Multivariate Datenanalyse

- Generelle Fragestellung
  Multidimensionale Merkmalsanalyse ≡ Multivariate Analyse ≡ MVA

  Es existieren 2 Kategorien von unterscheidbaren Daten, S und B, die durch einen Satz von diskreten Variablen beschrieben werden können. Welche “Kriterien” lassen sich zur Klassifikation verwenden?

  - Einzelne Kriterien sind nicht ausreichend zur Entscheidungsfindung von S oder B
  - Bestimmung der Datenstruktur (Funktionaler Zusammenhang, Regressionsproblem)
  - Reduktion des Variablenraums auf Wahrscheinlichkeiten für S oder B

- TMVA (Toolkit for Multi Variate Analyses)

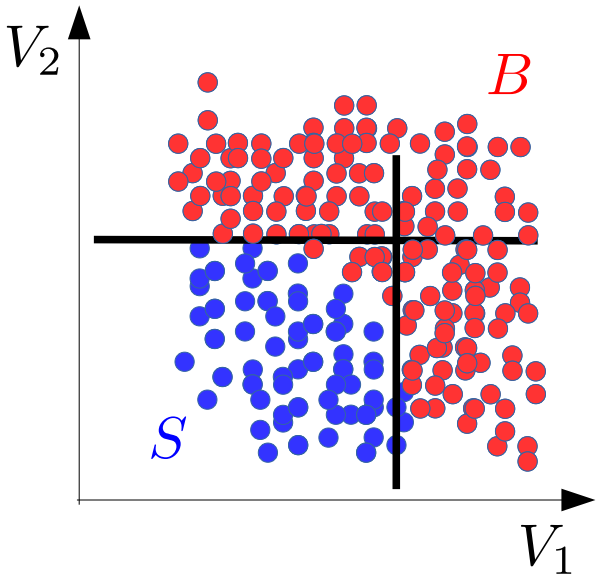
  - Dokumentation, wenn auch nicht vollständig aktuell
    https://root.cern/tmva
  - Users Guide
    https://root.cern.ch/download/doc/tmva/TMVAUsersGuide.pdf
  - Talks
    http://tmva.sourceforge.net/talks.shtml
  - ICHEP Proceedings
    http://iopscience.iop.org/article/10.1088/1742-6596/219/3/032057/pdf
  - Vollständig in ROOT integrierte Open Souce Software

Multivariate Analyse in ROOT - Einleitung

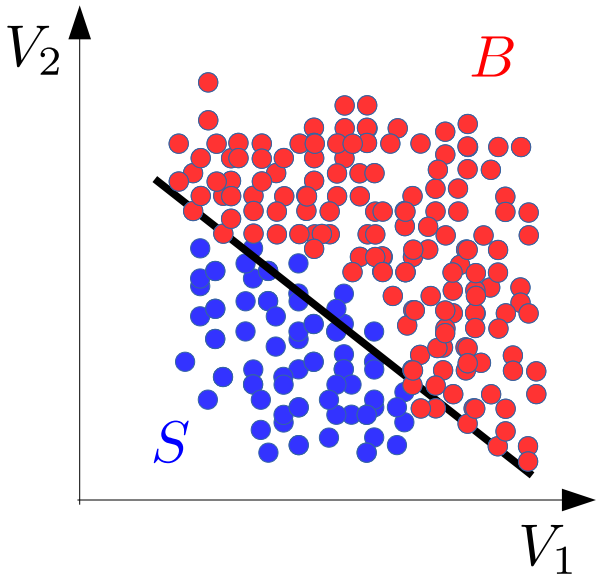
- TMVA ist ein in ROOT integriertes machine learning Toolkit zur Separation und Regression von Daten (https://root.cern.ch/TMVA)

  - Klassifikation / Separation von diskreten Messungen bezüglich einer oder mehrerer Observabler $(V_1, V_2, ..., V_n)$

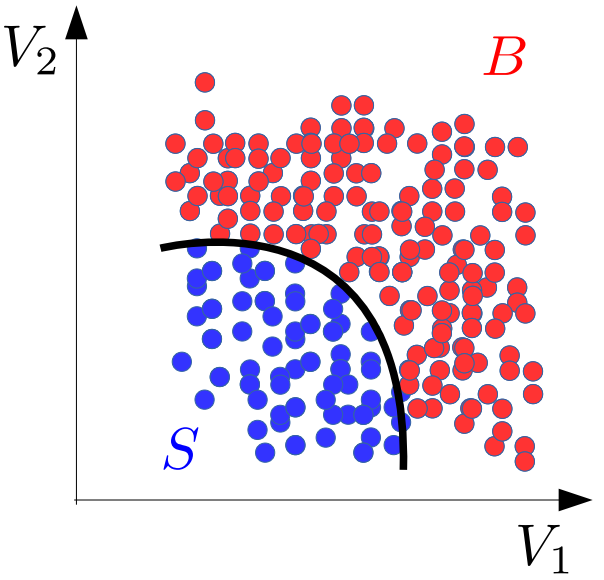
Rechteckige Schnitte



Lineare Funktion



Nicht-Lineare Funktion

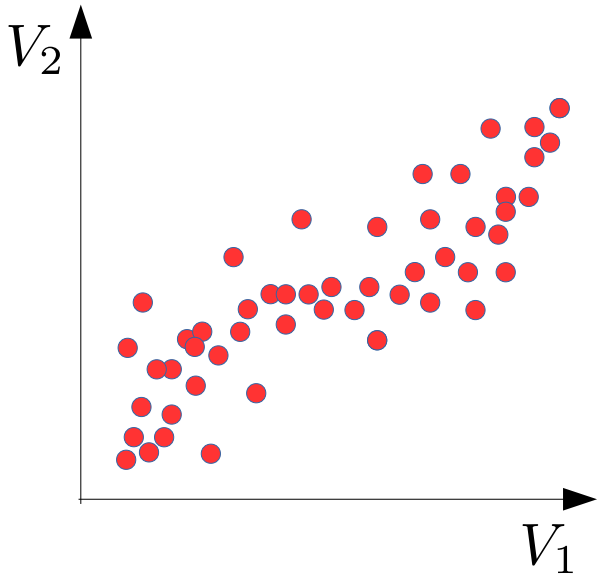


  - Mit TMVA lassen sich mit verschiedenen Methoden die optimalen Separationsbedingungen unter Berücksichtigung der Korrelationen der Observablen bestimmen.

Multivariate Analyse in ROOT - Einleitung

- TMVA ist ein in ROOT integriertes machine learning Toolkit zur Separation und Regression von Daten (https://root.cern.ch/TMVA)

  - Es besteht ein enger Zusammenhang zwischen Klassifikation und Regression. Aus einem Satz von gemessenen diskreten Observablen wird der funktionale Zusammenhang ermittelt und daraus dann optimale Werte für eine Ziel-Funktion bestimmt oder Grenzen im multidimensionalen Observablenraum $(V_1, V_2, ..., V_n)$ gesetzt. .



Konstante Funktion

Lineare Funktion

Nicht-Lineare Funktion
Abschnittsweise definierte Funktion
Splines

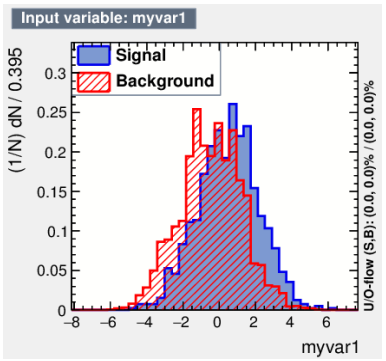
- supervised regression: Model ist bekannt
- unsupervised regression: Model ist unbekannt

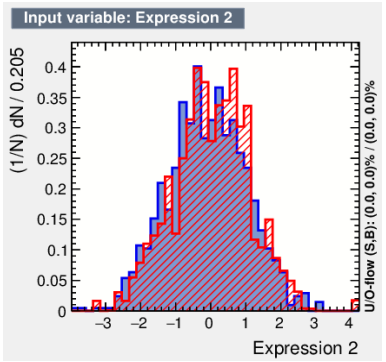
- Parameterbestimmung erfolgt mit Maximum Likelihood Fits

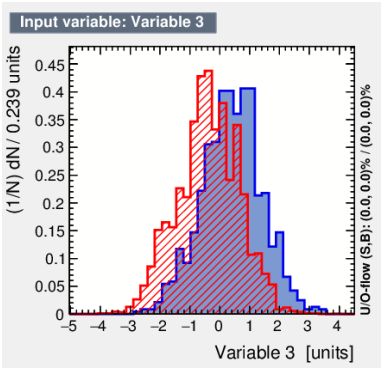
MVA Klassifikation in N-Dimensionen

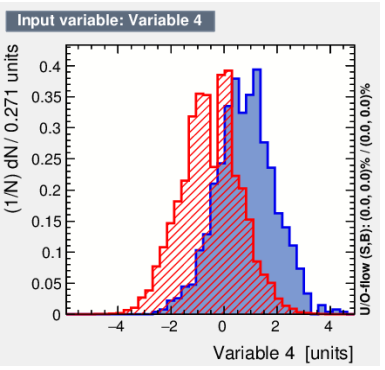
- Untergrund oder Signal ?

  - Für jedes Ereignis existieren N gemessene Variable







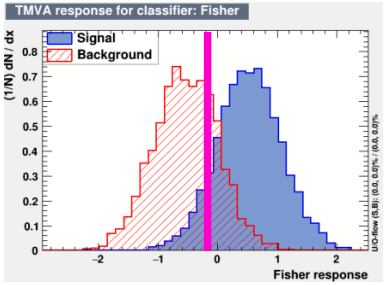


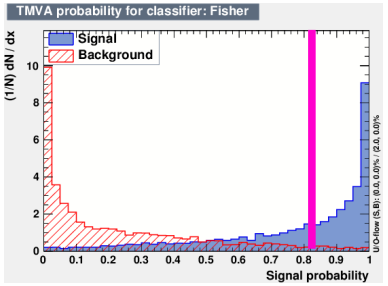
.....

$$\vec{V} = \{V_1, V_2, ...V_N\}$$

  - Ziel: Suche nach einer Abbildung des N-dimensionalen Input Variablenraumes auf einen 1-dimensionalen Output Raum

$$F(\vec{V}) : \mathbb{R}^N \rightarrow \mathbb{R}$$





  - Ein einfacher Schnitt in F implementiert komplexe Schnitte im N-dimensionalen Input Variablenraum.

  - Wie können wir $F(\vec{V})$ bestimmen?

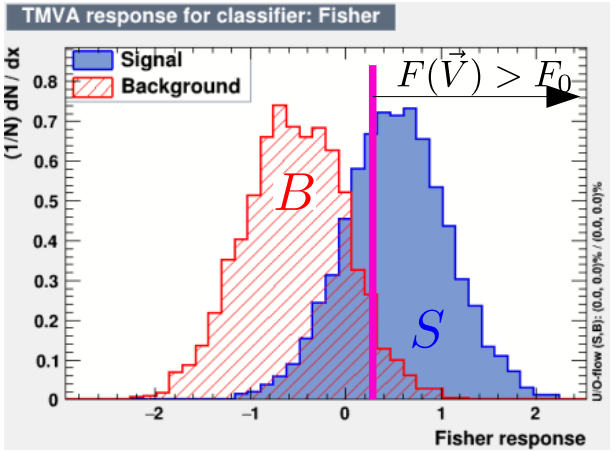
    Verwende ein Model (linear, nichtlinear, stückweise) und passe freie Parameter mit Hilfe vorhandener Typ Definition an

    I) Separiere Signal und Untergrund (Klassifikation)

    II) Optimale Beschreibung einer Ziel-Funktion (Regression)

MVA Klassifikation in N-Dimensionen

- Kenngrößen

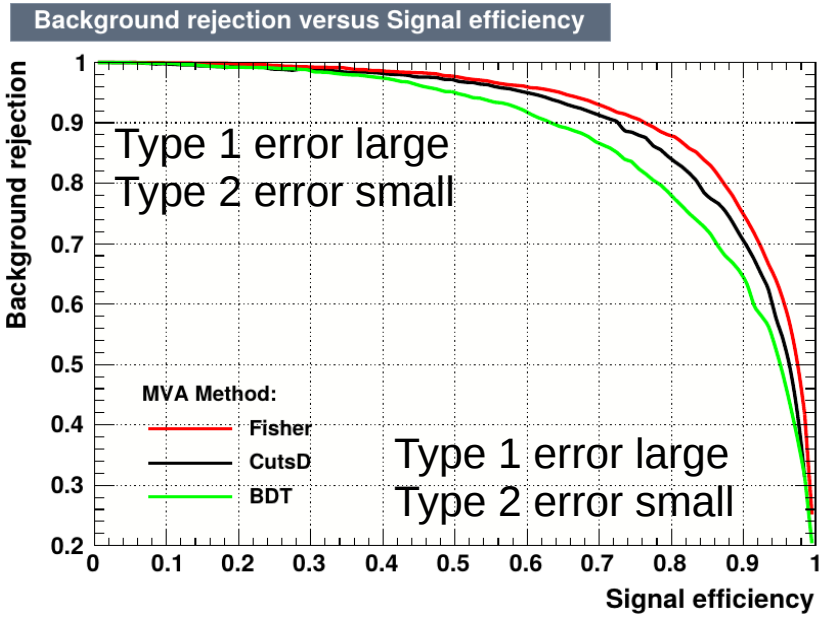


- Wichtige Größen zur Beurteilung der Qualität sind Effizienz (efficiency) und Reinheit (purity):

$$\epsilon = \frac{N_S(F > F_0)}{N_S}$$

$$\pi = \frac{N_S(F > F_0)}{(N_S + N_B)(F > F_0)}$$

- Reciever Operations Characterics (ROC) curve

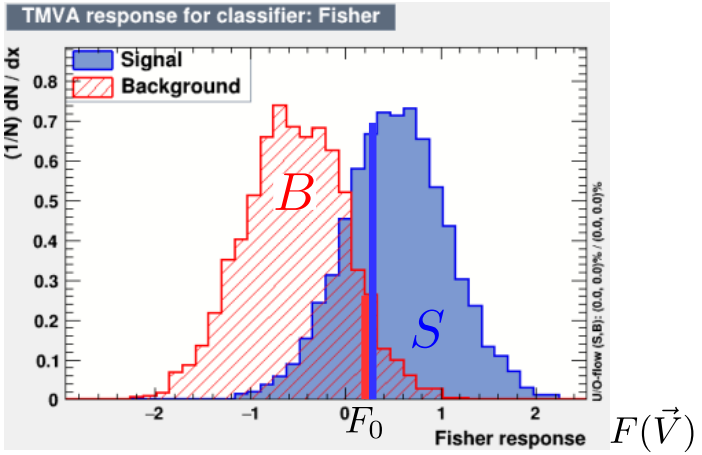


- Bei der Klassifikation aufttretende Fehler

Wahre Kat. \ Gefundene Kat.	Signal	Background	
Signal	Ok	Type 2 error	Verlust Reinheit
Background	Type 1 error	Ok	
	Verlust Effizienz		

MVA Klassifikation in N-Dimensionen

- Interpretation von $F(\vec{V})$



- Die Verteilung von $F(\vec{V}|S)$ und $F(\vec{V}|B)$ lassen sich als Wahrscheinlichkeitsdichtefunktion (probability density function, PDF) interpretieren, $PDF_S(F)$ und $PDF_B(F)$.

