神经网络

王思广

北京大学物理学院

讲课内容

- 多层感知器MLP（Multi-Layer Perceptron）
  - 对类型进行识别
  - 对连续函数进行描述
- 多变量分析TMVA（Multi-Variate-Analysis类）

siguang@pku.edu.cn

对类型进行识别

siguang@pku.edu.cn

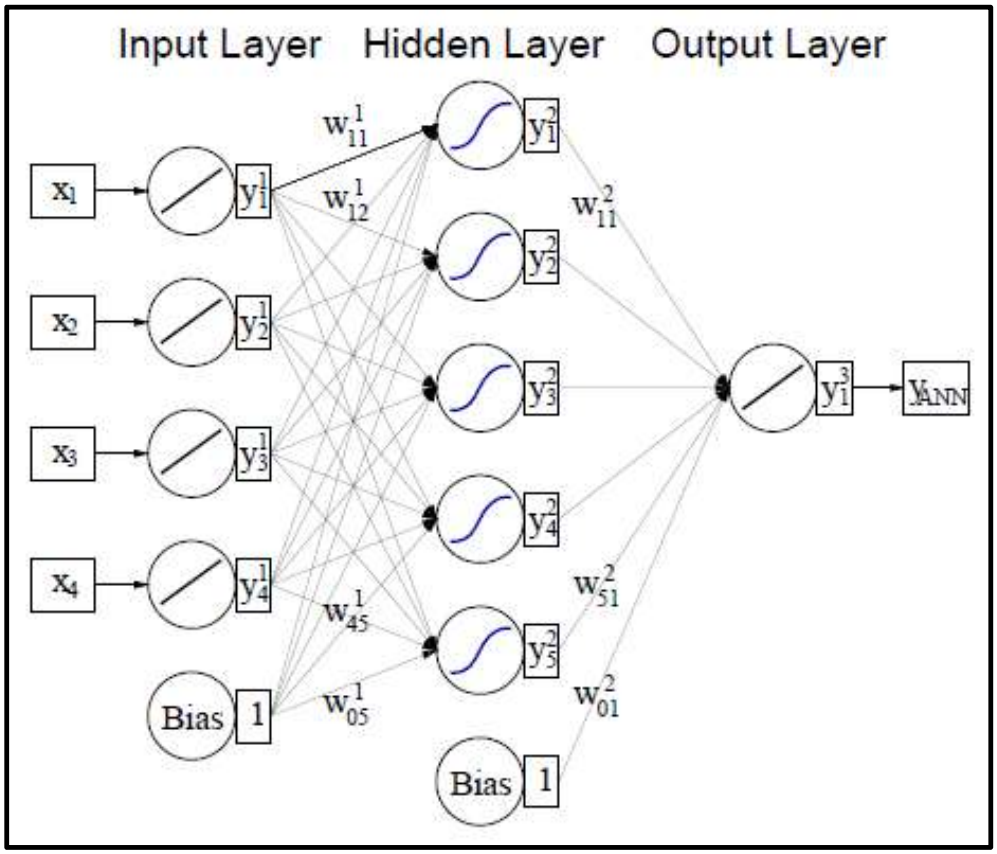
人的识别能力

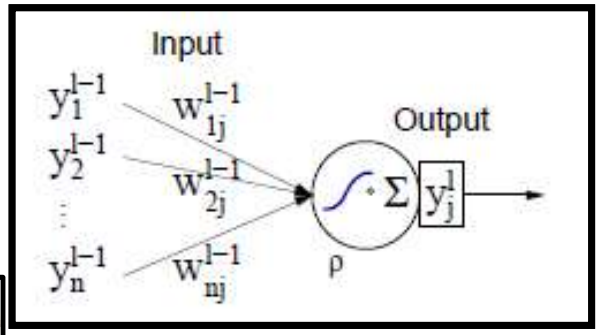
- 校园中，大家识别男、女同学的准确率应该很高。我常用的依据如下：
  - 衣服：颜色艳丽的是女同学的概率大
  - 头发：头发长的是女同学的概率大
  - 说话：音调高的是女同学的概率大
  - 。。。。

将以上所有概率在心中默默加合在一起，最后给出判断。

- 神经网络就是要模拟人的这个过程

一个隐藏层的多层感知器MLP工作示意图





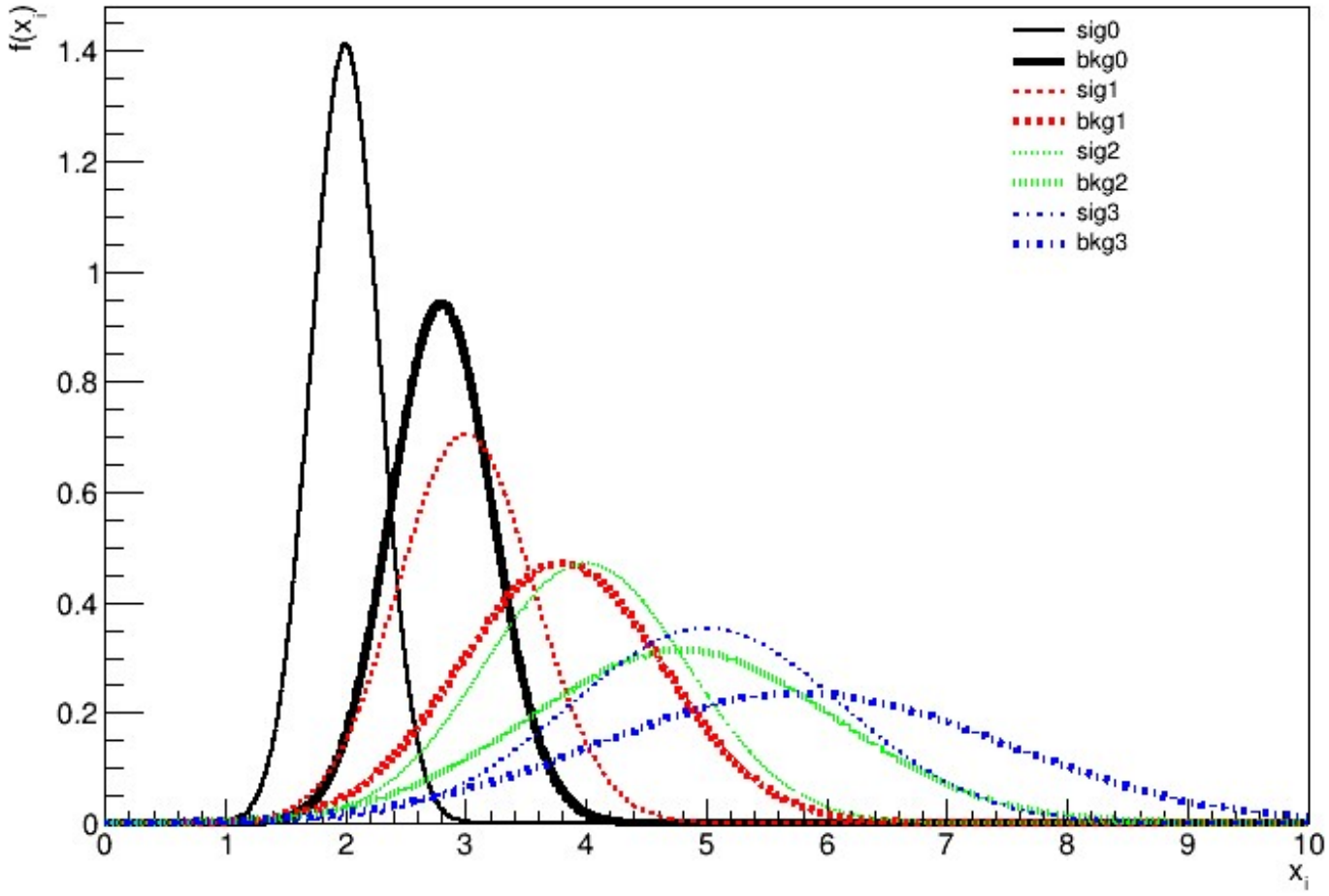
$$\rho = \alpha \circ \kappa$$

$$\kappa: (y_1^{(\ell)}, .., y_n^{(\ell)} | w_{0j}^{(\ell)}, .., w_{nj}^{(\ell)}) \rightarrow \begin{cases} w_{0j}^{(\ell)} + \sum_{i=1}^{n} y_i^{(\ell)} w_{ij}^{(\ell)} & \textit{Sum,} \\ w_{0j}^{(\ell)} + \sum_{i=1}^{n} \left(y_i^{(\ell)} w_{ij}^{(\ell)} \right)^2 & \textit{Sum of squares,} \\ w_{0j}^{(\ell)} + \sum_{i=1}^{n} |y_i^{(\ell)} w_{ij}^{(\ell)}| & \textit{Sum of absolutes,} \end{cases}$$

