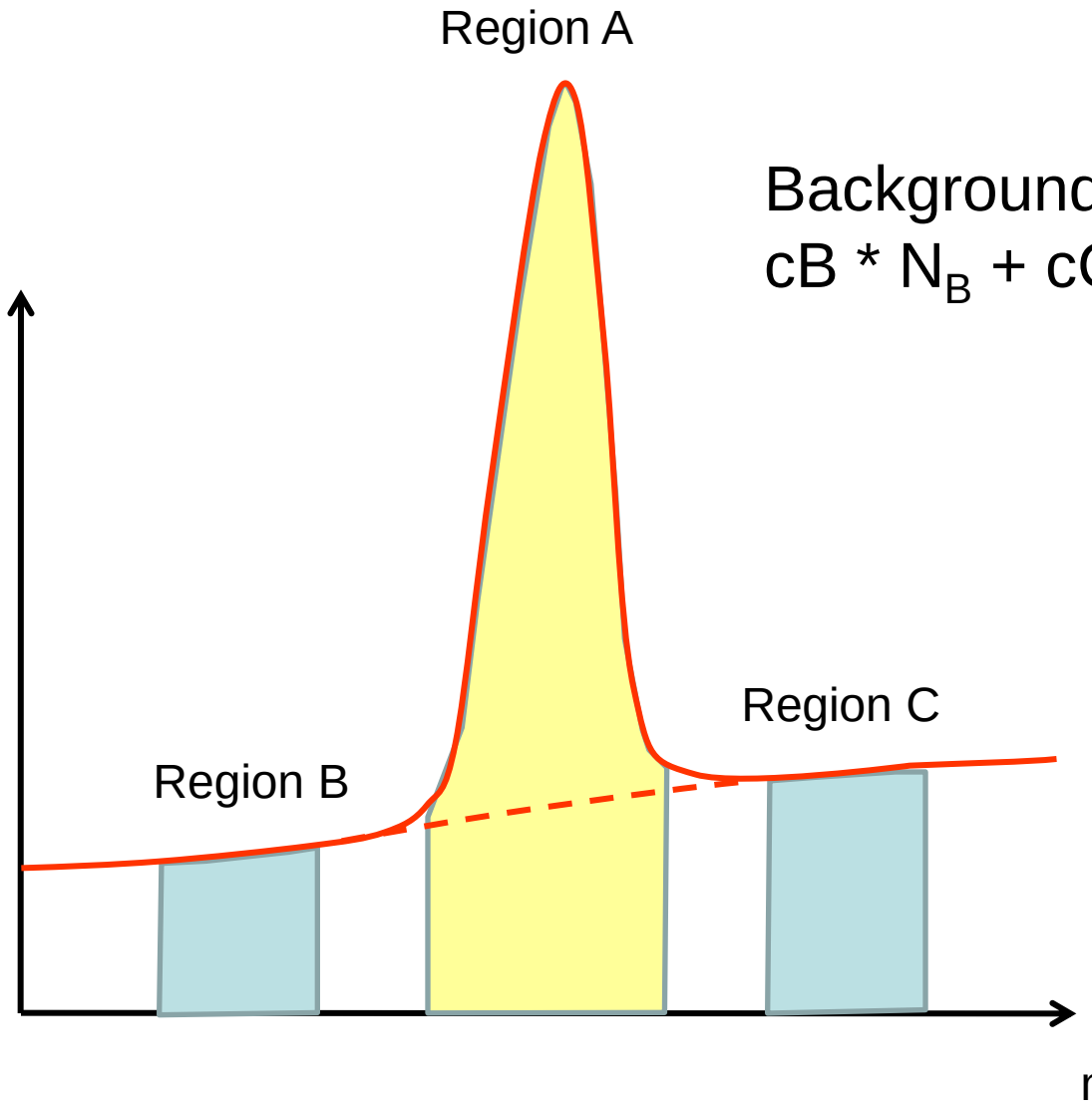
- What works for a counting analysis usually works for a MVA too.
- Examples:
  - Monte Carlo
  - Sidebands (also ABCD method)
  - Event Crossing

} works with data

universitätbonn

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

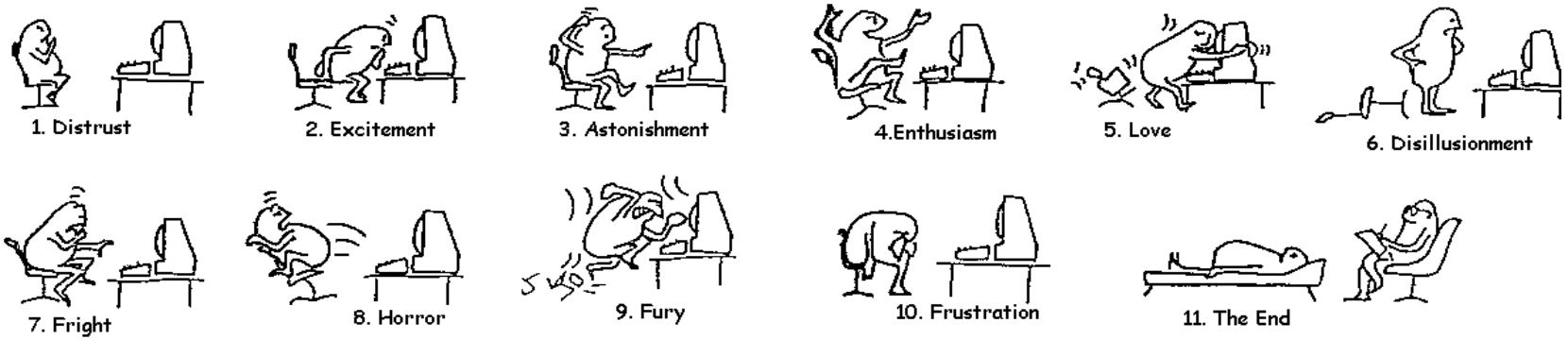
Initialize Trees

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.