- Daraus läßt sich für ein gegebenes F_0 die Wahrscheinlichkeit für Signal und Background bestimmen, wenn das S/B Verhältnis für die Daten bekannt ist.

$$P(data = S|F) = \frac{f_S \cdot PDF_S(F)}{f_B \cdot PDF_B(F) + f_S \cdot PDF_S(F)}$$

- Ein Schnitt in der eindimensionalen Variablen $F(\vec{V}) = F_0$ und Akzeptieren aller Ereignisse rechts des Schnittes legt die Signaleffizienz und die Backgroundeffizienz (Background rejection) fest. Eine systematische Änderung von $F(\vec{V})$ liefert die ROC curve.

- Der einfache Schnitt entspricht einer komplizierten Hyperfläche, die nicht notwendiger Weise, abhängig vom MVA Algorithmus, funktional beschreibbar sein muss.

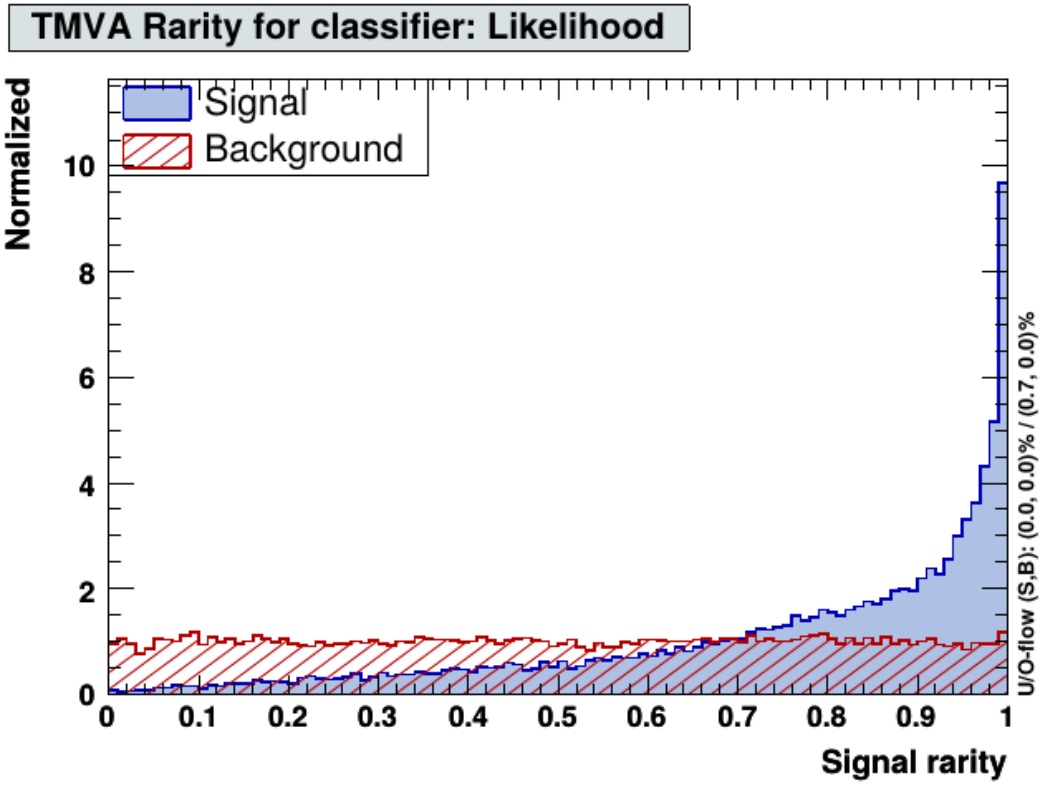
MVA Klassifikation in N-Dimensionen

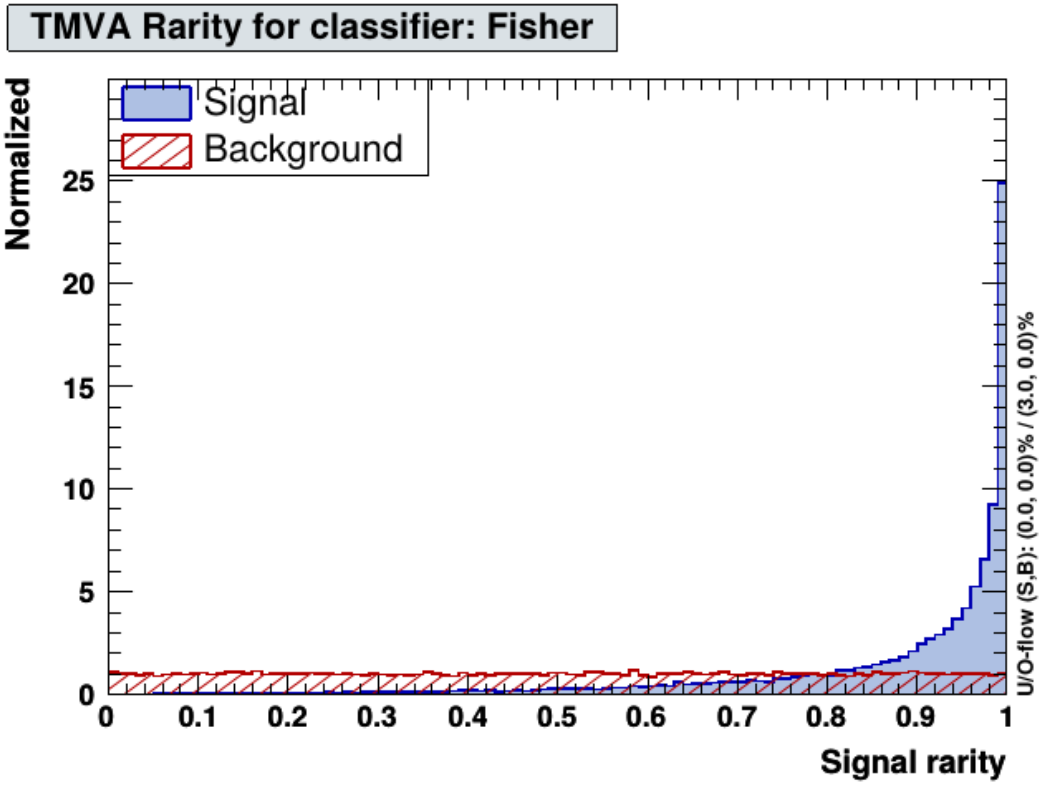
- Integraltransformation der Klassifikationswahrscheinlichkeit – Rarity $\mathcal{R}(y)$

  - Für die Klassifikationsvariable *F* wird das Integral

$$\mathcal{R}(F) = \int_{-\infty}^{F} PDF_B(F')dF'$$

bestimmt. $\mathcal{R}(F_B)$ ist für Untergrund zwischen 0 und 1 gleichverteilt, während $\mathcal{R}(F_S)$ ein Maximum bei 1 hat. Die Verteilung ist ein Maß für die Qualität des Classifiers. Je besser das Maximum bei 1 definiert ist, je besser eignet sich der Classifier.





MVA Klassifikation in N-Dimensionen

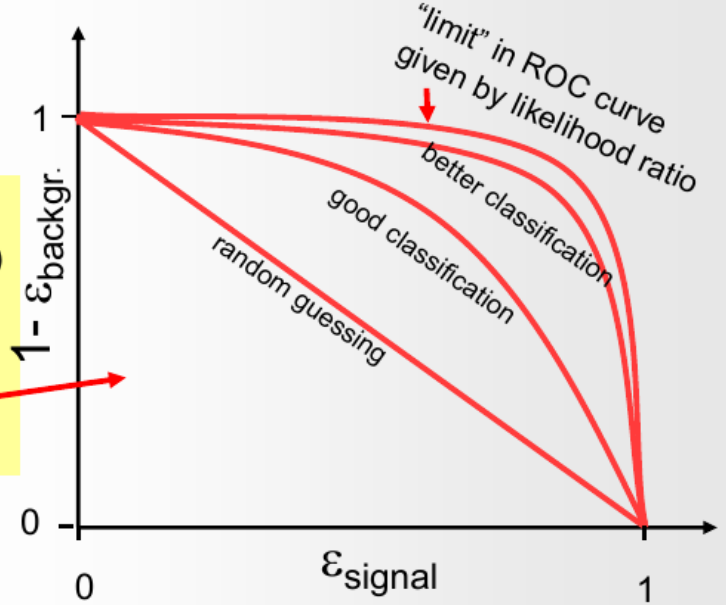
Neyman-Pearson Lemma

differential cross section of signal, bkg process

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

Neyman-Peason:
The Likelihood ratio used as "selection criterium" 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

$y(x) \equiv F(\vec{V})$
From Helge Voss, TMVA team



- Bemerkungen
  - Die wahren PDFs sind häufig nicht bekannt, daher nützt das obige Lemma nur bedingt.
  - Ausweg: Man bestimmt die Form von $P(data|F)$ in einem Training sample, in dem die PDFs und das Likelihood ratio bestimmt werden können.
    Dann findet man ein $F(\vec{V}) = F_0$, das eine Hyperebene in $\vec{V}$ festlegt, sodass Signal und Background optimal getrennt sind.
  - Wenn die Korrelationen zwischen den einzelnen Observablen $\vec{V} = (V_1, V_2, ..., V_n)$ schwach sind, ist die gesamte Likelihood

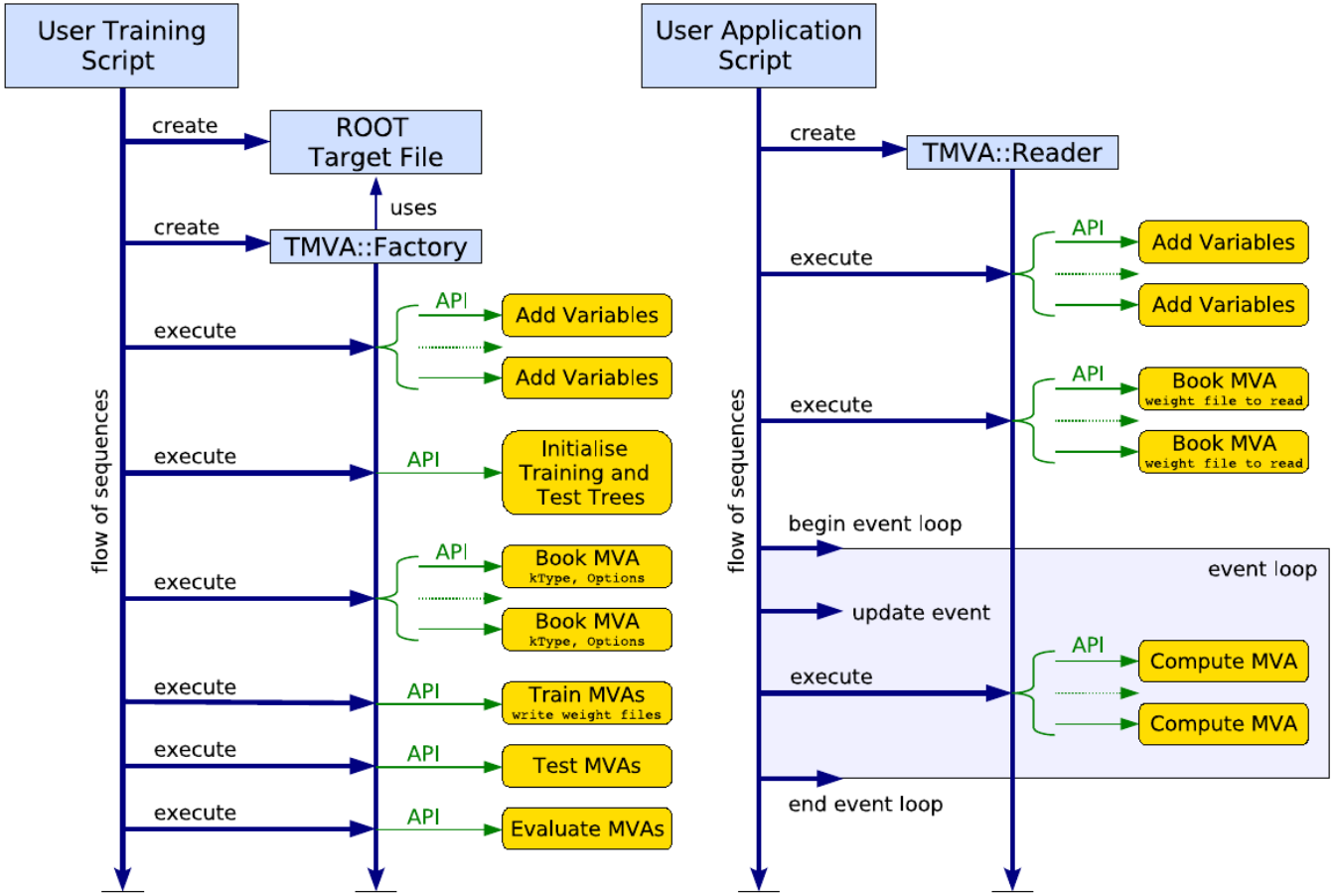
$$PDF(\vec{V}) = \prod_{i=1}^{n} PDF(V_i)$$

Ablauf einer Analyse mit TMVA

- Zwei unabhängige Schritte

  - Trainingsphase: Aufbau, Test und Bewertung von Klassifikationsmethoden und -größen durch die Verwendung von einem Datensatz, in dem Daten und Untergrund bekannt sind.