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

siguang@pku.edu.cn

产生训练神经网络的数据



- 分为两类：
  - 细线表示信号
  - 粗线表示本底
  - 如果产生信号，用细的4个函数分别产生一个随机数赋给变量1到4
  - 如果产生本底，用粗的4个函数分别产生一个随机数赋给变量1到4
- 具体函数形式见代码

siguang@pku.edu.cn

数据产生实现代码（createData.C）

```
#include "TMultiLayerPerceptron.h"
#include "TRandom3.h"
#include "TFile.h"
#include "TTree.h"
#include "TF1.h"
#include "TSystem.h"
#include "TMath.h"
#include "TROOT.h"
#include "TLegend.h"
#include "TCanvas.h"

void createData(Int_t tot=50000,Int_t nSeed=2){
  const Int_t Nx = 4;
  TF1 *fsig[Nx];
  TF1 *fbkg[Nx];
  const Double_t xmin=0;
  const Double_t xmax=10;
  const Double_t sqPi = sqrt(TMath::Pi());
  for(Int_t i=0; i<Nx; i++){
    fsig[i] = new TF1(Form("fsig%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
    fbkg[i] = new TF1(Form("fbkg%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
```

数据产生实现代码（createData.C 续）

```
    Double_t MeanS = i+2.0;
    Double_t MeanB = i+2.8;
    Double_t sigmaS = (i+1.)*0.4;
    Double_t sigmaB = (i+1.)*0.6;
    fsig[i]->SetParameters(1/(sqPi*sigmaS),MeanS,sigmaS);
    fbkg[i]->SetParameters(1/(sqPi*sigmaB),MeanB,sigmaB);
    fsig[i]->SetNpx(10000);
    fbkg[i]->SetNpx(10000);
    fsig[i]->SetLineColor(i+1);
    fbkg[i]->SetLineColor(i+1);
    fsig[i]->SetLineStyle(i+1);
    fbkg[i]->SetLineStyle(i+1);
    fsig[i]->SetLineWidth(2);
    fbkg[i]->SetLineWidth(4);
}
```

数据产生实现代码（createData.C 续）

```
TCanvas *c1 = new TCanvas("c1","");
c1->SetTopMargin(0.03);
c1->SetRightMargin(0.03);
TLegend *lg = new TLegend(0.75,0.7,0.96,0.96);
for(Int_t i=0; i<Nx; i++){
  if(i==0){
    fsig[i]->Draw();
    fsig[i]->SetTitle(";x_{i};f(x_{i})");
  }else{
    fsig[i]->Draw("same");
  }
  fbkg[i]->Draw("same");
  lg->AddEntry(fsig[i],Form("sig%d",i),"l");
  lg->AddEntry(fbkg[i],Form("bkg%d",i),"l");
}
lg->SetFillColor(0);
lg->SetLineWidth(0);
lg->SetLineColor(0);
lg->Draw();
```

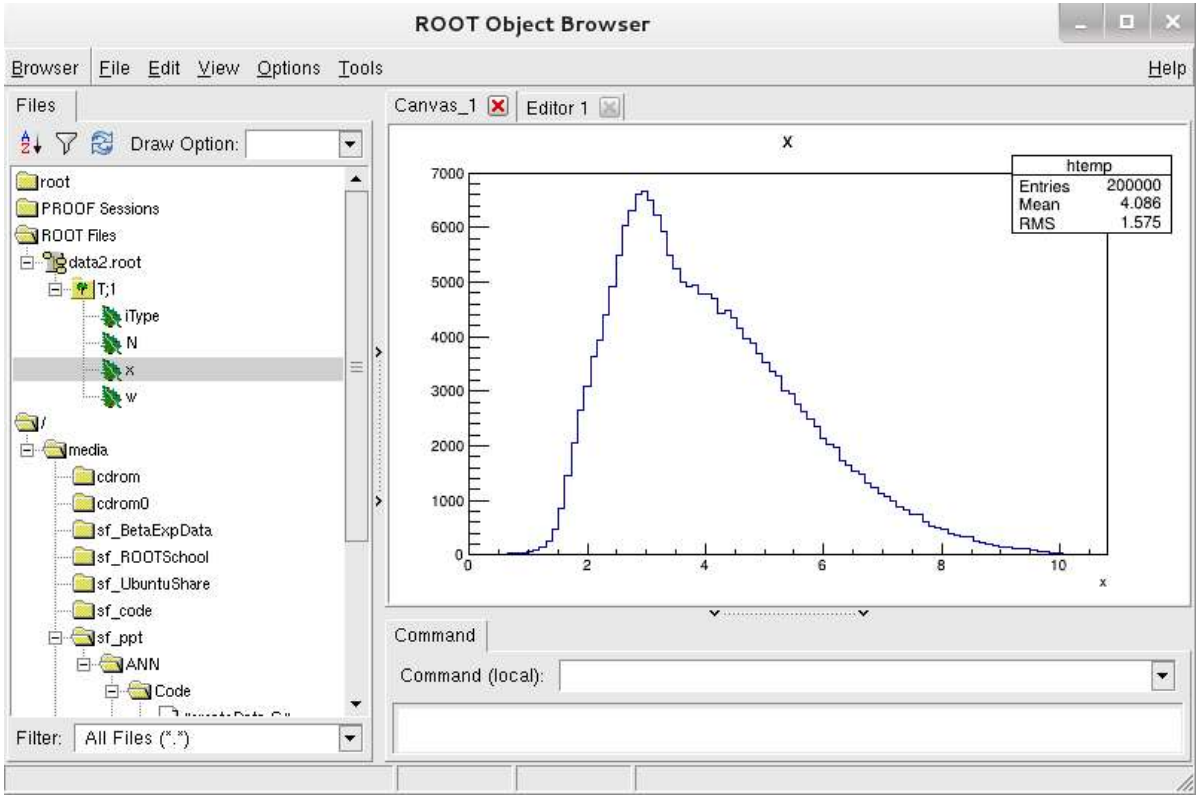
数据产生实现代码（createData.C 续）

```
//create data
TFile *fout = new TFile(Form("data%d.root",nSeed),"recreate");
TTree *T = new TTree("T","Tree for Sig and Background");
Double_t x[Nx];
Int_t N;
Int_t iType;
Double_t w;

T->Branch("iType",&iType,"iType/I");
T->Branch("N",&N,"N/I");
T->Branch("x",x,"x[N]/D");
T->Branch("w",&w,"w/D");
N = Nx;
gRandom->SetSeed(nSeed);
```

数据产生实现代码（createData.C 续）

```
  for(Int_t i=0; i<tot; i++){
    if(gRandom->Rndm()<0.75) iType = 0; //background 75% weight=0.25
    else iType = 1; //signal
    if(iType==1) {
      w = 0.75;  // 25% signal, 1 event with 75% weight
      for(Int_t j=0; j<N; j++){
        x[j] = fsig[j]->GetRandom();
      }
    }else{
      w = 0.25;  // 75% background, 1 event with 25% weight
      for(Int_t j=0; j<N; j++){
        x[j] = fbkg[j]->GetRandom();
      }
    }
    T->Fill();
  }
  T->Write();
  fout->Close();
}
```

训练数据文件的内容检查



练习：

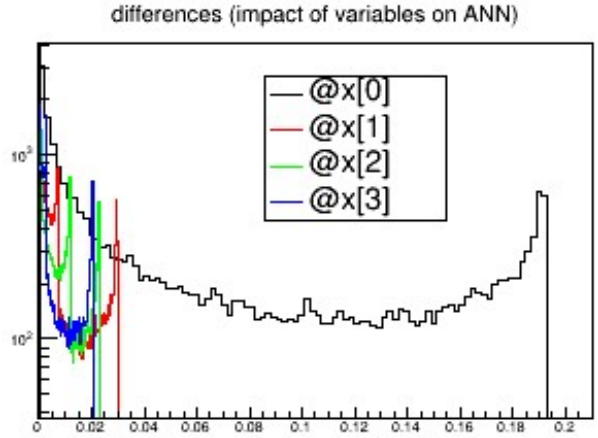
```
>root data2.root
Root[1]new TBrowser
root [2] T->Draw("x")
root [3] T->Draw("x[0]")
root [4] T->Draw("x[0]","iType==0")
root [5] T->Draw("x[0]","iType==1")
root [6] T->Draw("x[0]:x[1]","iType==1")
root [7] T->Draw("x[0]:x[1]","iType==1","surf")
root [8] T->Draw("x[0]:x[1]","iType==1","surf1")
root [9] T->Draw("x[0]:x[1]","iType==1","surf2")
root [10] T->Draw("x[0]:x[1]","iType==1","surf3")
root [11] T->Draw("x[0]:x[1]","iType==1","surf4")
root [12] T->Draw("x[0]:x[1]","iType==1","box")
root [13] T->Draw("x[0]:x[1]:x[3]","iType==1")
root [14] .q
```