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“);

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

Create Reader

Add variables, book 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");

universitätbonn

The tutorial

WHEN 15:30 – 18:00
WHERE: 60-6-015 (HERE)

Preparation

On Fri, Jan 14, 2011, Eckhard von Toerne evt@physik.uni-bonn.de wrote:

Dear all,
here is some additional information regarding the tutorial
at the TMVA workshop next week.

TMVA Tutorial, Jan 21, 15:30 - 18:00, CERN Room: 60-6-015

* Workshop web page:
http://indico.cern.ch/event/tmva_workshop

* Prerequisites for tutorial: Participants are kindly
requested to bring a notebook with ROOT, Version >=5.24
installed. (We recommend to use ROOT 5.28).
Alternatively, you might use a ssh connection to lxplus.

* We will use the latest TMVA version, 4.1.0. This version
can either be installed at the beginning of the tutorial
or before the workshop following the description here:
https://twiki.cern.ch/twiki/bin/view/TMVA/WebHome

* tutorial topics:
- Installing TMVA, running basic examples
- Example of a complex user classification analysis
- Example regression analysis

Emphasis will be on the last two topics

I am looking forward seeing you at CERN! Best regards, Eckhard

TMVA stand-alone vs. TMVA in ROOT

Using TMVA stand-alone: the TMVA in ROOT is ignored. The tmva/test directory may serve as an example setup. Do not forget to run setup.sh

Using TMVA in ROOT: use ROOT's include and library pathes, need to modify tmva/test/Makefile

- Erase all reference to stand-alone include directories
- Replace –l TMVA.1 by –l TMVA
- You no longer need to run setup.sh
- No changes for using macros (do not source setup.sh)

This tutorial: We will use TMVA-standalone

TMVA version 4.1.0, (the version included in ROOT 5.28)

universitätbonn

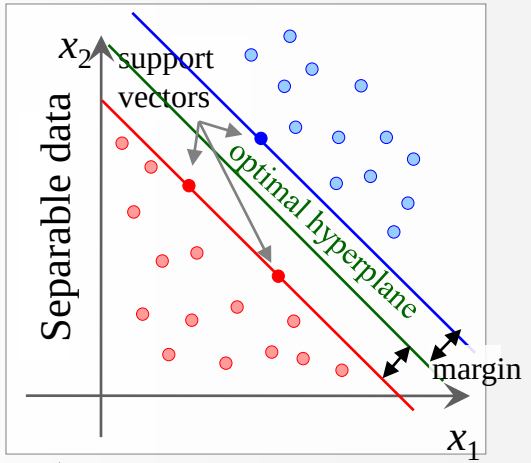
A TMVA method not covered so far

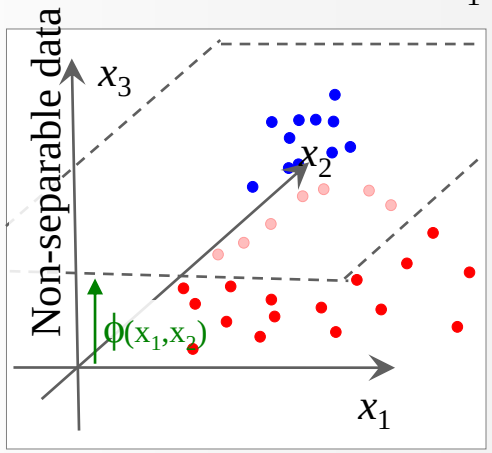
Support Vector Machines

(needed for exercise 2)

Support Vector Machines

- Linearly separable data:
  - find separating hyperplane (LD, Fisher)
- Non-separable data:
  - transform variables into a high dimensional space where linear separation is possible
- Advantages:
  - flexibility combined with robustness
- Disadvantages:
  - slow training
  - complex algorithm

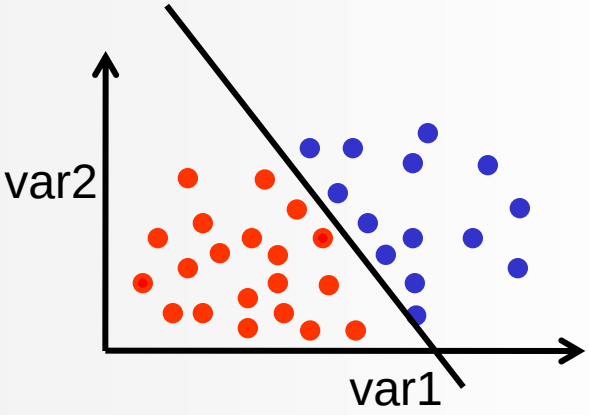




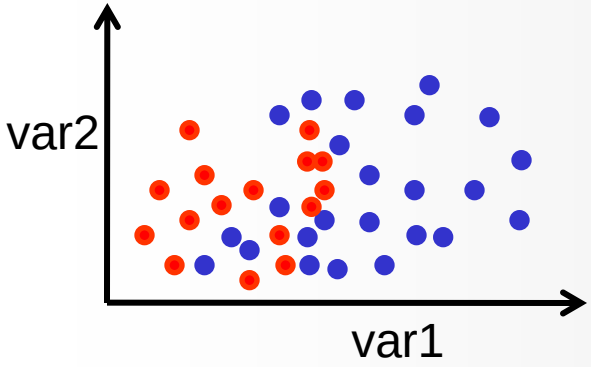
1

Performance		Speed		Robustness		Curse of Dim.	Transparency	Regression	
No/linear correlations	Nonlinear correlations	Training	Response	Overtraining	Weak input vars			1D	multi D
☺	☺	☹	😐	😐	😐	😐	☹	☺	☺

Exact linear separability



Linearly separable



Not linearly separable in two dimensions

Improve performance of linear classifier by adding extra dimensions to the problem.

With Support Vector Machine this is done in an elegant, although mathematically convoluted way.

Embedding in a larger-dimensional space

Example for the LD-classifier, not SVM!