  - Anwendungsphase: Anwendung der Klassifikationsgrößen auf den zu analysierenden Datensatz



TMVA Algorithmen

- Mit TMVA stehen in ROOT komplizierte multivariate Klassifikationsalgorithmen zur Verfügung, die gleichzeitig getuned und verglichen werden können.
  Folgende objektorientierte Verfahren sind implementiert

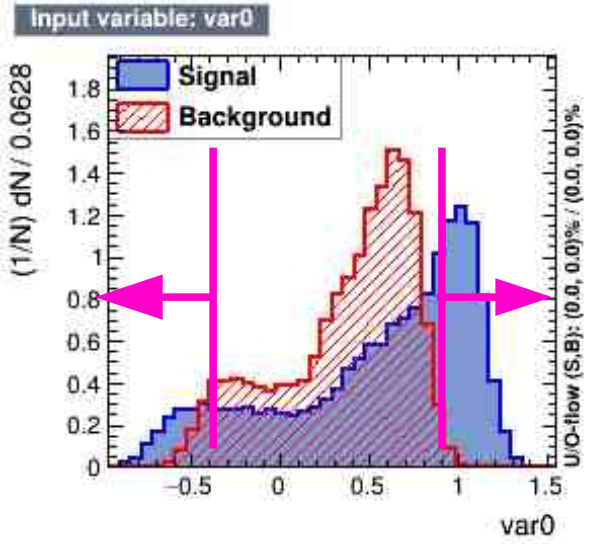
  - Rectangular cut optimization
  - Likelihood estimation (PDE, range search und k-NN)
  - Linear und nonlinear discriminant analysis (H-Matrix, Fisher, FDA)
  - Artificial Neural Networks
  - Boosted Decision Trees
  - Support Vector Machines
  - Predictive Learning via rule ensembles (rule fit)
  - Deep Neural Network (TMVA, Interface zu Keras/Tensorflow)

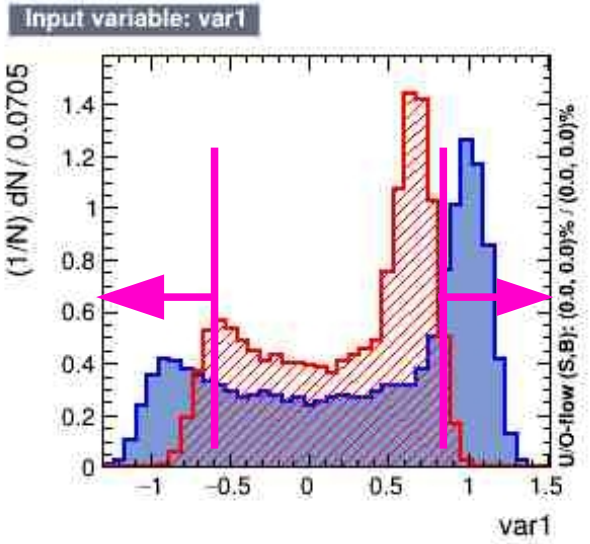
- Für einige Algorithmen ist eine Dekorrelation der Eingangsvariablen notwendig
  - Rotation im Variablenraum (so das die linearen Korrelationen minimal sind)
  - Verschiedene Transformationen für S und B

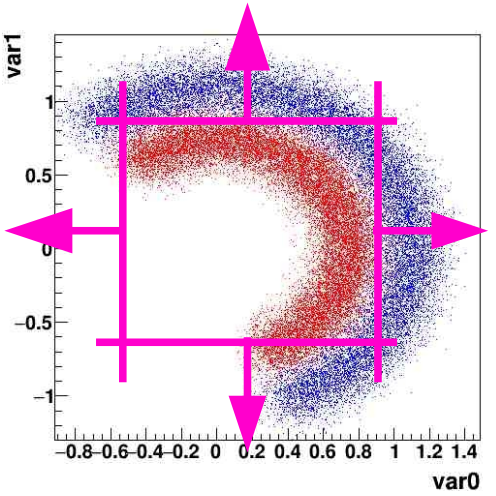
Schnitte in Variablen

- Der einfachste Klassifikator um Signal events zu selektieren sind Schnitte in allen Variablen, in denen eine Trennung zu beobachten ist.

  - Der Output ist eine binäre Antwort und keine Wahrscheinlichkeit für S oder B
  - Eine Optimierung der Schnitte wird durch Maximierung der Untergrund-unterdrückung für eine gegebene Signaleffizienz vorgenommen.
  - Signifikanz $sig = \epsilon_S \cdot N_S / \sqrt{\epsilon_S \cdot N_S + \epsilon_B(\epsilon_S) N_B}$

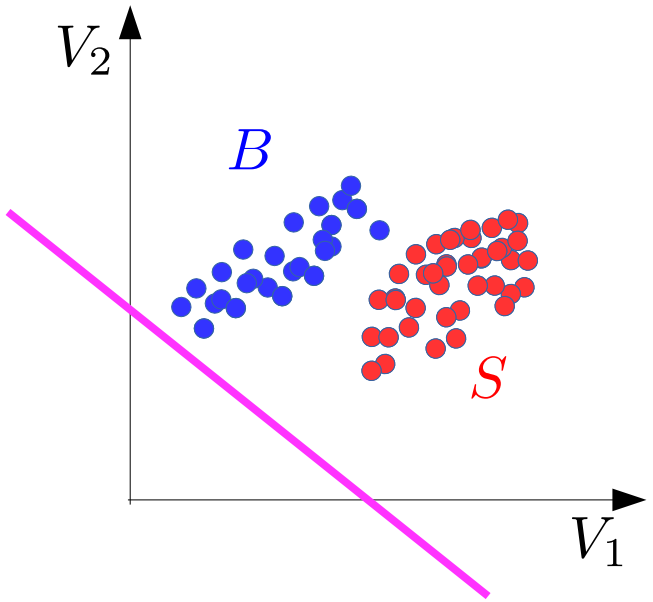






Fisher Diskriminante

- Idee: Finde eine Ebene, sodass die Projektion der Daten auf diese Ebene eine optimale Separation beider Ereignissklassen erlaubt.



- Die Fisher Diskiminante ist die gewichtete Linearkombination aller Input Variablen

$$F(\vec{V}) = \sum_i w_i \cdot V_i = \vec{w}^T \vec{V}$$

- $\vec{w}$ definiert die Orientierung der Ebene. Die Koeffizienten werden so bestimmt, das die Differenz der Erwartungswerte der beiden Klassen groß ist und die jeweiligen Varianzen klein. Dies wird durch Maximieren des folgenden Ausdrucks erreicht:

$$J(\vec{w}) = \frac{(F_S - F_B)^2}{\sigma_S^2 + \sigma_B^2} = \frac{\vec{w}^T K \vec{w}}{\vec{w}^T L \vec{w}}$$

- K ist die Kovarianzmatrix der Differenz der Erwartungswerte $F_S - F_B$ und L die Summe der Kovarianzmatrizen der beiden Klassen

- Zur Trennung beider Klassen wird ein kritischer Wert F_c benötigt, der durch Training bestimmt wird. Er legt die Effizienz und Reinheit der Klassenzugehörigkeit fest.

Verwendung von TMVA

- Zwei unabhängige Schritte

  - Trainingsphase: Aufbau, Test und Bewertung von Klassifikationsmethoden und -größen durch die Verwendung von einem Datensatz, in dem Daten und Untergrund bekannt sind.

  - Anwendungsphase: Anwendung der Klassifikationsgrößen auf den zu analysierenden Datensatz

- TMVA Test-Suite innerhalb der ROOT Installation

  - Aufsetzen der ROOT Umgebung:
    `source …../root/bin/thisroot.sh (Ihre eigene Installation)`
  - CIP: `source setroot.sh` und in der PATH Variablen auf shell den root Pfad suchen
  - TMVA Arbeitsdirectory
    `mkdir tmva ; cp -a …../root/tutorials/tmva/* tmva/.`
  - Ausführen der Test Klassifikationen unter Angabe der Algorithmen
    `cd tmva; root -l ./TMVAClassification.C\(\"Fisher,CutsD\"\)`
  - Während der Ausführung werden in der Konsole Informationen zu den im Aufruf angegebenen Algorithmen ausgegeben. Am Ende wird ein GUI geöffnet, in dem Kontrollverteilungen dargestellt werden können, die dann auch in `dataset/plots/` zu finden sind.
    Mit `root -l -e 'TMVA::TMVAGui()'` läßt sich das output ROOT File ansehen

Arbeitsvorschlag: TMVA Tutorial Test - Erste Schritte

- Versuchen Sie eine Anwendung aus dem Tutorial Bereich von TMVA auszuführen.
  I) Wo ist in der von uns verwendeten ROOT Installation das Tutorial directory mit tmva.
  II) Erzeugen Sie ein neues Arbeitsdirectory mit dem Namen tmva
  III) Führen zum Test im tmva directory folgenden code aus

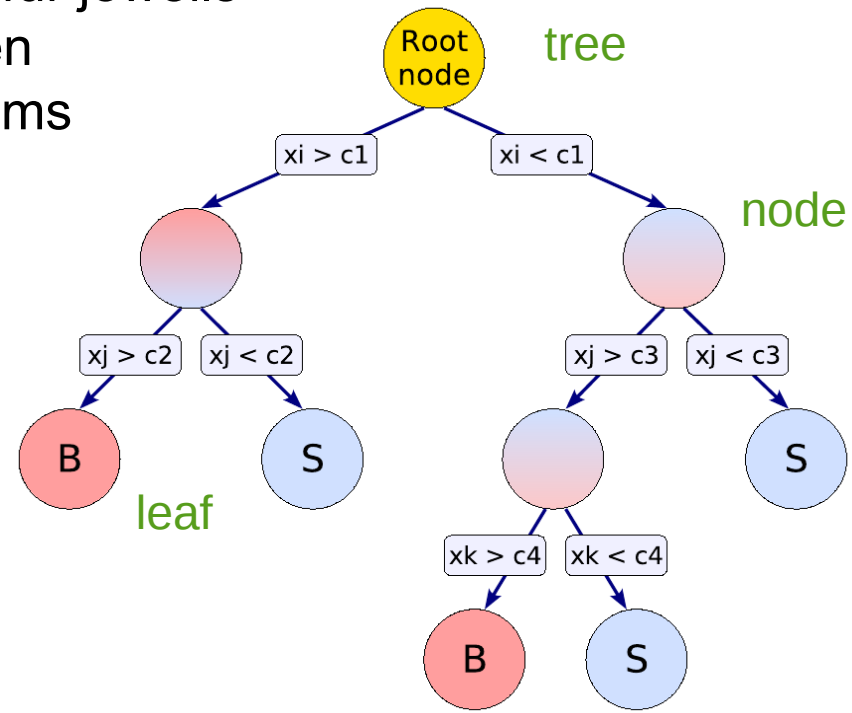
```
root -l ./TMVAClassification.C\(\"Fisher,CutsD\"\)
```

- Im Tutorial Bereich gibt es ein macro createData.C mit dem unter anderem Signal und Untergrund Daten mit verschiedenen Verteilungen erzeugt werden können.
  I) Wechseln Sie in ein neues Arbeitsdirectory `myTMVA` und kopieren Sie `createData.C`
  II) Die Funktion `create_circ` erzeugt die Datenverteilung, die für die Schnitte verwendet wurde.
  III) Erzeugen Sie eine shared library von `createData.C` und produzieren Sie einen Datensatz mit `create_circ ,` der 50000 Ereignisse enthält. Der default name ist `data.root` ; benennen Sie den in `CircleData.root` um. Wir wollen einen weiteren **unabhängigen** Datensatz erzeugen, der dann in der Anwendungsphase verwendet werden soll. Was können wir tun? Der Name des zweiten Datensatzes soll `CircleAppData.root` sein.

Decision Trees

- Decision tree

  Eine binäre baumartige Entscheidungsstruktur für jeweils eine Variable. Jede Teilung wird mit einer neuen Variablen bis zum Erreichen eines stop Kriteriums durchgeführt.
  Der Raum aller Variablen wird dadurch in viele Regionen unterteilt bis eine S oder B Klassifikation erreicht ist.