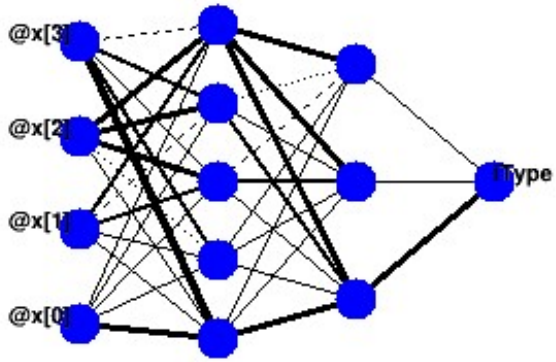
对类型进行识别
之
训练神经网络

siguang@pku.edu.cn

MLP神经网络结构及训练输出

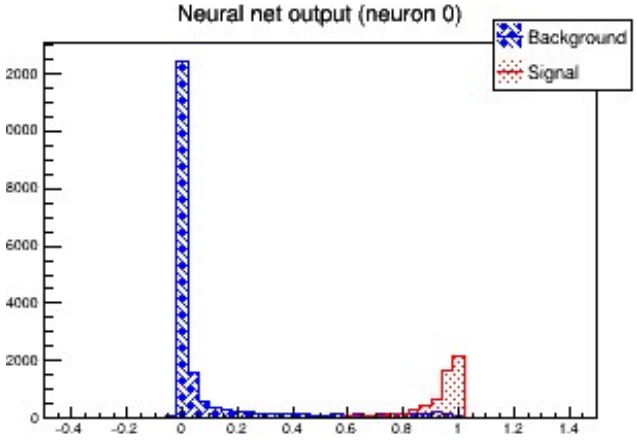
各参量对最后结果的影响

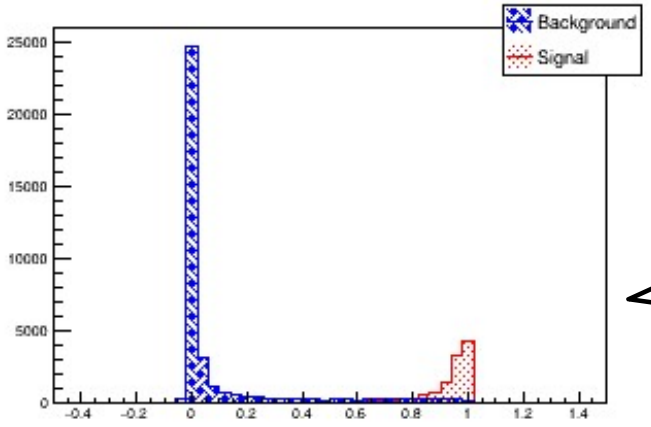




神经网络的结构。线的粗细反映权重大小

训练后的神经网络对测试数据的输出结果





训练后的神经网络对训练及测试数据的输出结果

siguang@pku.edu.cn

MLP神经网络训练代码（toTrain.C）

```
#include "TMultiLayerPerceptron.h"
#include "TMLPAnalyzer.h"
#include "TRandom3.h"
#include "TFile.h"
#include "TTree.h"
#include "TF1.h"
#include "TSystem.h"
#include "TMath.h"
#include "TROOT.h"
#include "TLegend.h"
#include "TCanvas.h"
#include "TH1F.h"
#include "THStack.h"

//Note: part of code from mlpHiggs.C under ROOT's tutorials/mlp
void toTrain(Int_t NumTrain=1000){
  if (!gROOT->GetClass("TMultiLayerPerceptron")) {
    gSystem->Load("libMLP");
  }

  TFile *f = new TFile("../createDataForTrain/data2.root"); //where the data is
  TTree *T;
  f->GetObject("T",T);
```

MLP神经网络训练代码（toTrain.C 续）

```
//Declaration of leaves types
Int_t           iType;
Int_t           N;
Double_t        x[4];
Double_t w; //weight of each event
// Set branch addresses.
T->SetBranchAddress("iType",&iType);
T->SetBranchAddress("N",&N);
T->SetBranchAddress("x",x);
T->SetBranchAddress("w",&w);
```

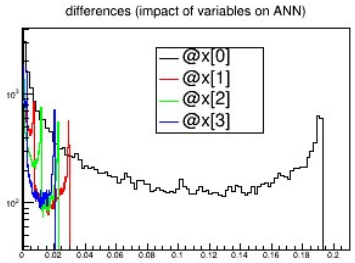
```
TMultiLayerPerceptron *mlp =
  new TMultiLayerPerceptron("@x[0],@x[1],@x[2],@x[3]:5:3:iType","w",T,"Entry$%2","(Entry$+1)%2");
//@--- to normalizate the variation before taking part in the training.
//5:3---number of neurons in each layers
//"w"---weight
//T---TTree pointer
//"Entry$%2" and "(Entry$+1)%2"--training and test list

mlp->Train(NumTrain, "text,graph,update=10");
mlp->Export("TClassANN","C++");
```

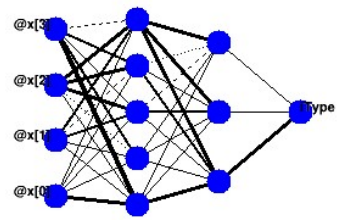
核心3句话！

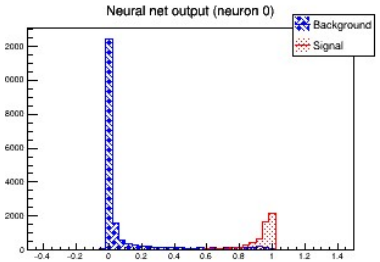
MLP神经网络训练代码（toTrain.C 续）

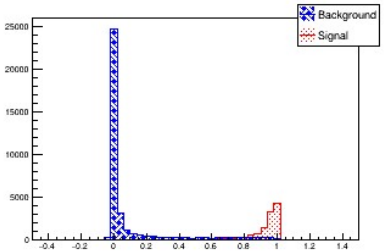
```
// Use TMLPAnalyzer to check more inforamtion
TCanvas* cTrain = new TCanvas("cTrain","Network analysis");
cTrain->Divide(2,2);
TMLPAnalyzer ana(mlp);
// Initialisation
ana.GatherInformations();
//
// output to the console
ana.CheckNetwork();
cTrain->cd(1);
// shows how each variable influences the network
ana.DrawDInputs();
cTrain->cd(2);
// shows the network structure
mlp->Draw();
cTrain->cd(3);
// draws the resulting network
ana.DrawNetwork(0,"iType==1","iType==0");
```

Network with structure: @x[0],@x[1],@x[2],@x[3]:5:3:iType
inputs with low values in the differences plot may not be needed
@x[0] -> 0.0158814 +/- 0.00631877
@x[1] -> 0.00728075 +/- 0.00291779
@x[2] -> 0.000366171 +/- 0.00022766
@x[3] -> 0.00112137 +/- 0.000837734









siguang@pku.edu.cn

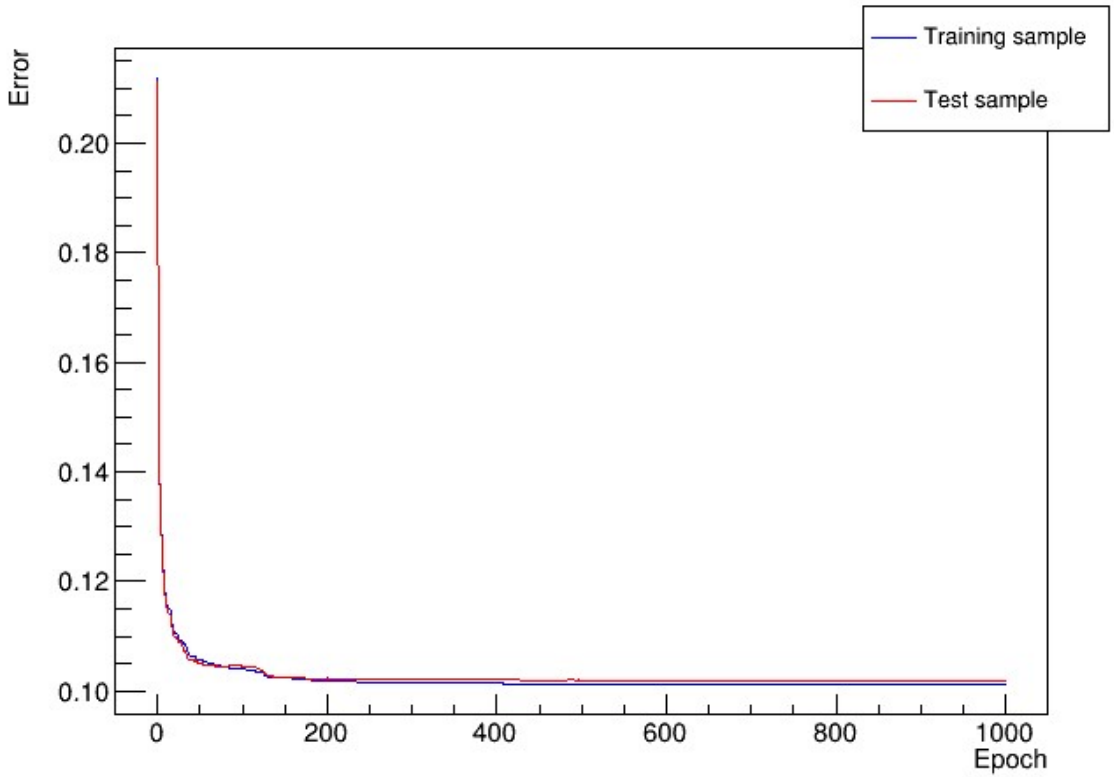
MLP神经网络训练代码（toTrain.C 续）

```
cTrain->cd(4);
// Use the NN to plot the results for each sample
// This will give approx. the same result as DrawNetwork.
// All entries are used, while DrawNetwork focuses on the test sample.
TH1F *bg = new TH1F("bgh", "NN output", 50, -0.5, 1.5);
TH1F *sig = new TH1F("sigh", "NN output", 50, -0.5, 1.5);
Double_t params[4];
for (Int_t i = 0; i < T->GetEntries(); i++) {
  T->GetEntry(i);
  for(Int_t j=0;j<N; j++) params[j] = x[j];
  if(iType==0) bg->Fill(mlp->Evaluate(0, params));
  else sig->Fill(mlp->Evaluate(0, params));
}
bg->SetLineColor(kBlue);
bg->SetFillStyle(3008);   bg->SetFillColor(kBlue);
sig->SetLineColor(kRed);
sig->SetFillStyle(3003); sig->SetFillColor(kRed);
bg->SetStats(0);
sig->SetStats(0);
```