A three-dimensional problem

```
factory->AddVariable("var0", 'F');
factory->AddVariable("var1", 'F');
factory->AddVariable("var2", 'F');
factory->BookMethod(
   TMVA::Types::kLD, "LD",  "!V"
   );
```

The LD classifier:

MVAvalue = $a_0 + \Sigma_{i=1,Nvar}\ a_i\ x_i$

The LD classifier performs poorly on non-linear problems but…
..simple
..small number of paras (N+1)

Embedded in a 30-dimensional space

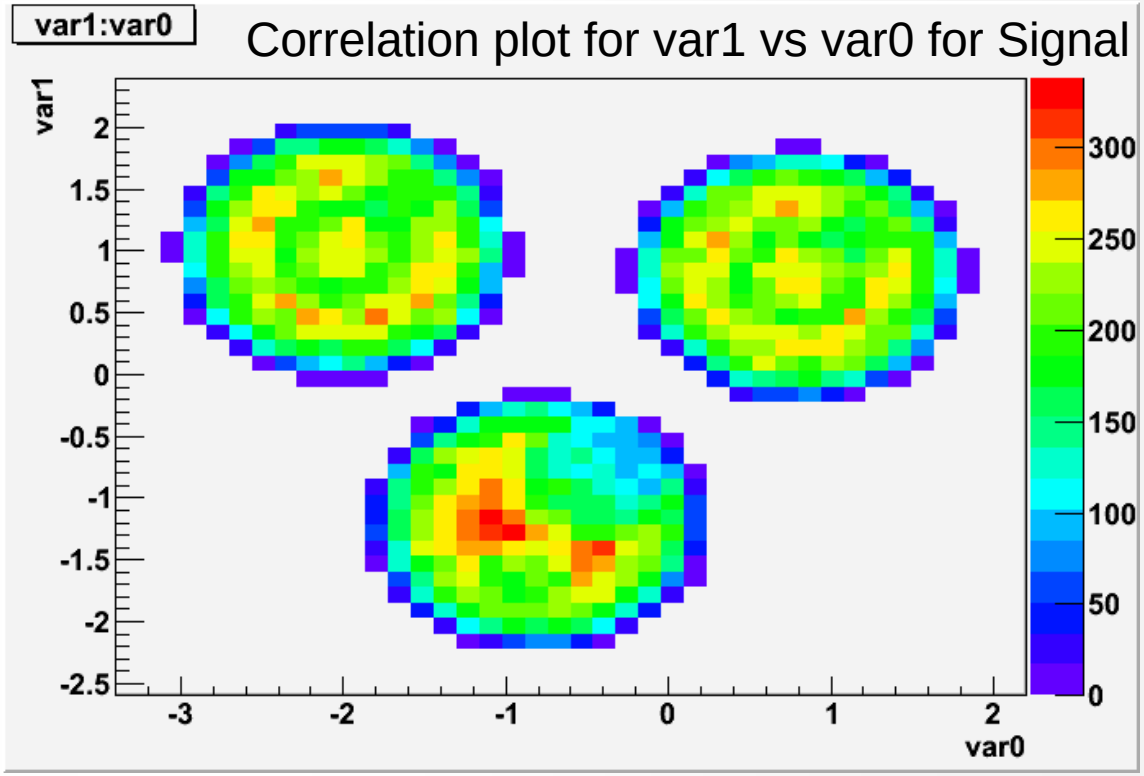
```
factory->AddVariable("var0", 'F');
factory->AddVariable("var1", 'F');
factory->AddVariable("var2", 'F');
// single term multinomials of second degree
factory->AddVariable("var0*var1", 'F');
factory->AddVariable("var0*var2", 'F');
factory->AddVariable("var1*var2", 'F');
// single term multinomials of third degree
factory->AddVariable("var0*var0*var0", 'F');
factory->AddVariable("var0*var0*var1", 'F');
...
factory->AddVariable("var2*var2*var2", 'F');
// single term multinomials of fourth degree
factory->AddVariable("var0*var0*var0*var0", 'F');
...
factory->AddVariable("var2*var2*var2*var2", 'F');
```

Exercise 1: Getting started

- Go to tutorial web page
  http://www.uni-bonn.de/~etoerne/tmva/
- Compile and run TMVAClassification and TMVAClassificationApplication
- All information is provided on the web page.

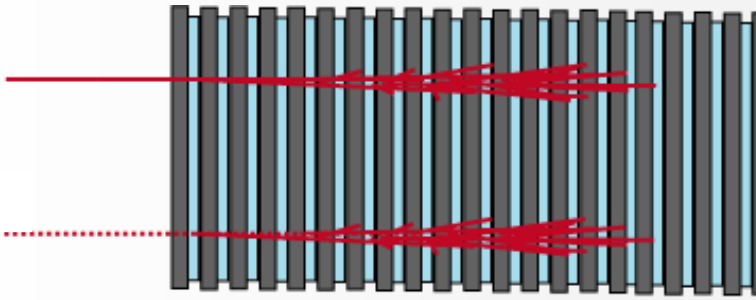
Exercise 2

- Classification Analysis on Data „testData.root“ on the tutorial page
- 3-dimensional data with complex signal and flat background
- Task: Find best classifier (choice of BDT,MLP,SVM, Likelihood)



Exercise 3: Measuring calorimeter energy

- Regression analysis: estimate of observable (target) based on input variables.
- data represent measurements in a toy-calorimeter.
- target to be estimated: energy of calo cluster.
- Calorimeter is segmented
  - five thin layers (“EM-CALO”)
  - followed by eight thicker layers
- Calorimeter is imperfect
  - leakage at the end of the calorimeter
  - dead regions
  - non-compensation.
- data are from jets and single particles.
- Always one cluster per event.