- Vorteile

  - Einfach zu verstehen
  - Gut geignet für multivariate Analyse
  - Einfaches schnelles Training

- Nachteile

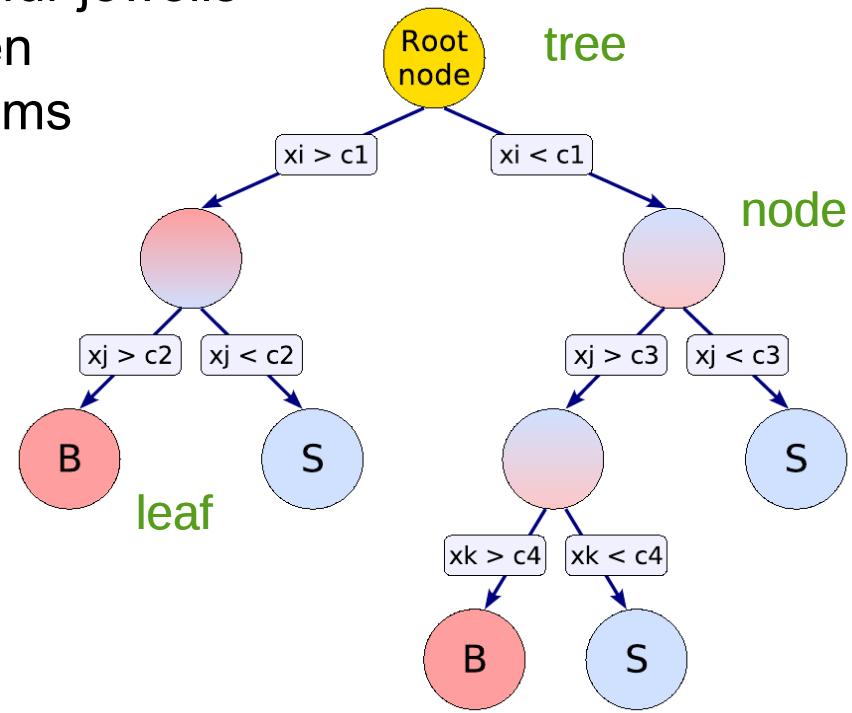
  - Empfindlich für statistische Fluktuationen in den Daten
  - Ein Baum hat nur unzureichende Klassifizierungsergebnisse

  → weitere Methoden notwendig, verwende viele decision trees (forest)
  Boosted Decision Tree (BDT)

Decision Trees

- Decision tree

  Eine binäre baumartige Entscheidungsstruktur für jeweils eine Variable. Jede Teilung wird mit einer neuen Variablen bis zum Erreichen eines stop Kriteriums durchgeführt.
  Der Raum aller Variablen wird dadurch in viele Regionen unterteilt bis eine S oder B Klassifikation erreicht ist.



- stop Kriterien

  - Untere Grenze der events / node erreicht
  - Obere Grenze der Anzahl der nodes erreicht
  - Maximale Tiefe erreicht
  - Weitere Unterteilung liefert keine Verbesserung (separation gain)

- separation gain

  - Definiere ein Maß für die Mischung von S und B in einem node
    Gini index: $G = p \cdot (1 - p), \quad p \equiv$ purity
  - separation gain: $N_{parent} \cdot G_{parent} - N_{left} \cdot G_{left} - N_{right} \cdot G_{right}$
    → Maximize

Boosted Decision Trees

- Boosting

Beschreibt eine Verbesserung der Klassifikation von MVA Algorithmen durch die Mehrfachverwendung von umgewichteten training samples. Das Resultat ist dann die gewichtete Summe. Typischer Weise wird durch die Kombination von vielen schwachen MVA Methoden ein erheblicher Gewinn erzielt.

- AdaBoost (adaptive boosting) Freund & Schapire, 1996

Misklassifizierte Events erhalten im Training der folgenden trees ein höheres Gewicht α, das über die Misklassifikationsrate err bestimmt wird

$$\alpha = \frac{1 - err}{err}$$

Die Gewichte des gesamten Samples werden so normiert, das die Summe der Gewichte konstant bleibt.

$$y_{Boost}(\vec{V}) = \frac{1}{N_{collection}} \sum_{i}^{N_{collection}} ln(\alpha_i) \cdot F_i(\vec{V})$$

y_{Boost} ist der gesamte gewichtete Klassifikator

$N_{collection}$ ist die Anzahl der umgewichteten training samples

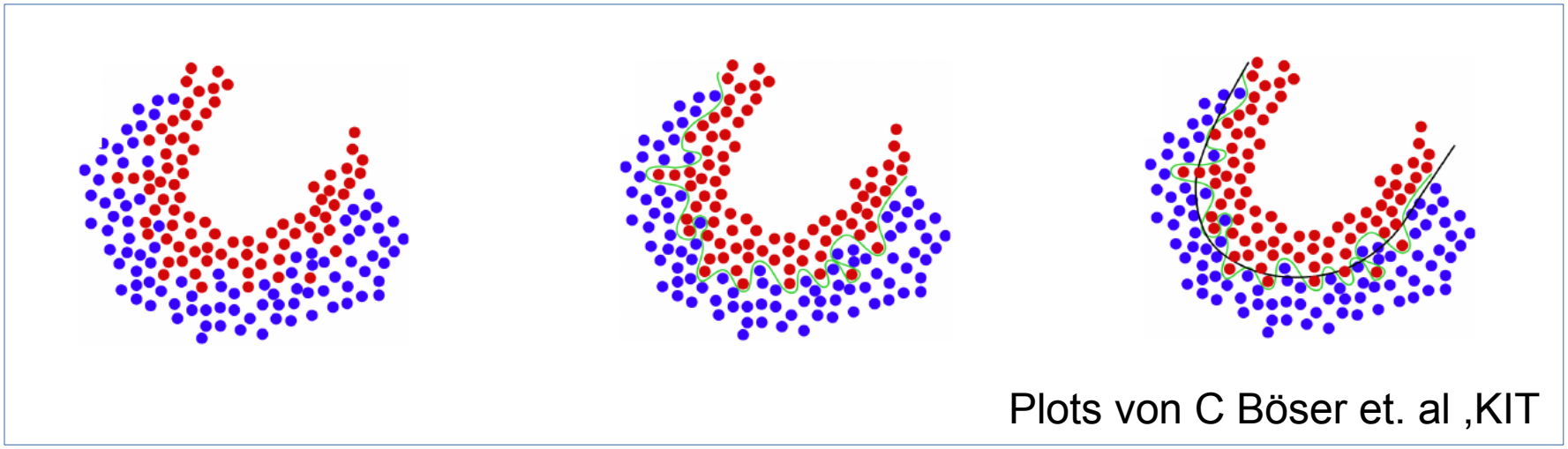
- Boost Methoden werden in TMVA nicht nur auf decision trees angewendet, sondern auch auf Fisher.

Boosted Decision Trees

- Overtraining

Ein boosted decision tree learnt statistische Fluktuation und betrachtet Sie als Eigenschaften des Trainingssamples.

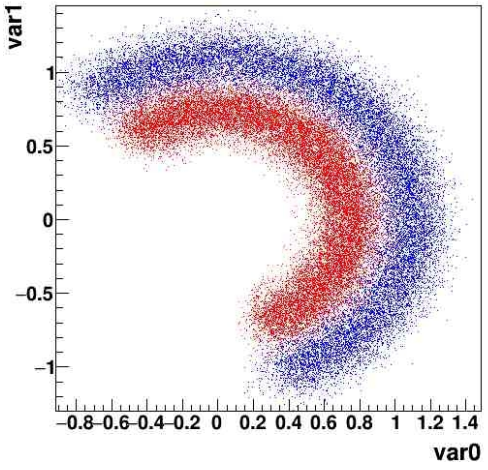
- Dies wird durch Aufspalten des Datensatzes in ein Trainings- und ein Testset verhindert. Es gibt Kontrollplots, die die Resultate der Outputvariablen von beiden Sets zeigen. Es sollte unbedingt überprüft werden, ob beide übereinstimmen.
- Insuffiziente Knoten können entfernt werden (pruning).

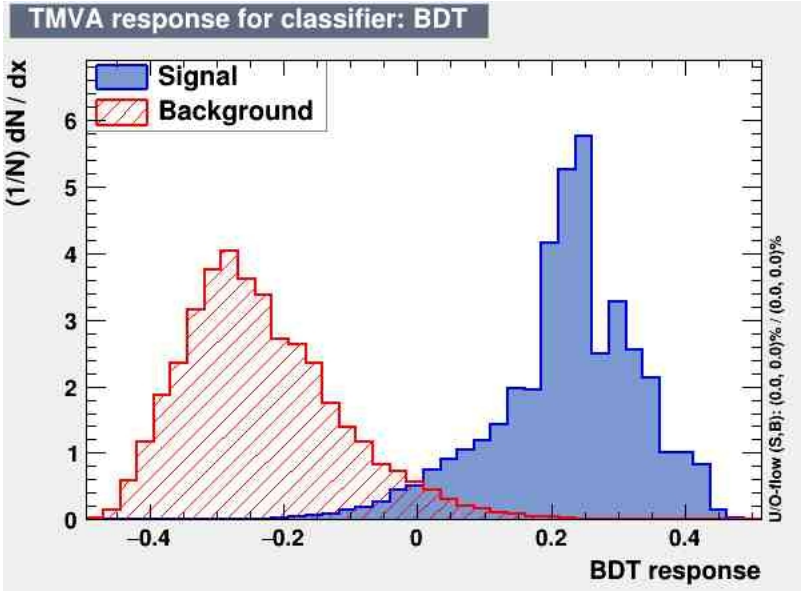


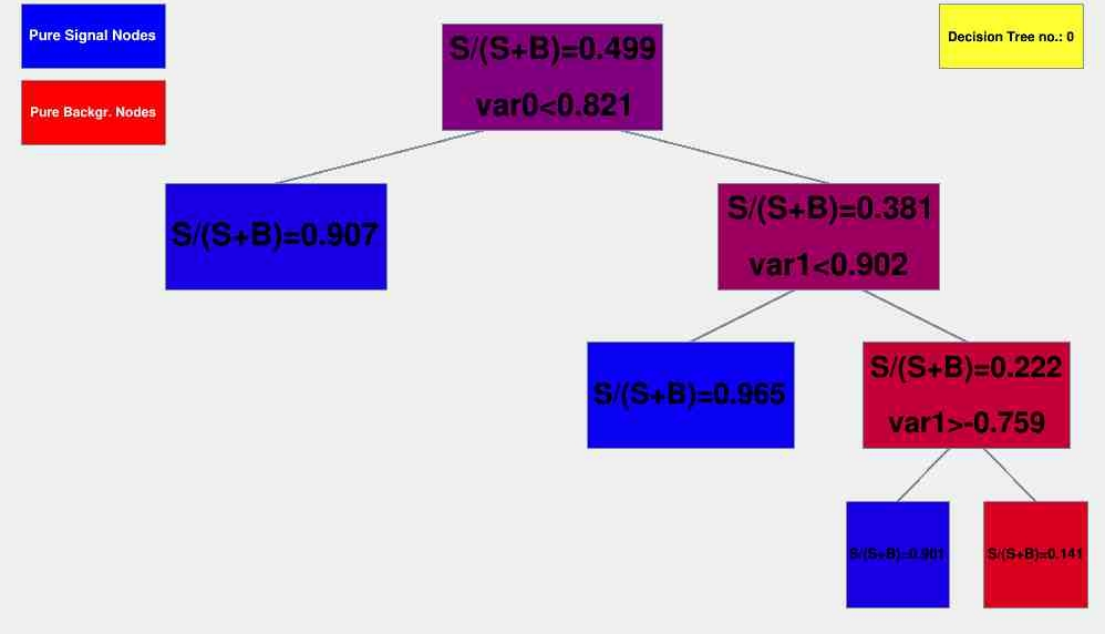
Plots von C Böser et. al ,KIT

Boosted Decision Trees

- Beispiel Halbkreise

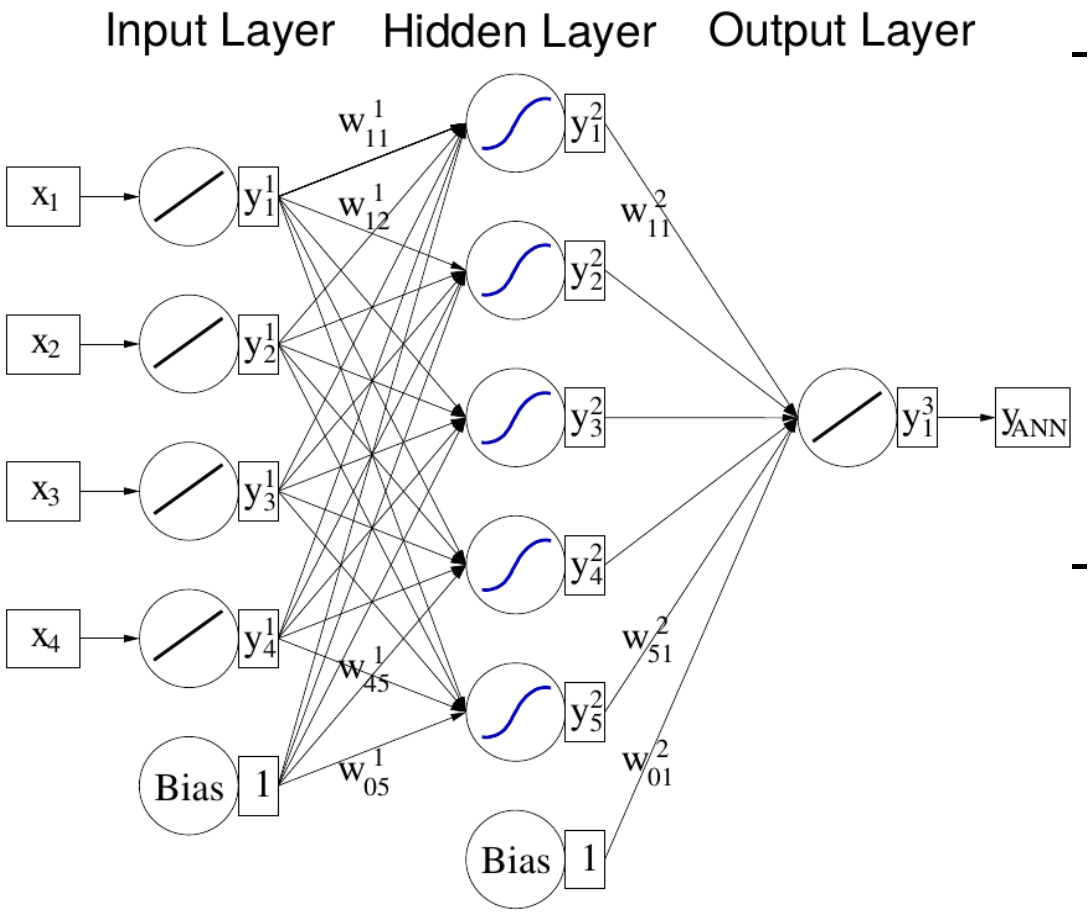






Artificial Neural Network (ANN)

- Ein weit verbreitetes Konzept ist das multi-layer perceptron
  - besteht aus Neuronen, die als input Signale oder eine Linearkombination des outputs einer vorhergehenden Menge von Neuronen verwenden. Jedes Neuron besitzt eine (nicht lineare) Aktivierungsfunktions, z.B. tanh .
  - Es werden während der Trainingsphase durch Minimieren von Fehlergrößen Gewichte bestimmt.