MLP神经网络训练代码（toTrain.C 续）

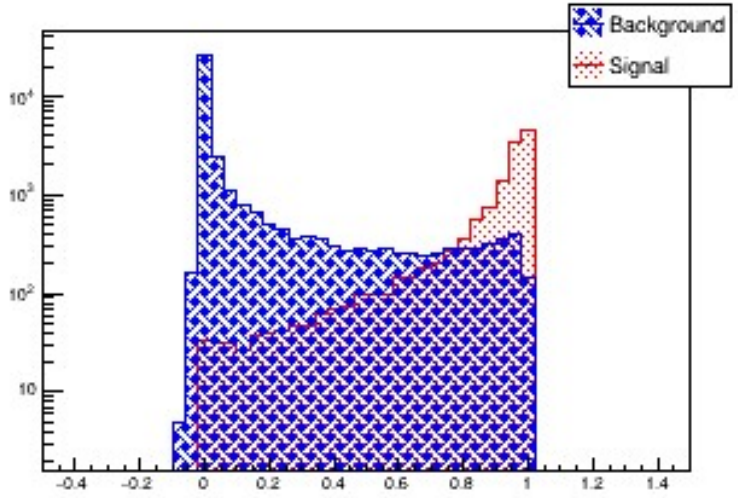
```
THStack *ht = new THStack();
ht->Add(sig);
ht->Add(bg);
ht->Draw("nostack");

TLegend *legend = new TLegend(.75, .80, .95, .95);
legend->AddEntry(bg, "Background");
legend->AddEntry(sig, "Signal");
legend->Draw();
cTrain->cd(0);
```