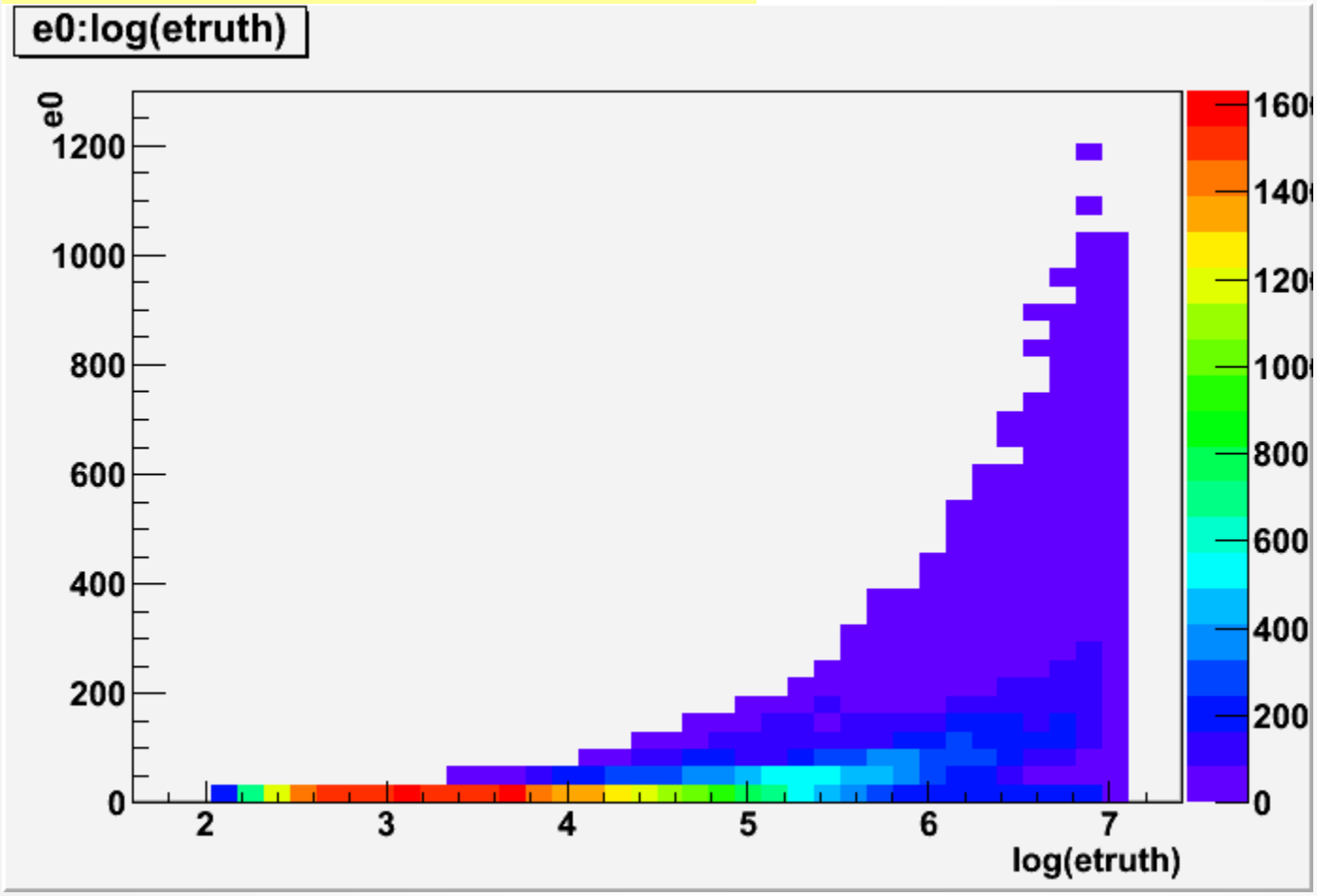
Exercise 3: Measuring calorimeter energy

- **Variables:**
  - **Energy in each layer: e0, e1, … e12. (Given in GeV)**
  - **Sum over all layers: esum**
  - **The true energy deposition: etruth**
  - **Cluster center-of-gravity in eta: eta, and phi: phi**
  - **Cluster centroid in layer 0 in eta and phi: eta0, phi0**