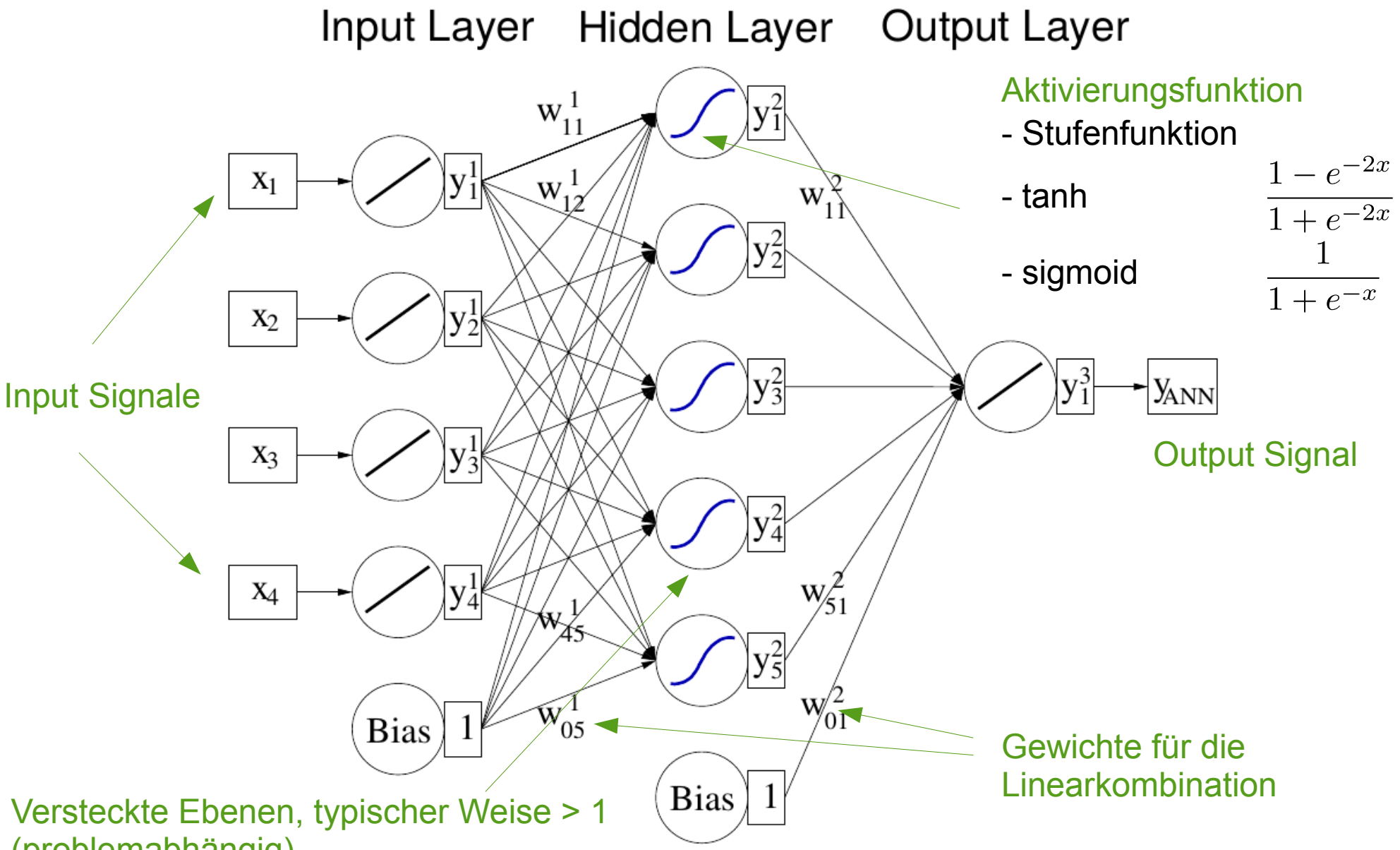


- Stellen eine Zuordnung der multidimensionalen Eingangs-signale auf einen eindimensionalen (mehrdimesionalen) output her.

- Stabiler gegen statistsche Fluktuationen im Trainingssample als BDTs.

Struktur eines ANN

- Begriffe



Struktur eines ANN

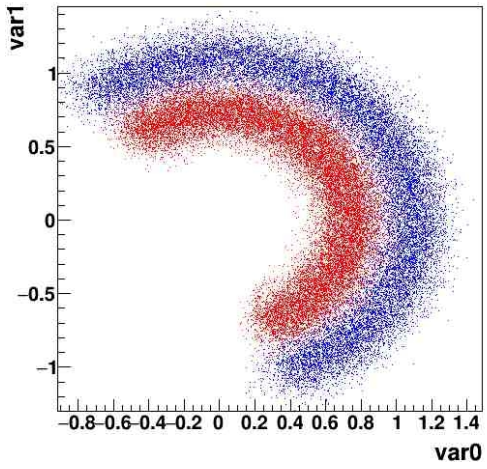
- Training
  - Jede Input Variable wird mit einem Gewichtsfaktor versehen und als Summe an die erste versteckte Ebene weitergegeben.
  - Die Input Variablen können korreliert sein, dies wird vom NN korrigiert.
  - Input Variable ohne diskrimierende Wirkung sollten nicht verwendet werden.
  - Es können auch Transformationen angewendet werden, wenn sich dadurch die diskriminierende Wirkung verbessert.
  - Jeder Knoten (Neuron) der versteckten Ebene besitzt eine Aktivierungsfunktion. Der output aller Knoten kann an weitere versteckte Ebenen mit jeweils eigenen Gewichten übergeben werden.
  - Die letzte versteckte Ebene bildet eine gewichtete Summe, den output Wert.
  - Die Gewichte werden durch die Minimierung einer Fehlergröße bestimmt, mit $y_{ANN,a}$ als output des ANN und y_a als gewünschter output des Netzes, = -1 Untergrund und =+1 für Signal.

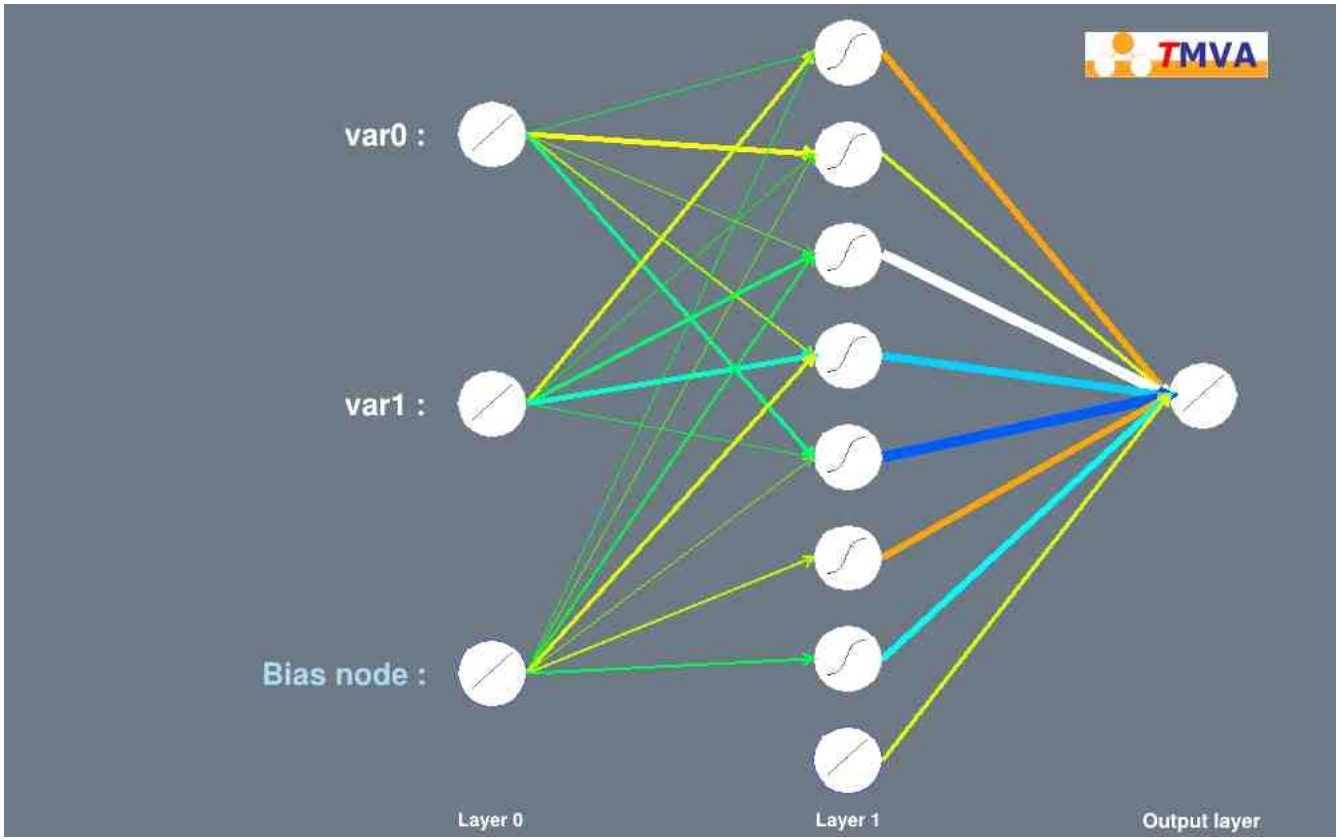
$$\sum_{a=1}^{N}(y_{ANN,a} - y_a)^2$$

  - Mit Hilfe des Gradientenverfahrens wird der Fehler minimiert und dabei die Gewichte angepasst.

Struktur eines ANN

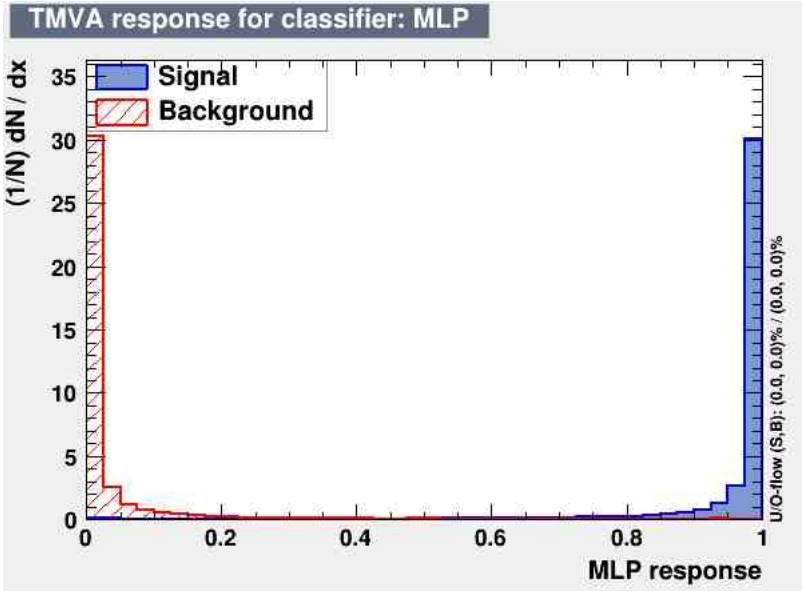
- Output
  - Der output Wert kann Real oder Integer sein. In der HEP findet man meinstens reelle Zahlen mit Werten zwischen -1 für Untergrund und +1 für Signal und mit Maxima bei ±1.
  - Die Qualität des NN hängt stark von der Struktur des NN und der Problemstellung ab.
  - Beispiel: 2 Halbkreise

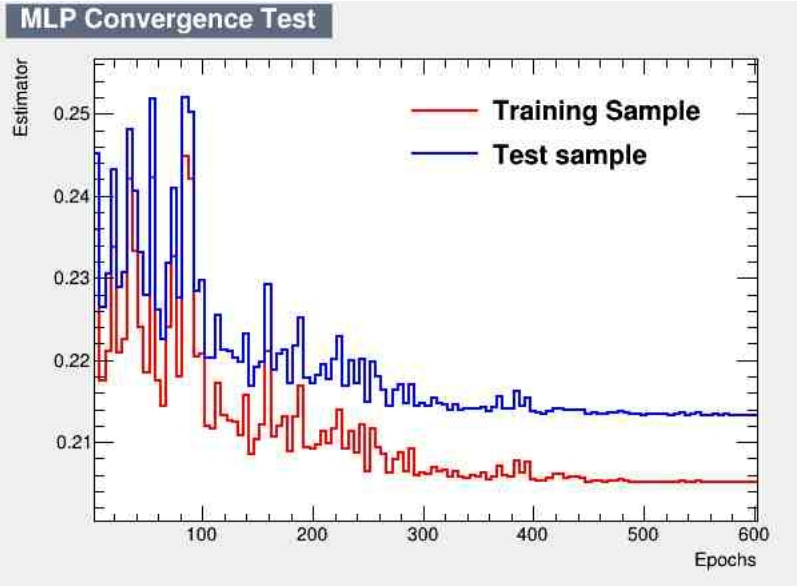




Struktur eines ANN

- Output
  - Der output Wert kann Real oder Integer sein. In der HEP findet man meinstens reelle Zahlen mit Werten zwischen -1 für Untergrund und +1 für Signal und mit Maxima bei ±1.
  - Die Qualität des NN hängt stark von der Struktur des NN und der Problemstellung ab.
  - Beispiel: 2 Halbkreise