训练过程中错误率的显示



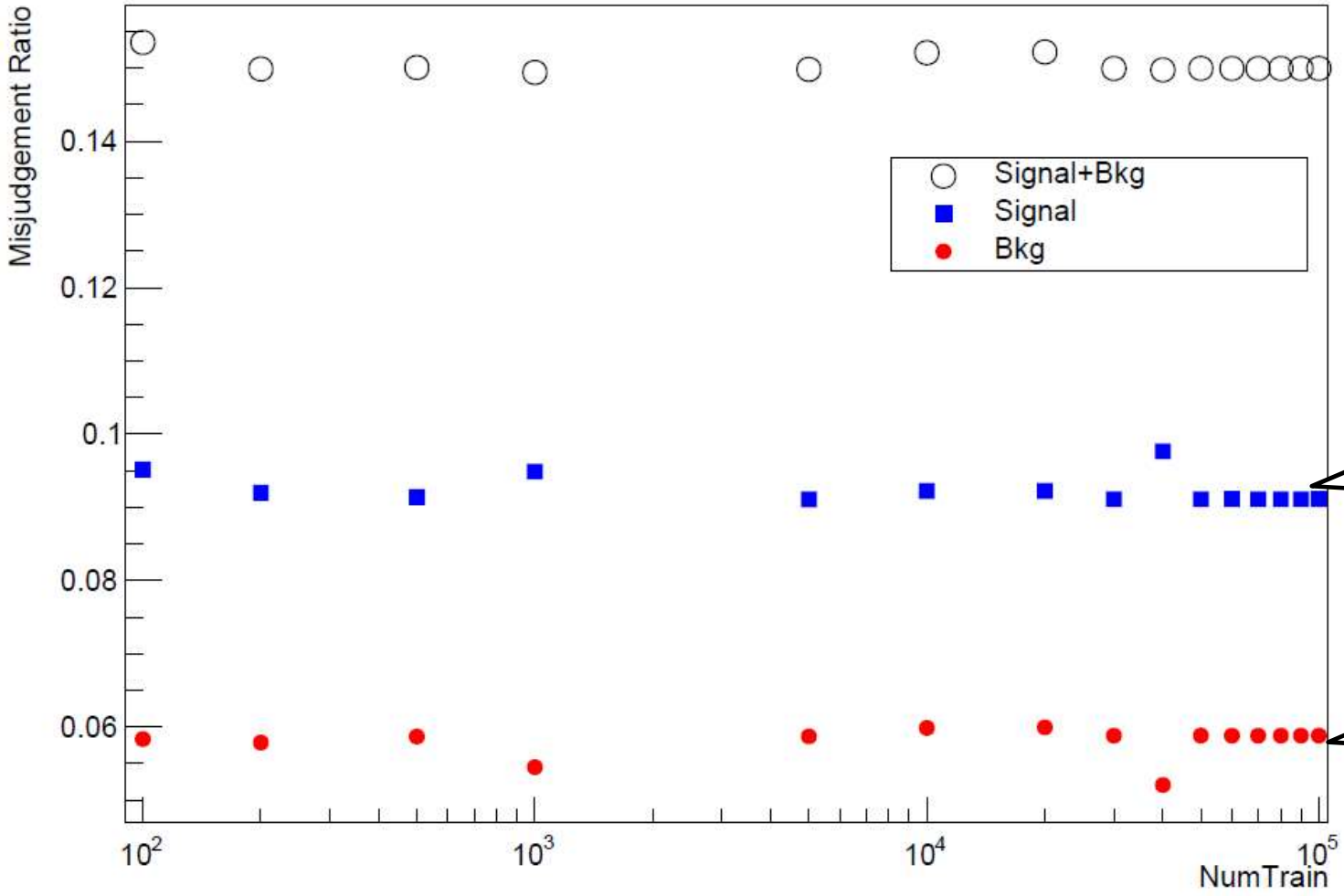
在训练的过程中，程序给出错误率的显示（如左图）。

如果随着训练代数的增加，错误率变大了，说明过度训练。建议改变层数及各层的神经元数目重新训练！



siguang@pku.edu.cn

对不同训练总代数的类进行检验



检验中**Value>0.5**的认为被鉴别为信号；否则就是本底

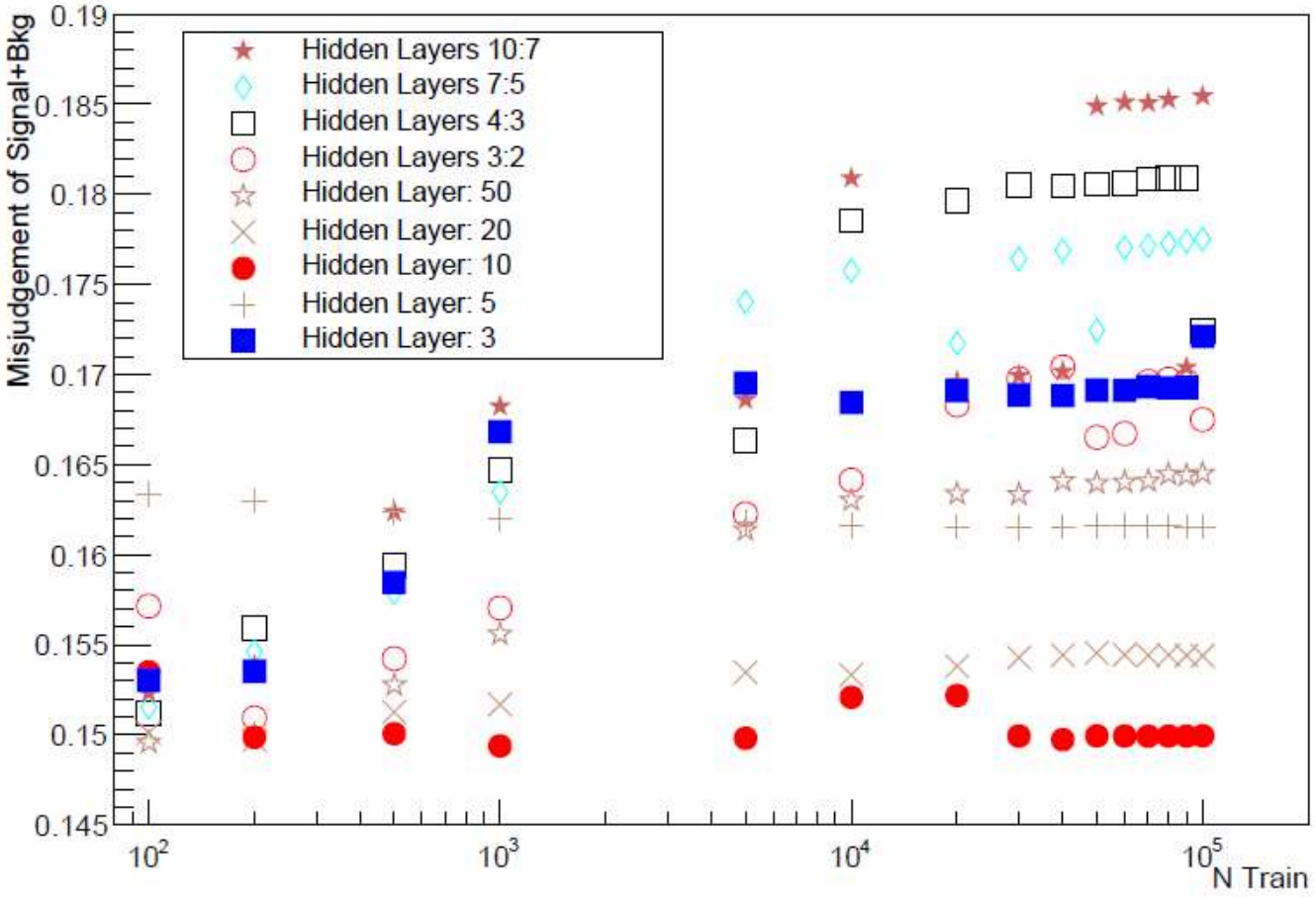
信号被误判为本底的比例

本底被误判为信号的比例

siguang@pku.edu.cn

MLP层数及各层神经元数目的确定

对于一个固定问题，需要用多少层及每层的神经元数目多少怎么确定？



对不同层数、不同神经元数目，由简单到复杂的神经网络结构进行不同训练代数的测试，找出

1）信号+本底错误率最小

2）随训练代数的增加比较稳定的结构！

siguang@pku.edu.cn

MLP神经网络训练生成的类（TClassANN.h）

```
#ifndef TClassANN_h
#define TClassANN_h

class TClassANN {
public:
   TClassANN() {}
   ~TClassANN() {}
   double Value(int index,double in0,double in1,double in2,double in3);
   double Value(int index, double* input);
private:
   double input0;
   double input1;
   double input2;
   double input3;
   double neuron0x2054d50();
   double neuron0x2055090();
   double neuron0x20553d0();
   double neuron0x2055710();
   double input0x2055b80();
   double neuron0x2055b80();
   double input0x2055ef0();
   double neuron0x2055ef0();

   ……
```

MLP神经网络训练生成的类（TClassANN.cxx）

```
#include "TClassANN.h"
#include <cmath>

double TClassANN::Value(int index,double in0,double in1,double in2,double in3) {
   input0 = (in0 - 2.59797)/0.526293;
   input1 = (in1 - 3.59619)/0.861869;
   input2 = (in2 - 4.58778)/1.22867;
   input3 = (in3 - 5.56243)/1.58495;
   switch(index) {
     case 0:
         return neuron0x2057f90();
     default:
         return 0.;
   }
}
```

```
double TClassANN::neuron0x2057f90() {
   double input = input0x2057f90();
   return (input * 1)+0;
}
```

```
double TClassANN::input0x2057f90() {
   double input = 0.0208308;
   input += synapse0x2058240();
   input += synapse0x2058280();
   input += synapse0x20582c0();
   return input;
}
```

MLP神经网络训练生成的类（TClassANN.cxx）

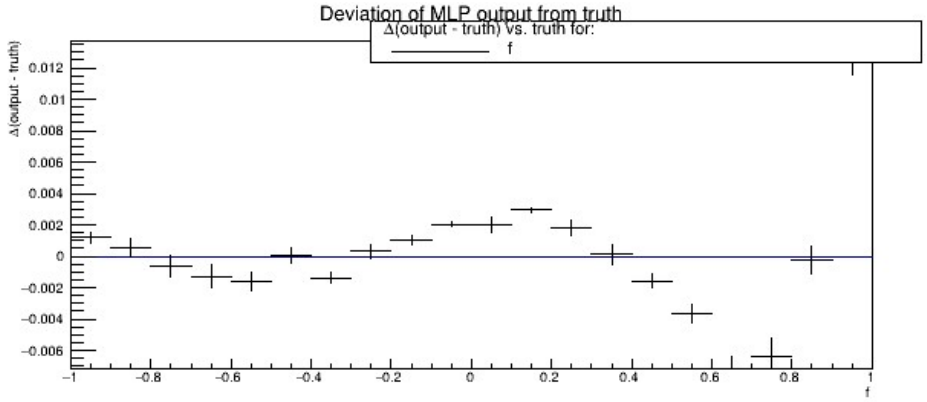
```
double TClassANN::Value(int index, double* input) {
    input0 = (input[0] - 2.59797)/0.526293;
    input1 = (input[1] - 3.59619)/0.861869;
    input2 = (input[2] - 4.58778)/1.22867;
    input3 = (input[3] - 5.56243)/1.58495;
    switch(index) {
      case 0:
          return neuron0x2057f90();
      default:
          return 0.;
    }
}
```