- **Either use etruth or etruth/esum as target.**

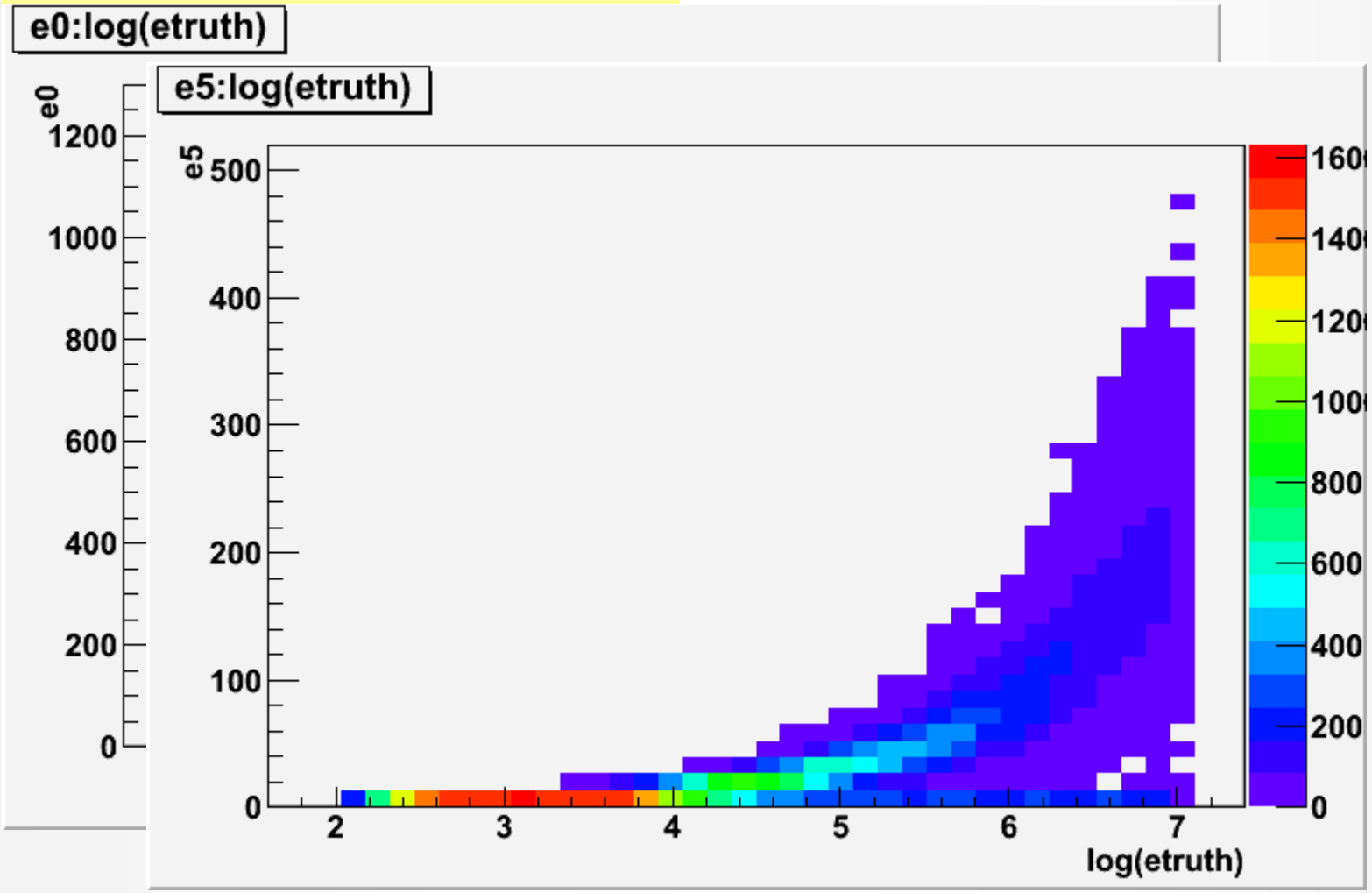
Exercise 3: Measuring calorimeter energy

Energy in first layer vs true energy



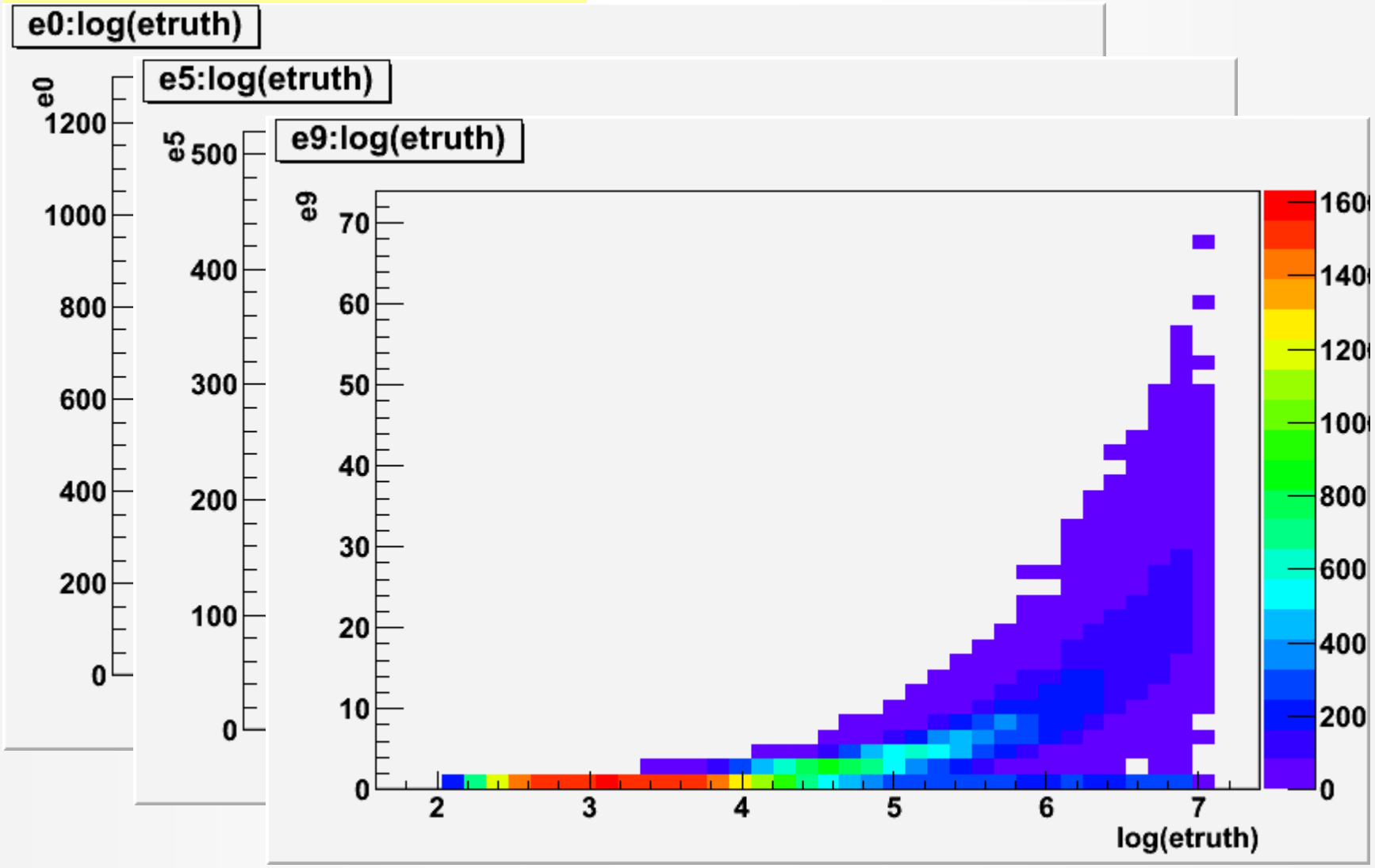
Exercise 3: Measuring calorimeter energy