Arbeitsvorschlag: TMVA Programm zur Trennung von Daten und Untergrund im erzeugten File `CircleData.root` 1. Teil

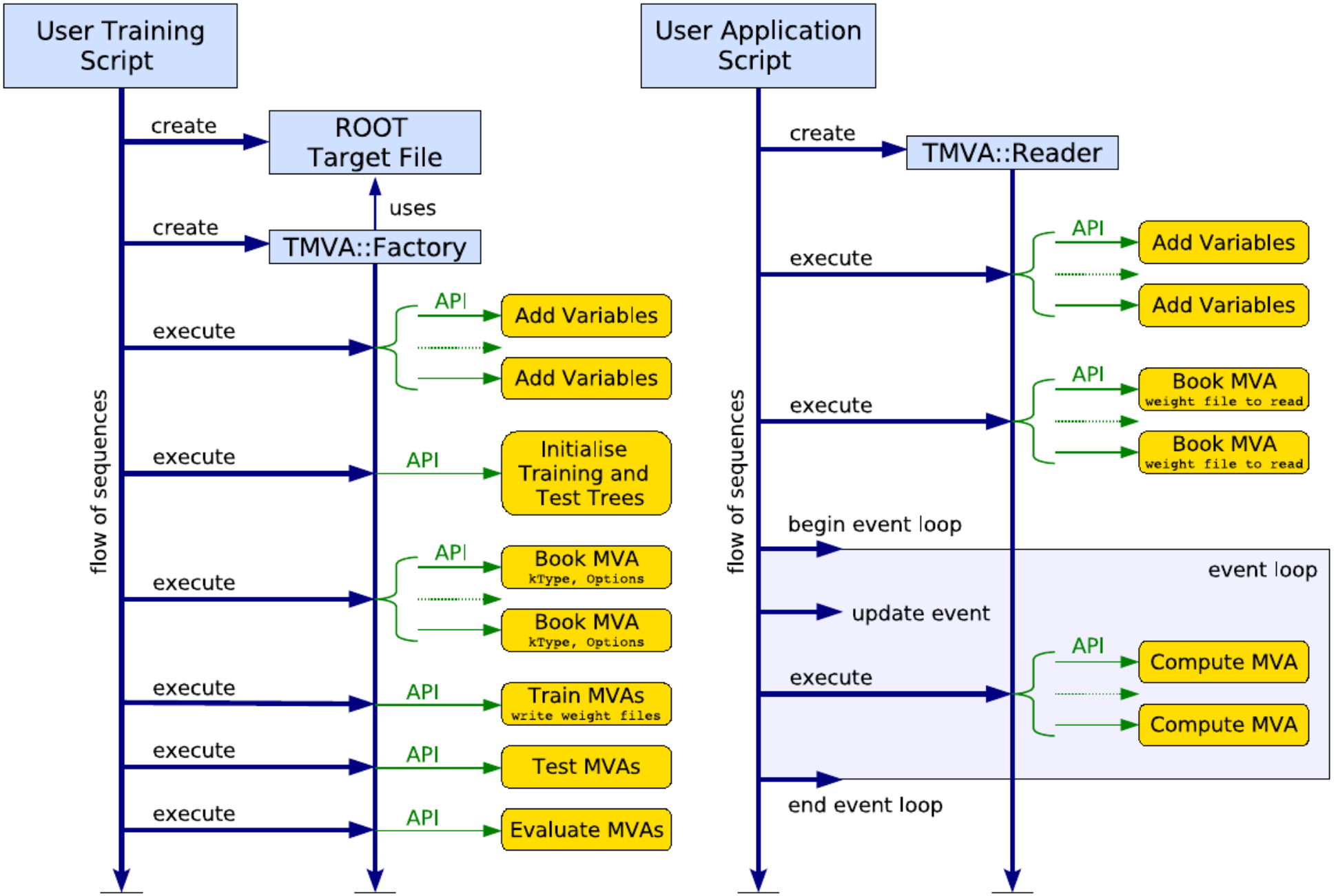
- Sie haben beide Datensätze mit verschiedenen random number seeds erzeugt. Bitte verwenden Sie die Namen `CircleData.root` und `CircleAppData.root`
  I) Laden Sie das File `tmvaCircle_0.cc` herunter und erzeugen Sie ein ausführbares Programm mit Hilfe der Instruktionen im File.
  II) Erzeugen Sie ein ROOT Output File `TMVACircle.root` und überzeugen Sie sich das es existiert.
  III) Öffnen Sie das Datenfile `CircleData.root`. Es enthält 2 Trees `TreeS` und `TreeB`. Jeder Tree enthält die Variablen `var0` und `var1` vom Typ `float`. Lesen Sie einen der beiden Trees und geben Sie die Variablen aus.

Was ist TMVA?

- Ein in ROOT integrierter Rahmen zum gleichzeitigen Ausführen und Testen von MVA Algorithmen
  - Open Source Software
  - Zugang mit C++ oder Python
  - Leicht zu verwendendes Dateninterface über ROOT trees
  - Gleiches Interface für Training und Anwendung für alle MVA Algorithmen mit den selben Daten
  - Erzeugt root File mit Kontrollverteilungen, die über ein GUI zugänglich sind
  - Nachteil: Dokumentation ist nicht aktuell.
  - Nützliche Tutorials, die die Möglichkeiten aufzeigen
    `…../root/tutorials/tmva/`
- Training session mit TMVA

from Helge Voss, TMVA team

Training und Anwendung



from the TMVA users guide

TMVA Training Software I

```
using namespace TMVA;
……..
int main(){
TString outfileName( "TMVACircle.root" );
TFile* outputFile = TFile::Open( outfileName , "RECREATE" );

TMVA::Factory* factory = new TMVA::Factory("tmvatest",
outputFile,"V:Silent:Color:DrawProgressBar:Transformations=I;D
;P;G,D:AnalysisType=Auto");

TMVA::DataLoader *dataloader=new TMVA::DataLoader("dataset");
dataloader->AddVariable( "var0", 'F' );
dataloader->AddVariable( "var1", 'F' );

TFile* input = new TFile("CircleData.root");
TTree *signal     = (TTree*)input->Get("TreeS");
TTree *background = (TTree*)input->Get("TreeB");
Double_t signalWeight = 1.0 , backgroundWeight = 1.0;

dataloader→AddSignalTree(signal,signalWeight);
dataloader→AddBackgroundTree(background,backgroundWeight);
dataloader->PrepareTrainingAndTestTree("","","nTrain_Signal=0:
nTrain_Background=0:SplitMode=Random:NormMode=NumEvents:!V");
```

TMVA output file

factory mit Optionen

daten interface

einlesen der Daten- und Background trees

tree und dataloader verknüpfen

TMVA Training Software I

```
using namespace TMVA;
.....
int main(){
TString outfileName( "TMVACircle.root" );
TFile* outputFile = TFile::Open( outfileName , "RECREATE" );
```

TMVA output file

factory mit Optionen

```
mations=I;D
```

daten interface

```
"dataset");
```

esen der Daten-
l Background
es

```
.0;
```

und dataloader
nüpfen

```
in_Signal=0:
vents:!V");
```

Option	Array	Default	Predefined Values	Description
V	–	False	–	Verbose flag
Color	–	True	–	Flag for coloured screen output (default: True, if in batch mode: False)
Transformations	–		–	List of transformations to test; formatting example: Transformations=I;D;P;U;G,D, for identity, decorrelation, PCA, Uniform and Gaussianisation followed by decorrelation transformations
Silent	–	False	–	Batch mode: boolean silent flag inhibiting any output from TMVA after the creation of the factory class object (default: False)
DrawProgressBar	–	True	–	Draw progress bar to display training, testing and evaluation schedule (default: True)
AnalysisType	–	Auto	Classification, Regression, Multiclass, Auto	Set the analysis type (Classification, Regression, Multiclass, Auto) (default: Auto)

Data processing options:
For test and visualization only

TMVA Training Software II

```
.....
 // BDT
 factory->BookMethod( dataloader, TMVA::Types::kBDT, "BDT", "!H:!
V:NTrees=850:MinNodeSize=2.5%:CreateMVAPdfs:MaxDepth=3:BoostType=AdaB
oost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction=0.5:Separat
ionType=GiniIndex:nCuts=20" );
 // fisher discriminant
 factory->BookMethod( dataloader, TMVA::Types::kFisher, "Fisher", "H:!
V:Fisher");
 // ANN
 factory->BookMethod( dataloader, TMVA::Types::kMLP, "MLP", "H:!
V:NeuronType=tanh:VarTransform=N:NCycles=600:HiddenLayers=N+5:TestRat
e=5:!UseRegulator" );
 // simple cuts
 //  factory->BookMethod( dataloader, TMVA::Types::kCuts, "Cuts","!H:!
V:FitMethod=MC:EffSel:SampleSize=200000:VarProp=FSmart" );
 factory->TrainAllMethods();
 factory→TestAllMethods();
 factory->EvaluateAllMethods();
 .........
```

Algorithmus mit Optionen

Options: TMVA users guide Seite 116

Options: TMVA users guide Seite 89

Options: TMVA users guide Seite 98

Options: TMVA users guide Seite 98

Train all MVAs

evaluate with test events

evaluate and compare performance

- Details und Optionen sind im TMVA users guide zu finden und/oder aus den Tutorialbeispielen kopierbar

TMVA Training Software III

- Test Boost Methods by Boosting the fisher algorithm

```
 ……...
 // Booking Methods
 //
 // fisher discriminant
 factory->BookMethod( dataloader, TMVA::Types::kFisher, "Fisher", "H:!
V:Fisher");
 //fisher discriminant using AdaBoost
 factory->BookMethod( dataloader,TMVA::Types::kFisher,
 "FisherBoostLog2","H:V:Fisher:Boost_Num=100:Boost_Transform=log:Boost
_Type=AdaBoost:Boost_AdaBoostBeta=2.0");
 //fisher discriminant using AdaBoost with changed parameters
 factory->BookMethod(dataloader, TMVA::Types::kFisher,"FisherBoost",
 "H:V:Boost_Num=100:Boost_Transform=step:Boost_Type=AdaBoost:Boost_Ada
BoostBeta=1.5" );
 ……...
```

for details check the users guide

Running TMVA

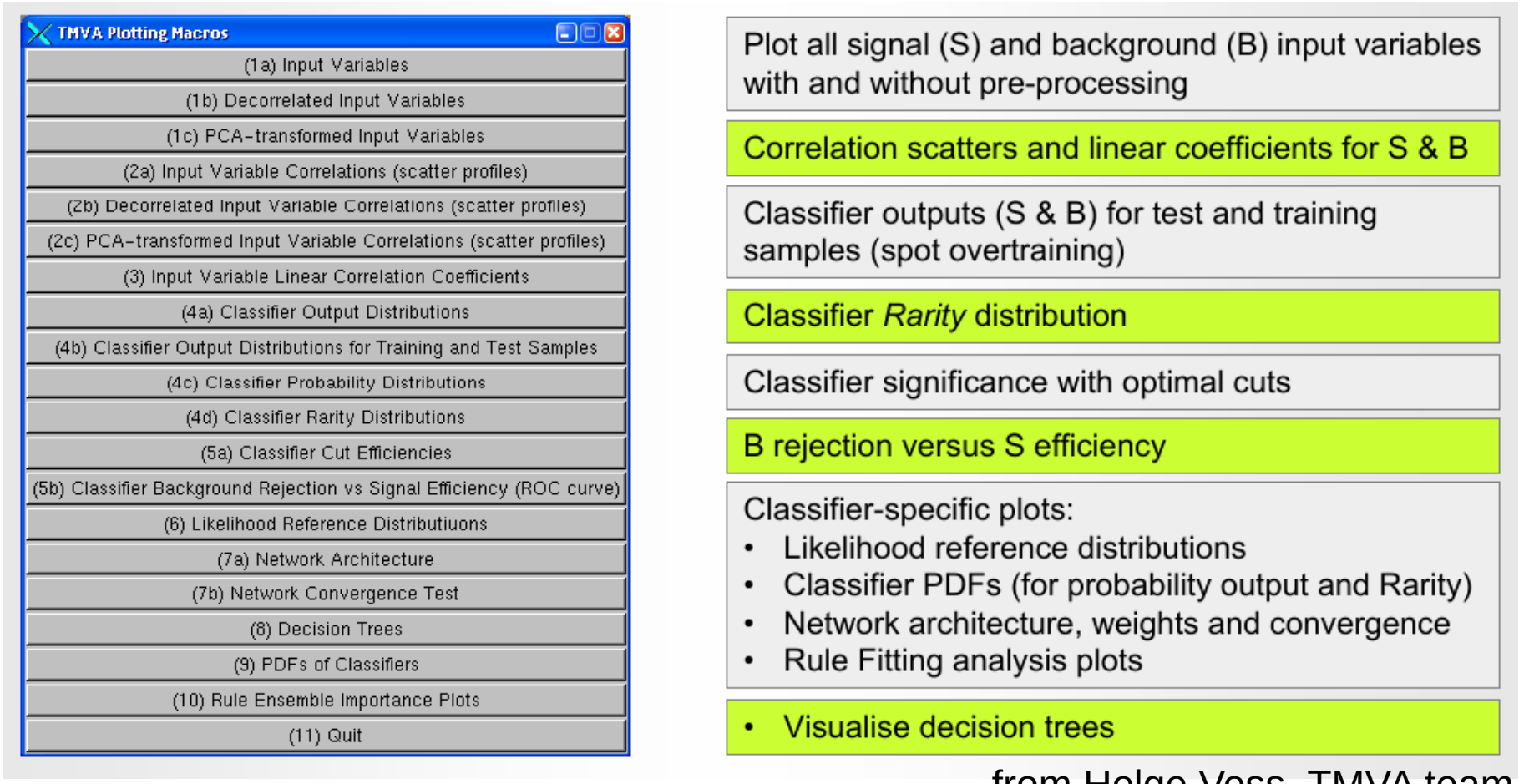
- provide data: ROOT TTree , ASCII-file or event-by-event
- choose variables (or functions ROOT Expressions thereof)
- pre-selection cuts (independent for signal and bkg)
- define global event weights for signal or background input files
- define individual event weight (any variable present in training data)
- choose splitting into training and test samples:
  - Block wise, Randomly, Periodically (i.e. periodically 3 test ev., 2 train ev., etc..)
  - User defined training and test trees
- choose pre-processing of input variables (e.g., de-correlation)
- choose classifiers(s) and it's configuration options
- train/test/evealuate

  → look at the results and diagnostics
- if happy, use trained classifier in the analysis

TMVA Training

- Im Trainingsprogramm wird ein TMVA output File erzeugt, das Kontrollplots aller verwendeter Algorithmen enthält und die mit `TMVAGui.C` angesehen werden können.