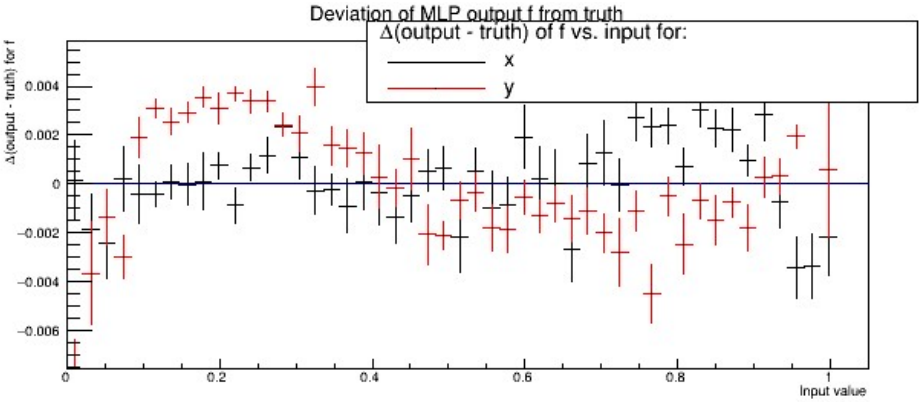
对训练出来的类，可以独立使用！

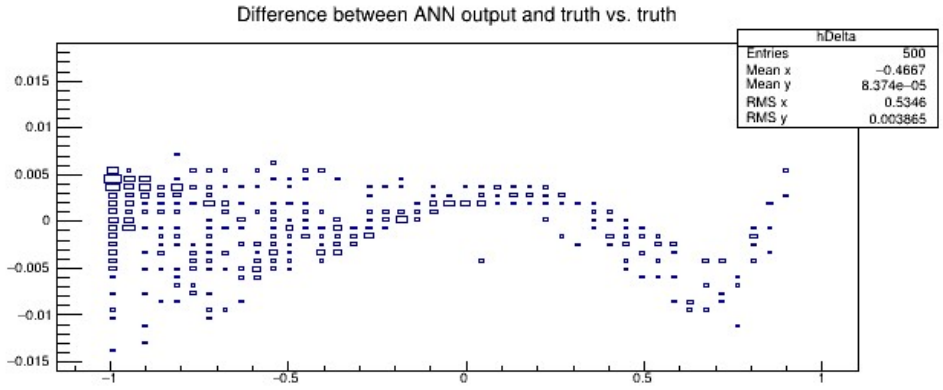
对连续函数进行描述

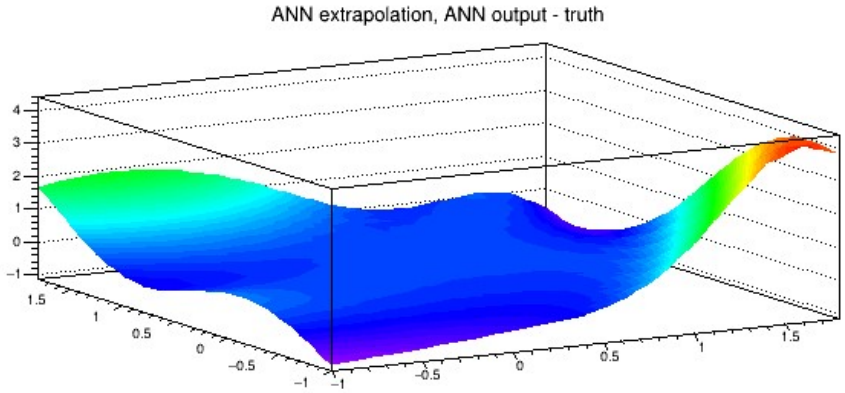
mlpRegression.C

- $ROOTSYS/tutorials/mlp$ emacs mlpRegression.C



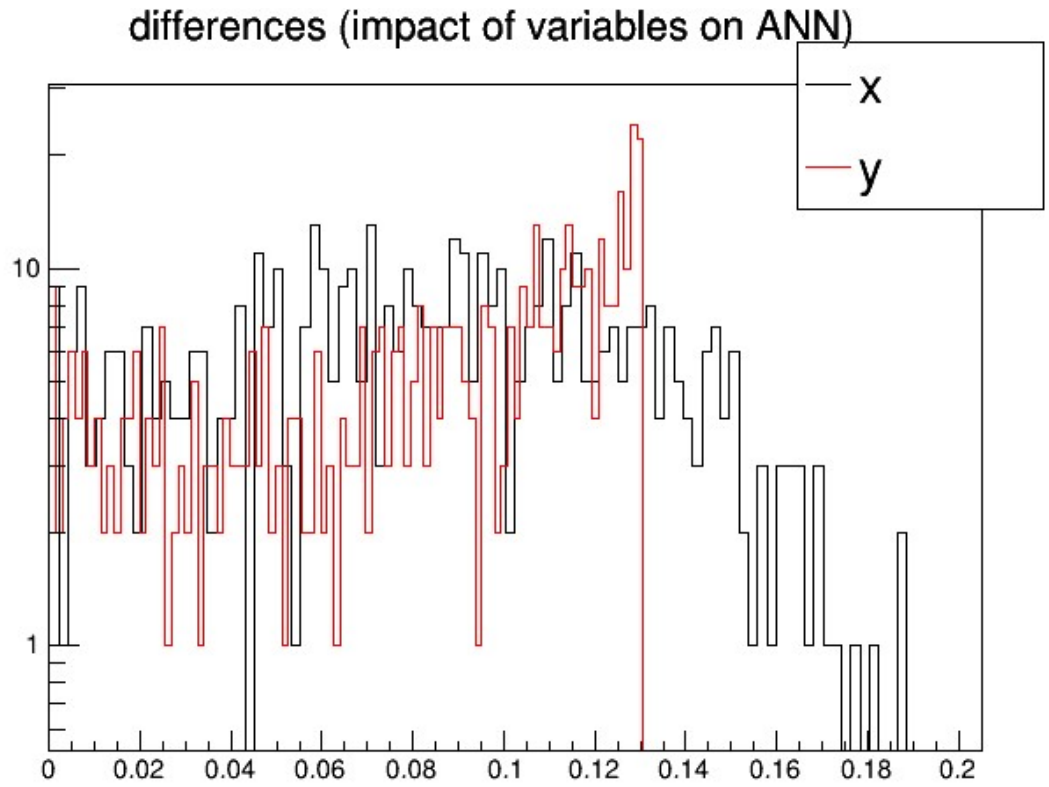






siguang@pku.edu.cn

mlpRegression.C （继续）



siguang@pku.edu.cn

mlpRegression.C （继续）

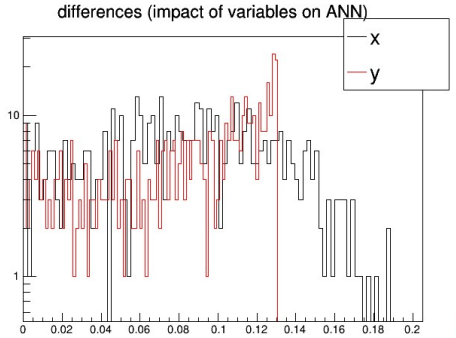
```
//This macro shows the use of an ANN for regression analysis:
//given a set {i} of input vectors i and a set {o} of output vectors o,
//one looks for the unknown function f(i)=o.
//The ANN can approximate this function; TMLPAnalyzer::DrawTruthDeviation
//methods can be used to evaluate the quality of the approximation.
//
//For simplicity, we use a known function to create test and training data.
//In reality this function is usually not known, and the data comes e.g.
//from measurements.
//
//Axel Naumann, 2005-02-02

Double_t theUnknownFunction(Double_t x, Double_t y) {
   return sin((1.7+x)*(x-0.3)-2.3*(y+0.7));
}
```

mlpRegression.C （继续）

```
void mlpRegression() {
   // create a tree with train and test data.
   // we have two input parameters x and y,
   // and one output value f(x,y)
   TNtuple* t=new TNtuple("tree","tree","x:y:f");
   TRandom r;
   for (Int_t i=0; i<1000; i++) {
      Float_t x=r.Rndm();
      Float_t y=r.Rndm();
      // fill it with x, y, and f(x,y) - usually this function
      // is not known, and the value of f given an x and a y comes
      // e.g. from measurements
      t->Fill(x,y,theUnknownFunction(x,y));
   }

   // create ANN
   TMultiLayerPerceptron* mlp=new TMultiLayerPerceptron("x,y:10:8:f",t,
      "Entry$%2","(Entry$%2)==0");
   mlp->Train(150,"graph update=10");
```

mlpRegression.C （继续）

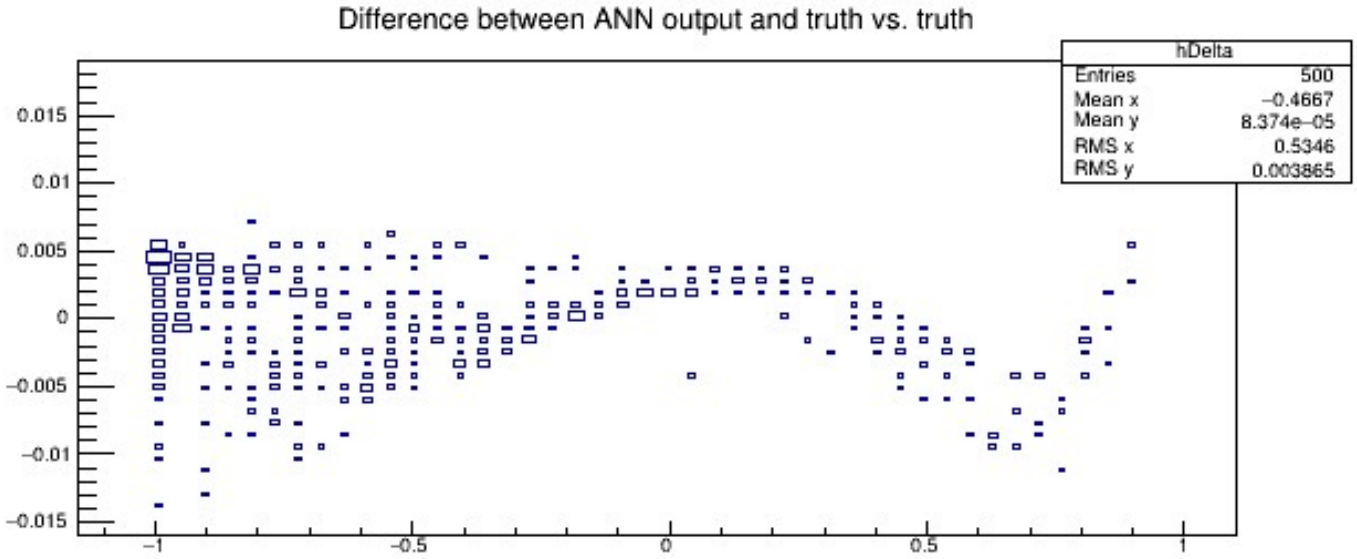
```
// analyze it
TMLPAnalyzer* mlpa=new TMLPAnalyzer(mlp);
mlpa->GatherInformations();
mlpa->CheckNetwork();
mlpa->DrawDInputs();
```




```
// draw statistics shows the quality of the ANN's approximation
TCanvas* cIO=new TCanvas("TruthDeviation", "TruthDeviation");
cIO->Divide(2,2);
cIO->cd(1);
// draw the difference between the ANN's output for (x,y) and
// the true value f(x,y), vs. f(x,y), as TProfiles
mlpa->DrawTruthDeviations();

cIO->cd(2);
// draw the difference between the ANN's output for (x,y) and
// the true value f(x,y), vs. x, and vs. y, as TProfiles
mlpa->DrawTruthDeviationInsOut();
```

mlpRegression.C （继续）

```
cIO->cd(3);
// draw a box plot of the ANN's output for (x,y) vs f(x,y)
mlpa->GetIOTree()->Draw("Out.Out0-True.True0:True.True0>>hDelta","","goff");
TH2F* hDelta=(TH2F*)gDirectory->Get("hDelta");
hDelta->SetTitle("Difference between ANN output and truth vs. truth");
hDelta->Draw("BOX");
```



mlpRegression.C （继续）

```
   cIO->cd(4);
   // draw difference of ANN's output for (x,y) vs f(x,y) assuming the ANN can extrapolate
   Double_t vx[225];
   Double_t vy[225];
   Double_t delta[225];
   Double_t v[2];
   for (Int_t ix=0; ix<15; ix++) {
      v[0]=ix/5.-1.;
      for (Int_t iy=0; iy<15; iy++) {
         v[1]=iy/5.-1.;
         Int_t idx=ix*15+iy;
         vx[idx]=v[0];
         vy[idx]=v[1];
         delta[idx]=mlp->Evaluate(0, v)-theUnknownFunction(v[0],v[1]);
      }
   }
   TGraph2D* g2Extrapolate=new TGraph2D("ANN extrapolation",
                                        "ANN extrapolation, ANN output - truth",
                                        225, vx, vy, delta);
   g2Extrapolate->Draw("TRI2");
}
```

多变量分析TMVA（Multi-Variate-Analysis类）

	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		⋆⋆	⋆⋆	⋆	⋆	⋆⋆	⋆⋆	○	○	○	○

ROOT有TMVA软件包：$ROOTSYS/tmva/test

在自学的时候建议先读该目录下的**README**

学习入口：
TMVAClassification.C 进行训练
TMVAClassificationApplication.C 展示怎么使用训练后的结果

因为TMVAClassification.C涉及到多种区分本底与信号的方法，为简便，本课对以上两程序进行简化：
展现BDT（Boosted Decision Trees 增强决策森林）和MLP（Multilayer Perceptron多层感知机）两种训练算法的使用

TMVA的训练

siguang@pku.edu.cn

TrainTMVA.C （简化后的TMVAClassification.C）

```
void TrainTMVA(Int_t totEvt=10000,Int_t NumTrain=200,Int_t nSeed=1){
  const Int_t Nx = 4;
  TF1 *fsig[Nx];
  TF1 *fbkg[Nx];
  const Double_t xmin=0;
  const Double_t xmax=10;
  const Double_t sqPi = sqrt(TMath::Pi());
  Int_t N = Nx;
  //prepare signal and background functions
  for(Int_t i=0; i<N; i++){
    fsig[i] = new TF1(Form("fsig%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
    fbkg[i] = new TF1(Form("fbkg%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
    Double_t MeanS = i+2.0;
    Double_t MeanB = i+2.8;
    Double_t sigmaS = (i+1.)*0.4;
    Double_t sigmaB = (i+1.)*0.6;
    fsig[i]->SetParameters(1/(sqPi*sigmaS),MeanS,sigmaS);
    fbkg[i]->SetParameters(1/(sqPi*sigmaB),MeanB,sigmaB);
    fsig[i]->SetNpx(10000);
    fbkg[i]->SetNpx(10000);
    fsig[i]->SetLineColor(i+1);
    fbkg[i]->SetLineColor(i+1);
```