Energy in layer 5 vs true energy

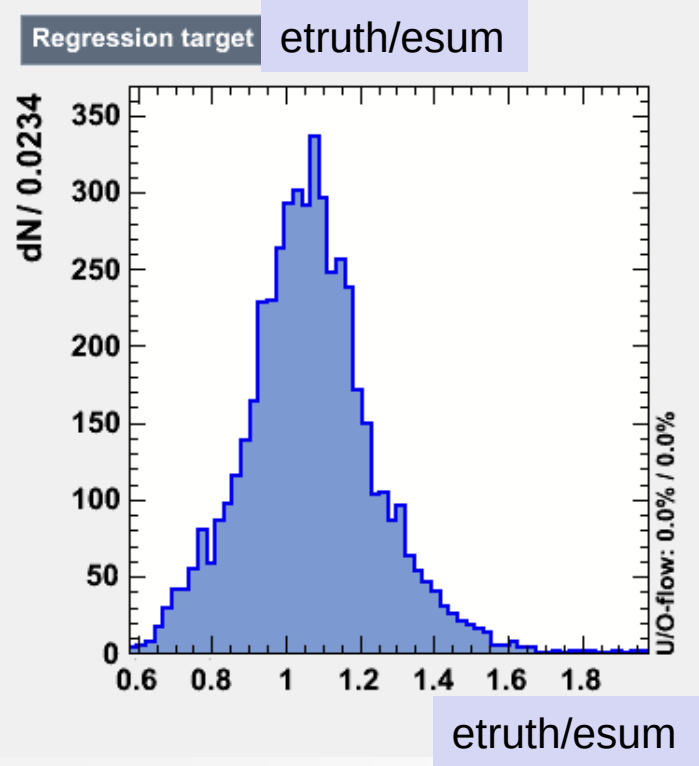


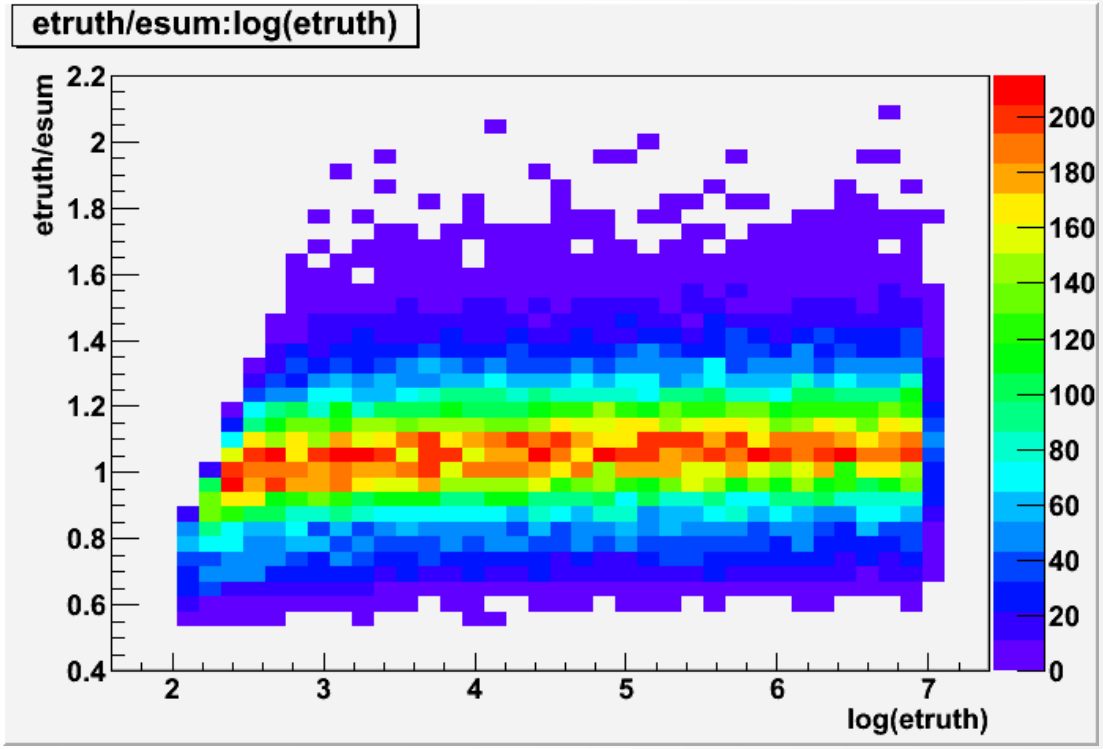
Exercise 3: Measuring calorimeter energy

Energy in layer9 vs true energy



Exercise 3: Measuring calorimeter energy





Average ratio < etruth/esum > = 1.06

Standard deviation of ratio etruth/esum = **0.175**

Regression-estimate (std-dev of estimate – truth) should be much less than 0.175.