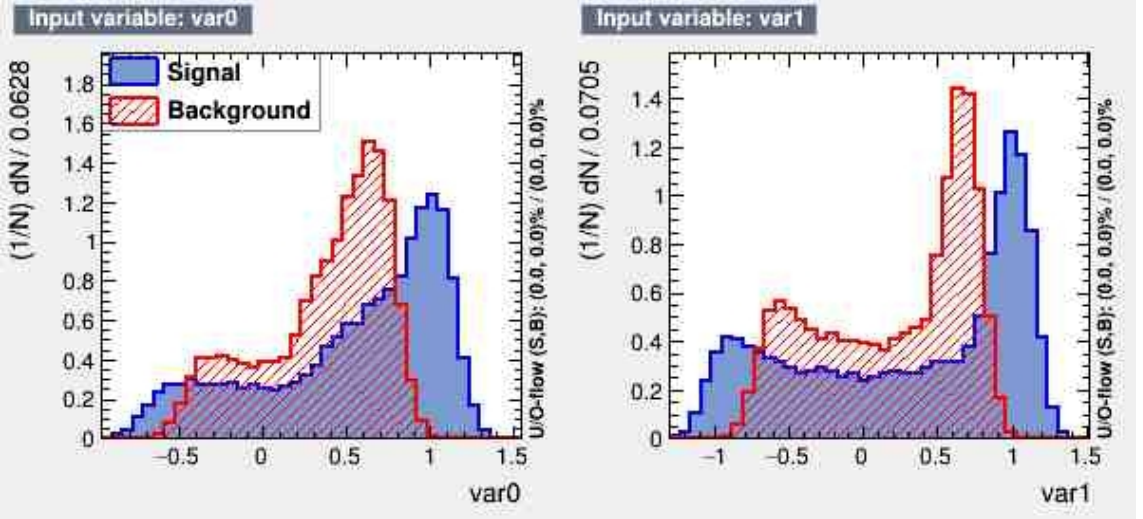
```
$> root
root [0] TMVA::TMVAGui("TMVACircle.root");
```



from Helge Voss, TMVA team

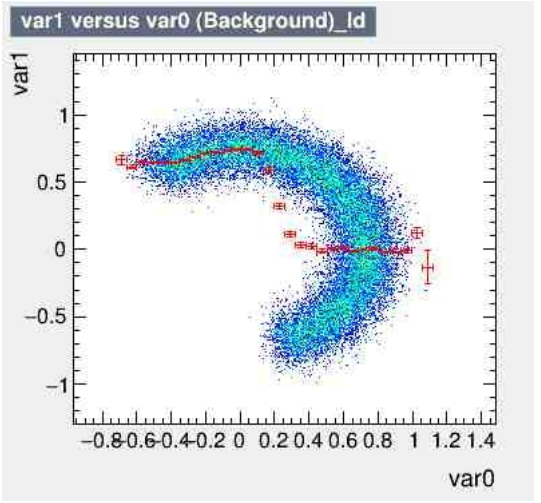
TMVA Training Plots

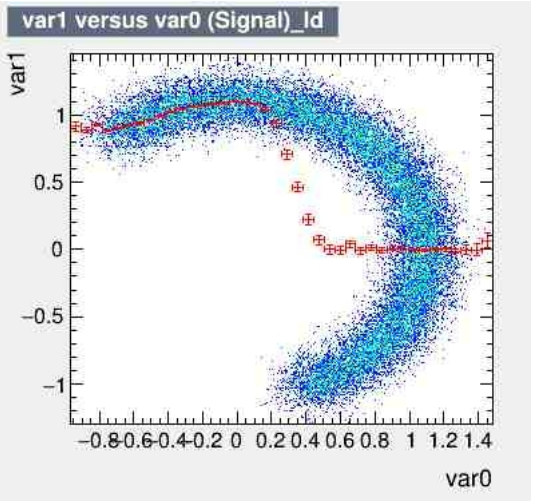
- Verteilung der Input Variablen und der Korrelationen




```
Number of training and testing events
---------------------------------------------------
Signal     -- training events             : 25000
Signal     -- testing events              : 25000
Signal     -- training and testing events: 50000
Background -- training events             : 25000
Background -- testing events              : 25000
Background -- training and testing events: 50000

Correlation matrix (Signal):
------------------------
            var0    var1
   var0:  +1.000  -0.540
   var1:  -0.540  +1.000
------------------------
Correlation matrix (Background):
------------------------
            var0    var1
   var0:  +1.000  -0.534
   var1:  -0.534  +1.000
------------------------
```



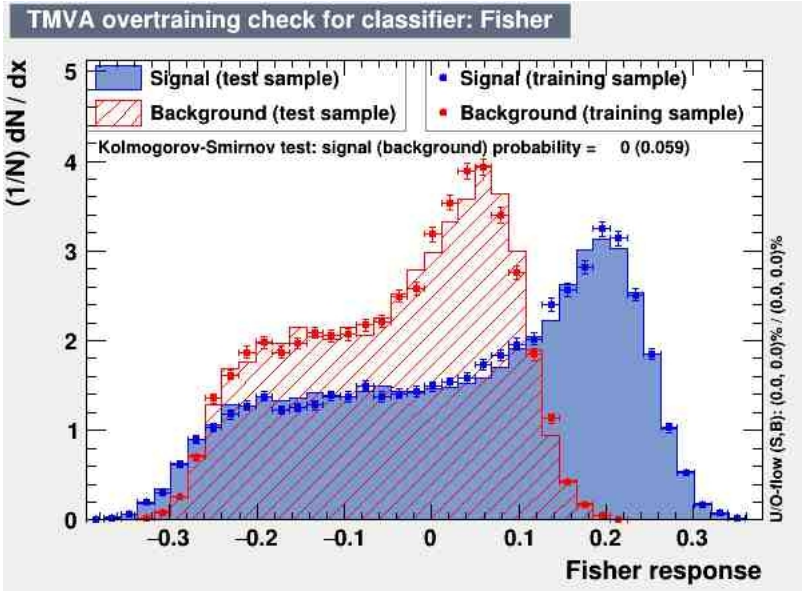


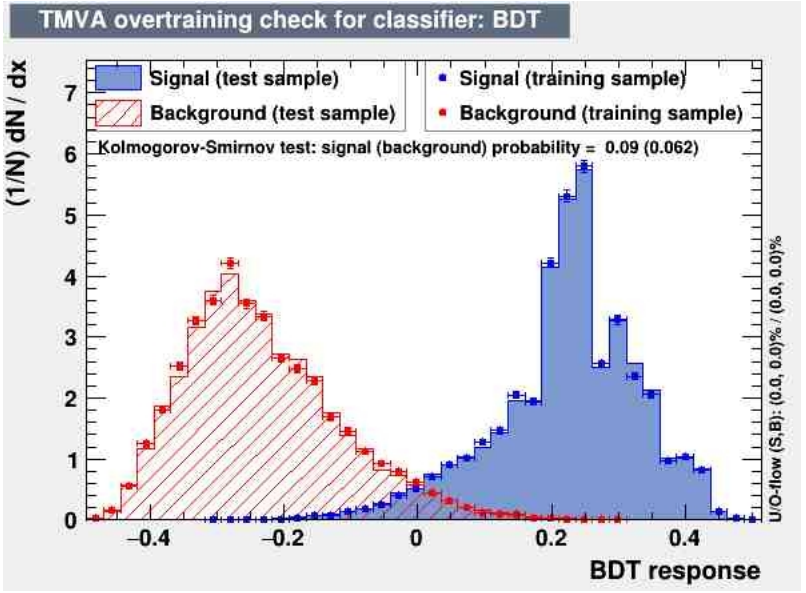

```
Preparing the Decorrelation transformation...
Variable        Mean        RMS   [        Min        Max ]
-----------------------------------------------------------
    var0:      1.2340     1.0000   [    -1.8385     3.5905 ]
    var1:     0.70552     1.0000   [    -2.0290     2.6522 ]
-----------------------------------------------------------
Preparing the Principle Component (PCA) transformation...
Variable        Mean        RMS   [        Min        Max ]
-----------------------------------------------------------
    var0:-2.6924e-11     0.70115   [    -1.4452     1.4484 ]
    var1:-4.4720e-11     0.36638   [   -0.98092    0.97001 ]
-----------------------------------------------------------
Preparing the Gaussian transformation...
Preparing the Decorrelation transformation...
Variable        Mean        RMS   [        Min        Max ]
-----------------------------------------------------------
    var0:  0.0046163     1.0000   [    -3.4034     6.2204 ]
    var1:  0.0046474     1.0000   [    -3.7156     6.1234 ]
-----------------------------------------------------------
```

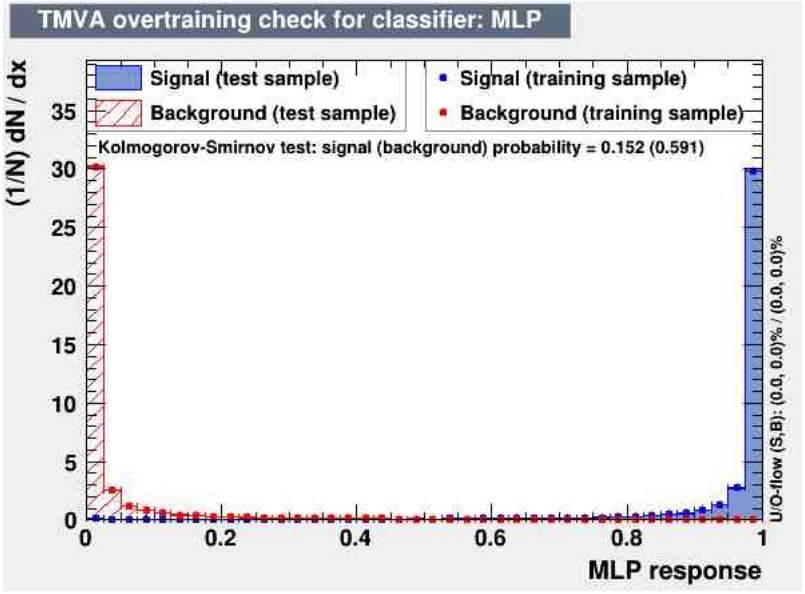
- Es stehen verschiedene Methoden zur Dekorrelation zur Verfügung (mit Plots)

TMVA Training Plots

- Classifier Output







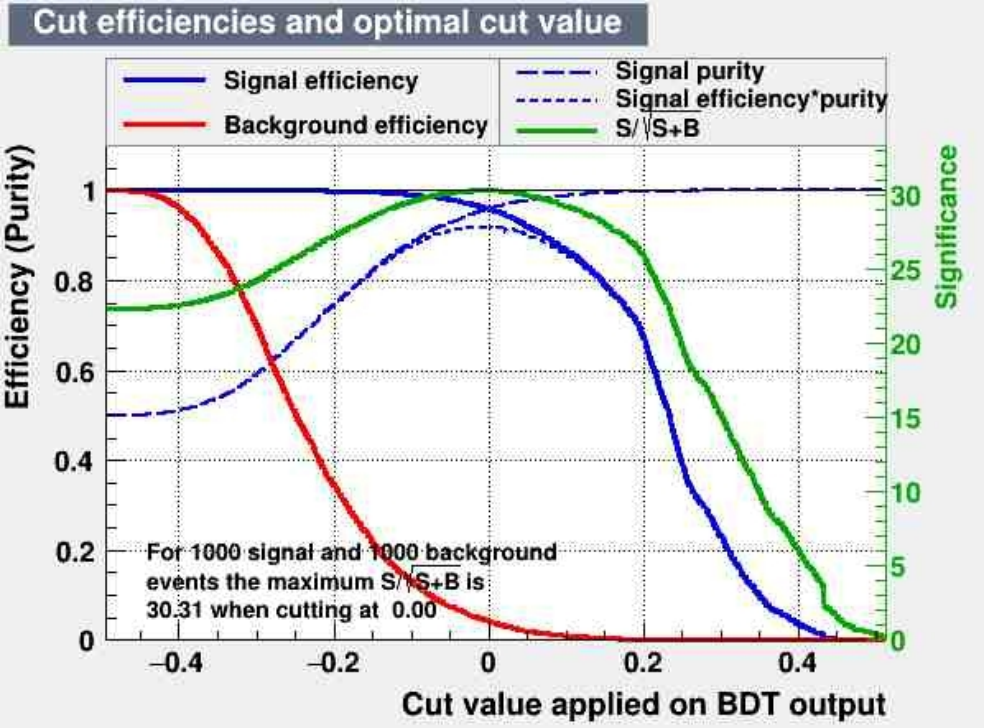
Hier werden test und trainingssample in einer Abbildung dargestellt. Weichen beide voneinander ab, ist das ein Zeichen für overtraining.