TrainTMVA.C （续）

```
   fsig[i]->SetLineStyle(i+1);
   fbkg[i]->SetLineStyle(i+1);
   fsig[i]->SetLineWidth(2);
   fbkg[i]->SetLineWidth(4);
}

//
//create data

TTree *Tsig = new TTree("Tsig","Tree for Signal and Background");
Double_t var1,var2,var3,var4;
Double_t w;
Tsig->Branch("var1",&var1,"var1/D");
Tsig->Branch("var2",&var2,"var2/D");
Tsig->Branch("var3",&var3,"var3/D");
Tsig->Branch("var4",&var4,"var4/D");
Tsig->Branch("w",&w,"w/D");

TTree *Tbkg = Tsig->CloneTree(0);
```

不拷贝内容，
只克隆结构

TrainTMVA.C （续）

```
gRandom->SetSeed(nSeed);
Int_t nSig = totEvt/(1+3);
Int_t nBkg = totEvt - nSig;
for(Int_t i=0; i<nSig; i++){
  w = 3.0;  // signal weight 3
  var1 = fsig[0]->GetRandom();
  var2 = fsig[1]->GetRandom();
  var3 = fsig[2]->GetRandom();
  var4 = fsig[3]->GetRandom();
  Tsig->Fill();
}

for(Int_t i=0; i<nBkg; i++){
  w = 1.0;  // background weight 1
  var1 = fbkg[0]->GetRandom();
  var2 = fbkg[1]->GetRandom();
  var3 = fbkg[2]->GetRandom();
  var4 = fbkg[3]->GetRandom();
  Tbkg->Fill();
}
```

TrainTMVA.C （续）

```
//define TMVA factory-----
// Create a ROOT output file where TMVA will store ntuples, histograms, etc.
TString outfileName( "TMVAoutput.root" );
TFile* outputFile = TFile::Open( outfileName, "RECREATE" );

TMVA::Factory *factory = new TMVA::Factory( "TrainTMVA", outputFile,
                                            "!V:!Silent:Color:DrawProgressBar:
                                            Transformations=I;D;P;G,D:AnalysisType=Classification" );
//V---Verbose flag
//Silent---Batch mode: boolean silent flag inhibiting any output from TMVA
//after the creation of the factory class object (default: False)
//Color---Flag for coloured screen output (default: True, if in batch mode: False
//DrawProgressBar---Draw progress bar to display training, testing and evaluation schedule (default: True)
//Transformations---List of transformations to test; formatting example: "Transformations=I;D;P;G,D",
//for identity, decorrelation, PCA, and Gaussianisation followed by decorrelation transformations"

factory->AddVariable( "var1", "Variable 1", "units", 'F' );
factory->AddVariable( "var2", "Variable 2", "units", 'F' );
factory->AddVariable( "var3", "Variable 3", "units", 'F' );
factory->AddVariable( "var4", "Variable 4", "units", 'F' );
//AddVariable(const TString& expression, const TString& title, const TString& unit, char type = 'F',
// Double_t min = 0, Double_t max = 0)
```

PCA（Principal Component Analysis）是一种常用的数据分析方法。PCA通过线性变换将原始数据变换为一组各维度线性无关的表示，可用于提取数据的主要特征分量，常用于高维数据的降维。

TrainTMVA.C （续）

```
// Add signal or background trees
factory->AddSignalTree    ( Tsig );
factory->AddBackgroundTree( Tbkg);
factory->SetSignalWeightExpression("w");
factory->SetBackgroundWeightExpression("w");

// Apply additional cuts on the signal and background samples (can be different)
TCut mycuts = ""; // for example: TCut mycuts = "abs(var1)<0.5 && abs(var2-0.5)<1";
TCut mycutb = ""; // for example: TCut mycutb = "abs(var1)<0.5";
factory->PrepareTrainingAndTestTree( mycuts, mycutb,
                                     "nTrain_Signal=0:nTrain_Background=0:
                                     SplitMode=Random:NormMode=NumEvents:!V" );
```

TrainTMVA.C （续）

```
// Adaptive Boost
factory->BookMethod( TMVA::Types::kBDT, "BDT",
                     "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:BoostType=AdaBoost:
                     AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction=0.5:
                     SeparationType=GiniIndex:nCuts=20" );
//BDT options (http://tmva.sourceforge.net/optionRef.html):
//H---Print method-specific help, (default False)
//V---Verbose output, (False)
//NTrees---Number of trees in the forest, default 800
//MinNodeSize--Minimum percentage of training events required in a leaf node
//                                (default: Classification: 5%, Regression: 0.2%)
//MaxDepth---Max depth of the decision tree allowed (3)
//BoostType--- Boosting type for the trees in the forest (AdaBoost)
//AdaBoostBeta---Learning rate for AdaBoost algorithm(0.5)
//UseBaggedBoost---Use only a random subsample of all events for growing the trees in each iteration (False)
//BaggedSampleFraction---Relative size of bagged event sample to original size of the data sample
//                                (used whenever bagging is used (i.e. UseBaggedGrad, Bagging,) (0.6)
//SeparationType---Separation criterion for node splitting (GiniIndex)
//nCuts---Number of grid points in variable range used in finding optimal cut in node splitting (20)
```

TrainTMVA.C （续）

```
  // Tmlp(Root)ANN
  factory->BookMethod( TMVA::Types::kTMlpANN, "TMlpANN", "!H:!V:NCycles=200:HiddenLayers=N+1,N:LearningMethod=BFGS:ValidationFraction=0.3"   );
  //OPtions of TMlpANN (http://tmva.sourceforge.net/optionRef.html)
  //HiddenLayers---Specification of hidden layer architecture (N stands for number of variables; any integers may also be used)
  //ValidationFraction---Fraction of events in training tree used for cross validation(0.5)

  // ---- Now you can tell the factory to train, test, and evaluate the MVAs

  // Train MVAs using the set of training events
  factory->TrainAllMethods();

  // ---- Evaluate all MVAs using the set of test events
  factory->TestAllMethods();

  // ----- Evaluate and compare performance of all configured MVAs
  factory->EvaluateAllMethods();

  // --------------------------------------------------------------

  // Save the output
  outputFile->Close();

  delete factory;

}
```

TrainTMVA.C 训练结果

训练后会在当前目录下生成weights子目录，该目录下针对BDT和MLP两种方法，生成训练后可独立使用的c++类和xml文件。

MLP的C++类文件名：TrainTMVA_TMlpANN.cxx 及头文件TrainTMVA_TMlpANN.h
MLP的xml文件名： TrainTMVA_TMlpANN.weights.xml

```
#ifndef TrainTMVA_TMlpANN_h
#define TrainTMVA_TMlpANN_h

class TrainTMVA_TMlpANN {
public:
   TrainTMVA_TMlpANN() {}
   ~TrainTMVA_TMlpANN() {}
   double Value(int index,double in0,double in1,double in2,double in3);
   double Value(int index, double* input);
private:
   double input0;
   double input1;
   double input2;
   double input3;
```

MLP通过TMVA训练处的类与root自带的TMultiLayerPerceptron训练出类的类一样。

TrainTMVA.C 训练结果

BDT的C++类文件名：TrainTMVA_BDT.class.C
BDT的xml文件名：　TrainTMVA_BDT.weights.xml

TrainTMVA_BDT.class.C

```
class ReadBDT : public IClassifierReader {

 public:

   // constructor
   ReadBDT( std::vector<std::string>& theInputVars )
   ......
   double GetMvaValue__( const std::vector<double>& inputValues ) const;
   ......
```

>0 信号
否则本底

TMVA的训练结果的应用：用xml文件测试训练结果

训练后的Xml的内容 --TrainTMVA_TMlpANN.weights.xml

```
<?xml version="1.0"?>
<MethodSetup Method="TMlpANN::TMlpANN">
  <GeneralInfo>
    <Info name="TMVA Release" value="4.2.0 [262656]"/>
    <Info name="ROOT Release" value="5.34/36 [336420]"/>
    <Info name="Creator" value="wsg"/>
    <Info name="Date" value="Mon Jul 31 03:42:06 2017"/>
    <Info name="Host" value="Linux debian 3.2.0-4-amd64 #1 SMP Debian 3.2.78-1 x86_64 GNU/Linux"/>
    <Info name="Dir" value="/media/sf_UbuntuShare/rootEduBOOK/chapter3/tmva"/>
    <Info name="Training events" value="5000"/>
    <Info name="TrainingTime" value="6.33900785e+00"/>
    <Info name="AnalysisType" value="Classification"/>
  </GeneralInfo>
  <Options>
    <Option name="V" modified="Yes">False</Option>
    <Option name="VerbosityLevel" modified="No">Default</Option>
    <Option name="VarTransform" modified="No">None</Option>
    <Option name="H" modified="Yes">False</Option>
    <Option name="CreateMVAPdfs" modified="No">False</Option>
    <Option name="IgnoreNegWeightsInTraining" modified="No">False</Option>
    <Option name="NCycles" modified="Yes">200</Option>
    <Option name="HiddenLayers" modified="Yes">N+1,N</Option>
    <Option name="ValidationFraction" modified="Yes">3.000000e-01</Option>
    <Option name="LearningMethod" modified="Yes">bfgs</Option>
  </Options>
  <Variables NVar="4">
    <Variable VarIndex="0" Expression="var1" Label="var1" Title="Variable 1" Unit="units" Internal="var1" Type="F" Min="9.57743347e-01" Max="4.37795401e+00"/>
    <Variable VarIndex="1" Expression="var2" Label="var2" Title="Variable 2" Unit="units" Internal="var2" Type="F" Min="5.56589127e-01" Max="6.46939230e+00"/>
    <Variable VarIndex="2" Expression="var3" Label="var3" Title="Variable 3" Unit="units" Internal="var3" Type="F" Min="1.47316948e-01" Max="8.63263226e+00"/>
    <Variable VarIndex="3" Expression="var4" Label="var4" Title="Variable 4" Unit="units" Internal="var4" Type="F" Min="2.01887488e-01" Max="9.99827957e+00"/>
  </Variables>
```