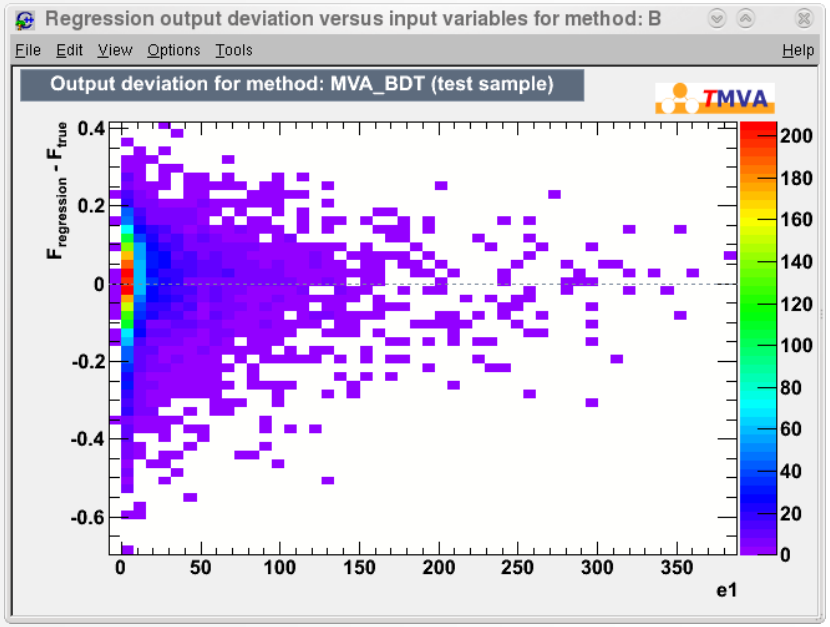
MVA methods for regression

Several TMVA methods are still under development for regression

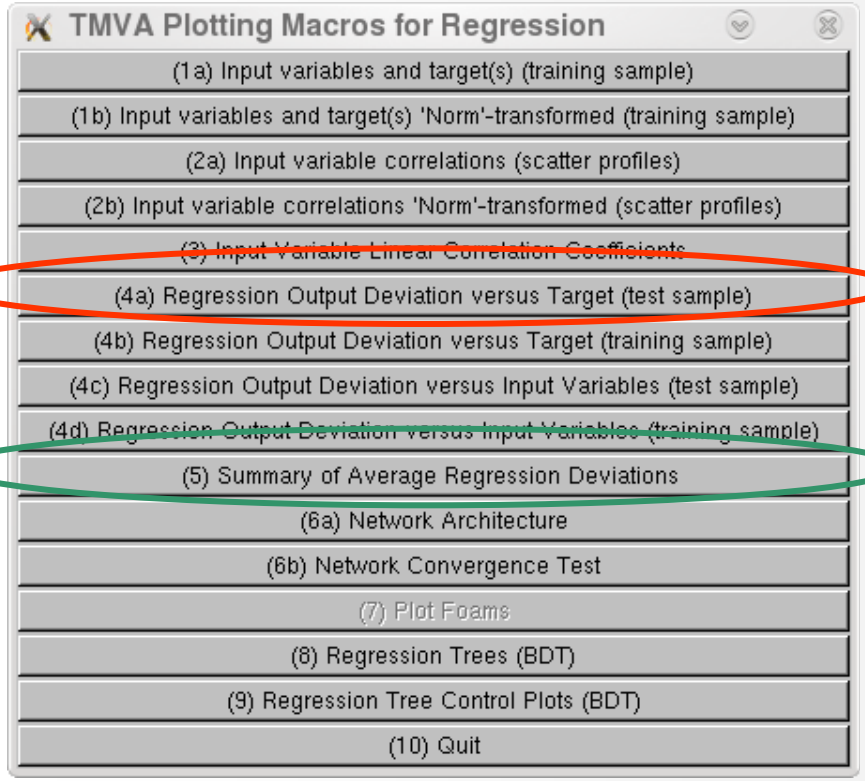
For this exercise, consider to use:

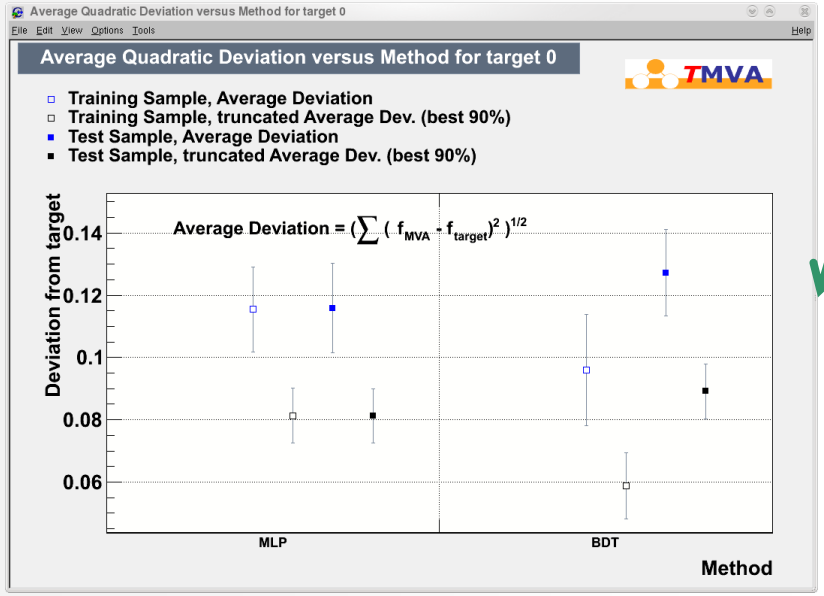
- MLP with BFGS training (option TrainingMethod=BFGS)
- BDT with BoostMethod=Grad
- PDEFoam
- FDA