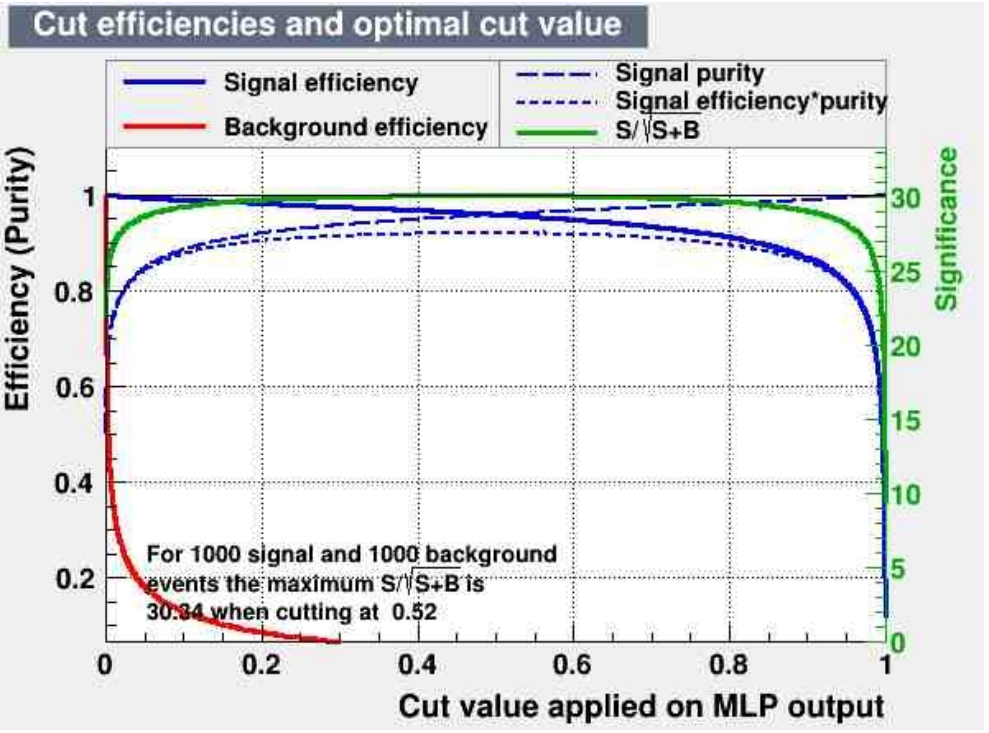
Hier läßt sich schon qualitativ sehen, welcher Algorithmus besser geeignet ist.

TMVA Training Plots

- Welcher Arbeitspunkt ist der Beste für meinen Algorithmus?

  - Suche nach dem Maximum in S/sqrt(S+B). Abhängig vom Problem werden auch andere Größen optimiert.

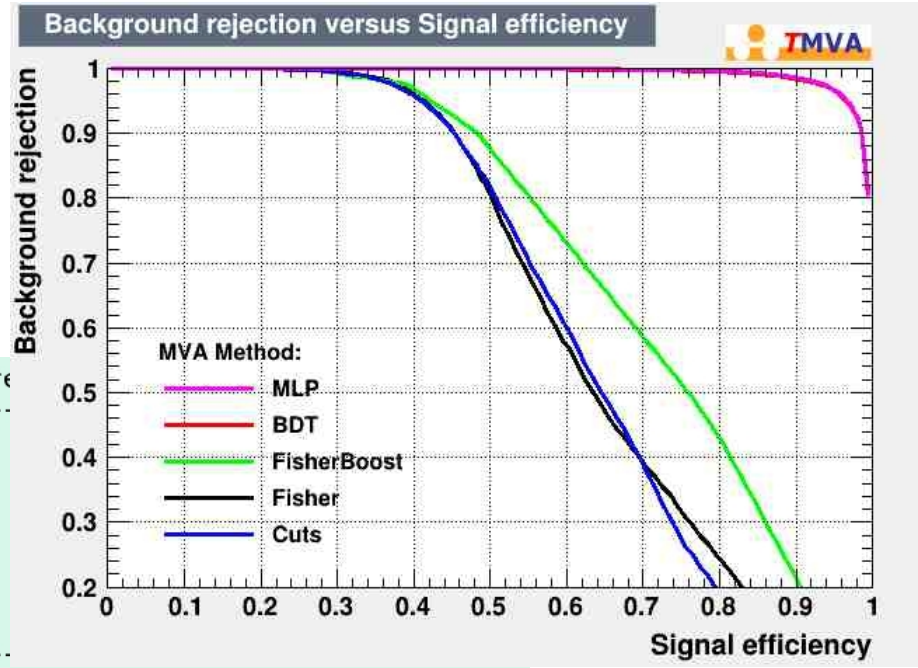




TMVA Training Plots

- Welcher Algorithmus Beste für mein Problem?

  - Reciever Operations Characterics (ROC) curve
    Je größer die Fläche unter der Kurve je besser ist der Algorithmus




```
Evaluation results ranked by best signal efficiency and purity (are
-------------------------------------------------------------------
DataSet       MVA
Name:         Method:          ROC-integ
dataset       MLP            : 0.993
dataset       BDT            : 0.992
dataset       FisherBoost    : 0.731
dataset       Fisher         : 0.656
dataset       Cuts           : 0.648
-------------------------------------------------------------------

Testing efficiency compared to training efficiency (overtraining check)
-------------------------------------------------------------------------------------------
DataSet              MVA              Signal efficiency: from test sample (from training sample)
Name:                Method:          @B=0.01             @B=0.10            @B=0.30
-------------------------------------------------------------------------------------------
dataset              MLP            : 0.874 (0.886)       0.986 (0.985)      1.000 (1.000)
dataset              BDT            : 0.862 (0.883)       0.986 (0.985)      1.000 (1.000)
dataset              FisherBoost    : 0.313 (0.314)       0.483 (0.482)      0.621 (0.621)
dataset              Fisher         : 0.332 (0.337)       0.453 (0.454)      0.542 (0.544)
dataset              Cuts           : 0.327 (0.342)       0.453 (0.457)      0.552 (0.555)
-------------------------------------------------------------------------------------------
```

Arbeitsvorschlag: TMVA Programm zur Trennung von Daten und Untergrund im erzeugten File `CircleData.root` 2. Teil

- Schreiben Sie ein TMVA Trainingsscript `tmvaCircle.cc` , das mit dem Datensatz `CircleData.root` Signal und Untergrund trennt.
  - I) Erweitern Sie dazu Ihre Datei `tmvaCircle_I.cc` und verwenden Sie zunächst den Fischer Algorithmus. Die Optionen finden Sie im TMVA users guide oder Sie können die BDT Zeile aus dem TMVA Tutorial Bereich in dem Programm `TMVAClassification.C` verwenden.
  - II) Öffnen Sie das TMVA ROOT output file und sehen Sie die Kontrollplots mit dem TMVAGui an.
  - III) Fügen Sie nun die Algortihmen für ein ANN und BDT hinzu. Welcher Algorithmus ist der Beste?
  - IV) Erweitern Sie den Fisher Algorithmus um die AdaBoost Methode und vergleichen das Ergebnis mit Hilfe des TMVAGui.

TMVA Anwendungssoftware

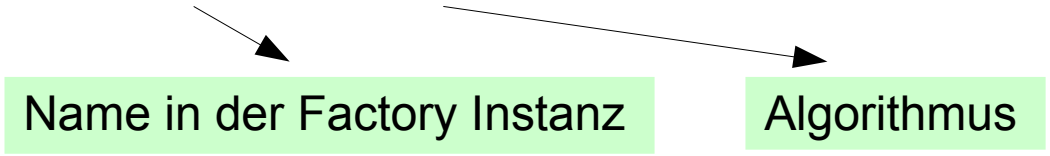
- Wie können wir nun die Trainingsergebnisse auf unsere Analyse Programme anwenden?

  - Der Trainingsprozess erzeugt ein directory `dataset` im Arbeitsdirectory mit Unterdirectories `plots` und weights

  - TMVAGui greift auf plots zu und speichert dort alle Abbildungen die produziert wurden.

  - Im direcetory `weights` stehen für jeden verwendeten Algorithmus xml Files und jeweils eine Klasse, die zum Lesen der Gewichte und Parameter durch den TMVA Reader benötigt werden.

```
$> ls dataset/weights
$> tmvaTest_FisherBoost.class.C
   tmvaTest_FisherBoost.weights.xml
   tmvaTest_MLP.class.C
   tmvaTest_MLP.weights.xml
```



- Mit der TMVA Reader Instanz werden nach einer Instanzierung des Algorithmus die Daten aus dem Tree gelesen und dann die Output Variable des Algorithmus bestimmt. Auf die Outputvariable kann dann zur Selektion geschnitten werden. Die Tree Struktur muss mit der im Training übereinstimmen oder umkopiert werden.

TMVA Anwendungssoftware I

```
…...
using namespace TMVA;

int main(){
// Create the Reader object
   TMVA::Reader *reader = new TMVA::Reader( "!Color:!Silent" );

// Declare the set of variables
   Float_t var0, var1 ;
   reader->AddVariable( "var0",&var0 );
   reader->AddVariable( "var1",&var1 );

// Spectator variables, if declared in training

// Book method(s)
   reader→BookMVA("MLP",”dataset/weights/tmvaTest_MLP.weights.xml"));

// Book output histograms contains the probability
   TH1F *histNn = new TH1F( "MVA_MLP","MVA_MLP",120,-0.1, 1.1 );

// Dataset with the data to be loaded
   TFile* input = new Tfile("CircleAppData.root");

// Prepare the event tree
   TTree* theTree = (TTree*)input->Get("TreeS");
   Float_t var0, var1;
   theTree->SetBranchAddress( "var0", &var0 );
   theTree->SetBranchAddress( "var1", &var1 );
```

TMVA Anwendungssoftware II

```
.....
// Event loop
   Float_t yNN;
   for (Long64_t ievt=0; ievt<theTree->GetEntries();ievt++) {
      if (ievt%1000 == 0)
        std::cout << "--- ... Processing event: " << ievt << std::endl;
// Get tree entry
      theTree->GetEntry(ievt);
// Output of NN
      yNN  =  reader→EvaluateMVA("MLP");
      // Fill Output of the NN in Histogramm
      histNn→Fill(yNN);
   } // End Event loop
// Write histograms
   TFile *target  = new TFile( "TMVApp.root","RECREATE" );
   histNn→Write();
// Clean up
   delete reader;
   target->Close();

   return 0 ;
}
```

Arbeitsvorschlag: TMVA Programm zur Trennung von Daten und Untergrund im erzeugten File `CircleData.root` 3. Teil

- Schreiben Sie ein einfaches TMVA Anwendungsprogramm `tmvaCircleApp.cc`, das mit dem Datensatz `CircleDataApp.root` Signal und Untergrund trennt. Wir haben immer noch einen Datensatz, der je nach Tree Namen, Signal und Untergrund getrennt enthält. Lesen Sie den Signal Datensatz und bestimmen Sie den Anteil, der als Untergrund identifiziert wird und umgekehrt.
  I) Erweitern Sie dazu die Datei `tmvaCircleApp_0.cc`. Dazu suchen Sie zunächst das weight file eines von Ihnen verwendeten Algorithmus, z.B. BDT.
  II) Fügen Sie nun die Reader Instanzen ein, öffnen und lesen Sie die entsprechenden Trees.
  III) Fügen Sie zwei 2D Histogramme hinzu, in denen für Signalereignisse jeweils Signal und Untergrund zu finden ist. Dann zwei für Untergrundereignisse. Schreiben Sie ein ROOT File mit den Histogrammen.
  IV) Stellen Sie mit dem TBrowser Ihre Histogramme dar.

Generalities for (T)MVA Analyses

- There is no magic in MVA-Methods:
  - "black boxes" ? → they are not sooo hard to understand
  - you typically still need to make careful tuning and do some "hard work"
  - no "artificial intelligence" … just "fitting decision boundaries" in a given model
- The most important thing at the start is finding good observables
  - good separation power between S and B
  - little correlations amongst each other
  - watch correlation between selection variables and the parameters you try to measure!
- Think also about possible combination of variables
  - this may allow you to eliminate correlations
    - rem.: you are MUCH more intelligent than what the algorithm will do

Generalities for (*T*)MVA Analyses

- Apply pure preselection cuts and let the MVA only do the difficult part.
- "Sharp features should be avoided" → numerical problems, loss of information when binning is applied
  - simple variable transformations (i.e. log(variable)) can often smooth out these areas and allow signal and background differences to appear in a clearer way

Welche MVA Methode

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	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	⋆[39]	⋆	⋆⋆
	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.

Taken from the TMVA users guide

Zusammenfassung

- Mit TMVA steht ein in ROOT integriertes Werkzeug mit verschiedenen MVA Algorithmen zur Verfügung.
  - Unterstützung der Trainingsphase und Auffinden / Vergleichen des für das Problem am besten geeigneten Algorithmus bei bekanntem Trainingssample.
  - Anwendung der im Training bestimmten Parameter auf das unbekannte Sample.

- Sie haben mit einem einfachen Beispiel gelernt TMVA mit in ROOT gespeicherten Daten zu verwenden.
  - Sie haben sowohl ein Trainingsprogramm als auch ein Anwendungsprogamm geschrieben.
  - Sie können Ihre gemessenen Daten in ROOT Trees speichern und haben damit einen einfachen Zugang zu MVA Algorithmen.

- Viel Spass beim Ausprobieren eines eigenen kleinen Beispiels.