Regression macros



TMVARegGui.C



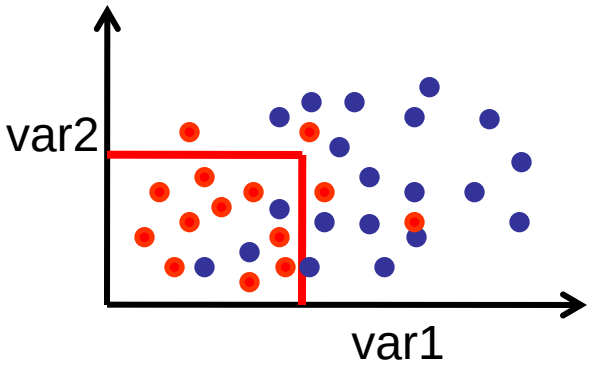


universitätbonn

BACKUP

Cuts

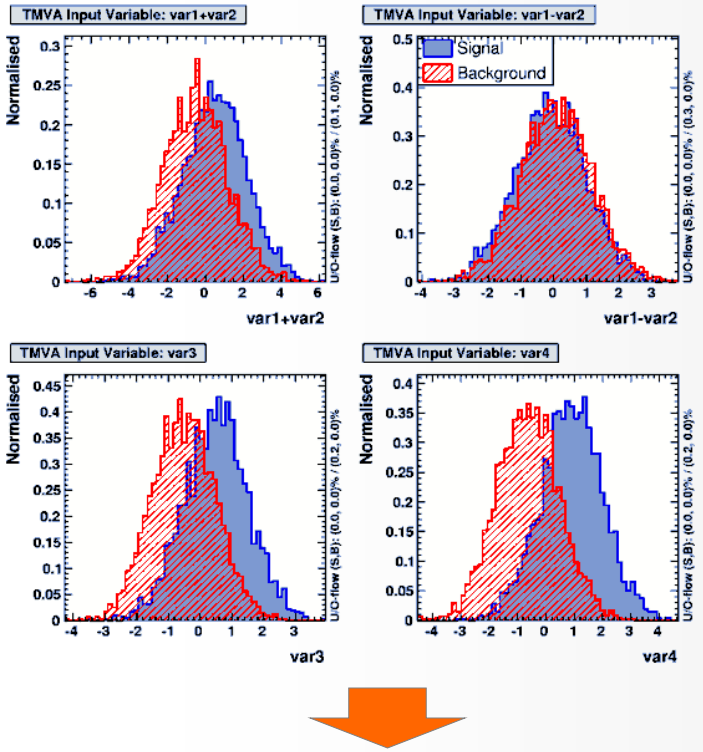
- Simplest multi-variate classification method: **Cuts**
- Signal region is defined by a series of cuts:
  - Var1 > x1
  - Var2 < x2 ...



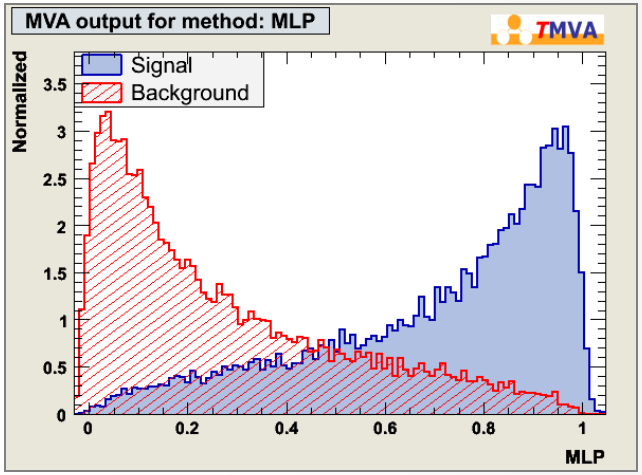
- However not necessarily the easiest approach
  - Cuts have difficulties if variables have low separation power and many variables are involved
- Whenever cut selections are employed, more sophisticated multi-variate methods may also work

What is a multi-variate analysis

- "Combine" all input variables into one output variable
- Supervised learning means learning by example: the program extracts patterns from training data
- Methods for un-supervised learning → not common in HEP and not covered in this lecture



Input Variables

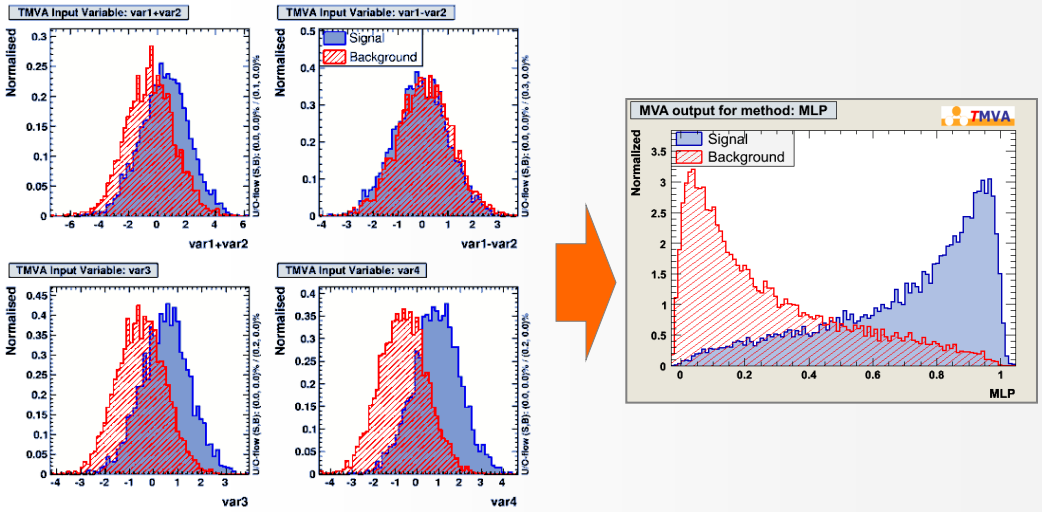


Classifier Output

Typical multi-variate analysis steps

- Choice of input variables
- Define preselection
- Choice of MVA method
- Training the MVA method using samples with known signal/background
- Choice of working point

physics input is crucial



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');

  factory->AddSignalTree ( (TTree*)input->Get("TreeS"), 1.0 );
  factory->AddBackgroundTree ( (TTree*)input->Get("TreeB"), 1.0 );
  //factory->AddRegressionTree ( (TTree*)input->Get("regTree"), 1.0 );
  factory->PrepareTrainingAndTestTree( "", "",
  "nTrain_Signal=200:nTrain_Background=200:nTest_Signal=200:nTest_Background=200:NormMode=None!V" );

  factory->BookMethod( TMVA::Types::kLikelihood, "Likelihood",
  "!V:!TransformOutput:Spline=2:NSmooth=5:NAvEvtPerBin=50" );
  factory->BookMethod( TMVA::Types::kMLP, "MLP",
  "!V:NCycles=200:HiddenLayers=N+1,N:TestRate=5" );

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

Create Factory

Add variables/ targets

Initialize Trees

Book MVA methods

Train, test and evaluate