训练后的Xml的内容--TrainTMVA_TMlpANN.weights.xml （续）

```
  <Spectators NSpec="0"/>
  <Classes NClass="2">
    <Class Name="Signal" Index="0"/>
    <Class Name="Background" Index="1"/>
  </Classes>
  <Transformations NTransformations="0"/>
  <MVAPdfs/>
  <Weights>
    <Architecture BuildOptions="@var1,@var2,@var3,@var4:5:4:type"/>
    <input>
      0.524228 2.59941 0.855831 3.60237 1.21668 4.5976 1.58965 5.57355
    </input>
    <output>
      1 0
    </output>
    <neurons>
      -0.485791 0.240646 0.0774717 -0.0232713 0.861459 2.53571 3.17827 0.568858 -0.0317327 1.55842 -0.737349 0.275889 -6.63192 -0.0816457
    </neurons>
    <synapses>
      -1.02722 -2.63409 -1.46141 -2.63127 -1.27809 -0.494163 -0.120283 -0.212213 -3.23353 0.459102 1.72003 -1.3362 -3.00814 -2.72479 0.697151 1.48738 -3.16787 0.166163 -1.80745 -0.220989 -2.37366 -1.3176 0.48645 -0.421324 1.64198 -0.332904 0.457442 -0.55447 0.0389644 -1.29349 -0.55402 0.60312 0.840831 -0.359233 -0.780982 1.9062 -3.16468 3.03454 1.57723 2.35888 0.0588785 0.397996 -0.113308 3.65965
    </synapses>
  </Weights>
</MethodSetup>
```

训练后的xml---TrainTMVA_BDT.weights.xml

```
<?xml version="1.0"?>
<MethodSetup Method="BDT::BDT">
  <GeneralInfo>
    <Info name="TMVA Release" value="4.2.0 [262656]"/>
    <Info name="ROOT Release" value="5.34/36 [336420]"/>
    <Info name="Creator" value="wsg"/>
    <Info name="Date" value="Mon Jul 31 03:41:57 2017"/>
    <Info name="Host" value="Linux debian 3.2.0-4-amd64 #1 SMP Debian 3.2.78-1 x86_64 GNU/Linux"/>
    <Info name="Dir" value="/media/sf_UbuntuShare/rootEduBOOK/chapter3/tmva"/>
    <Info name="Training events" value="5000"/>
    <Info name="TrainingTime" value="1.81201911e+00"/>
    <Info name="AnalysisType" value="Classification"/>
  </GeneralInfo>
  <Options>
    <Option name="V" modified="Yes">False</Option>
    <Option name="VerbosityLevel" modified="No">Default</Option>
    <Option name="VarTransform" modified="No">None</Option>
    <Option name="H" modified="Yes">False</Option>
    <Option name="CreateMVAPdfs" modified="No">False</Option>
    <Option name="IgnoreNegWeightsInTraining" modified="No">False</Option>
    <Option name="NTrees" modified="Yes">850</Option>
```

。 。 。 。 。 。

训练后的xml---TrainTMVA_BDT.weights.xml（续）

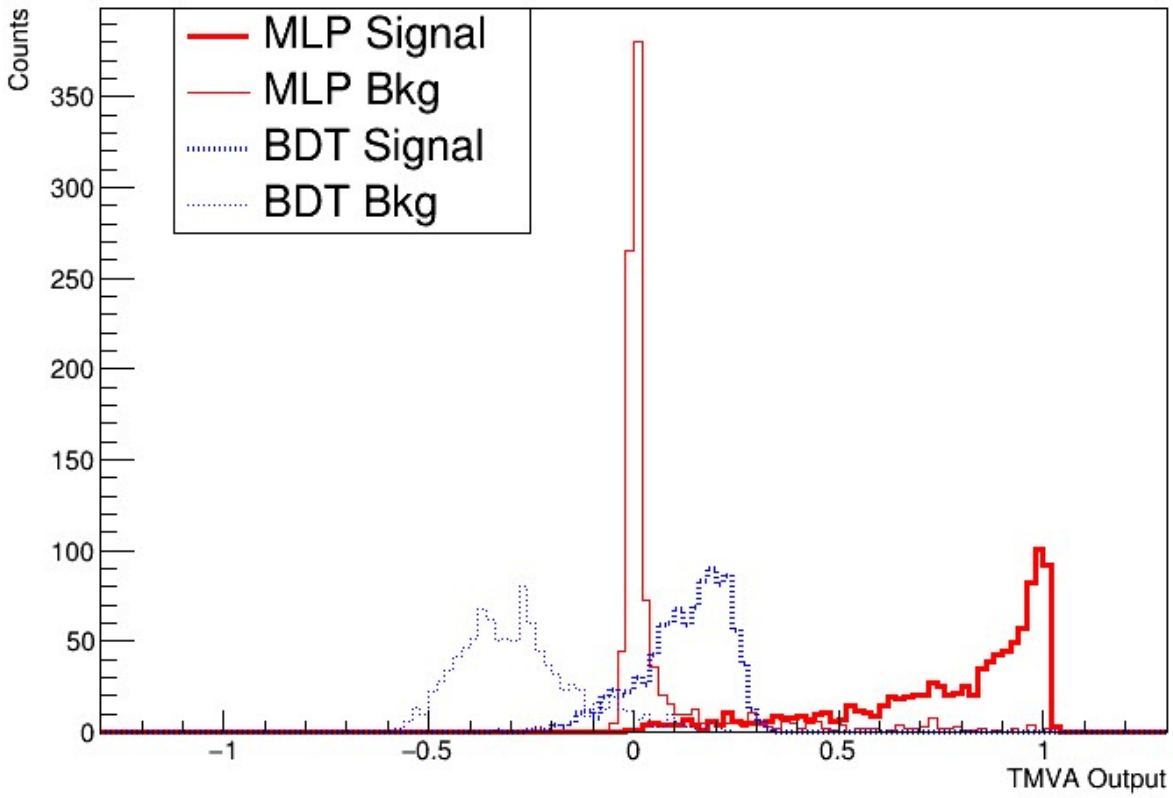
```
  <Weights NTrees="850" AnalysisType="0">
    <BinaryTree type="DecisionTree" boostWeight="9.9781444396469765e-01" itree="0">
      <Node pos="s" depth="0" NCoef="0" IVar="0" Cut="2.3041086196899414e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="5.0131785869598389e-01" nType="0"
s>
        <Node pos="l" depth="1" NCoef="0" IVar="1" Cut="3.9850797653198242e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="8.7936508655548096e-01" nType="
s0">
          <Node pos="l" depth="2" NCoef="0" IVar="-1" Cut="5.4512405395507812e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="9.1946703195571899e-01" nTyp
se="1"/>
          <Node pos="r" depth="2" NCoef="0" IVar="-1" Cut="0.0000000000000000e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="4.5731708407402039e-01" nTyp
se="-1"/>
        </Node>
        <Node pos="r" depth="1" NCoef="0" IVar="-1" Cut="2.6019032001495361e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="1.2605041265487671e-01" nType=
s"-1"/>
      </Node>
    </BinaryTree>
```

第850个（从序号0开始）

```
    <BinaryTree type="DecisionTree" boostWeight="1.4243556483244563e-01" itree="849">
      <Node pos="s" depth="0" NCoef="0" IVar="2" Cut="2.5716927051544189e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="4.4853436946868896e-01" nType="0"
s>
        <Node pos="l" depth="1" NCoef="0" IVar="-1" Cut="0.0000000000000000e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="8.2950592041015625e-01" nType=
s"1"/>
        <Node pos="r" depth="1" NCoef="0" IVar="3" Cut="4.8414001464843750e+00" cType="0" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="4.3779209256172180e-01" nType="
s0">
          <Node pos="l" depth="2" NCoef="0" IVar="3" Cut="6.5428924560546875e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="4.7115877270698547e-01" nType
s="0">
            <Node pos="l" depth="3" NCoef="0" IVar="-1" Cut="0.0000000000000000e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="5.0120848417282104e-01" nT
ype="1"/>
            <Node pos="r" depth="3" NCoef="0" IVar="-1" Cut="0.0000000000000000e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="3.3787220716476440e-01" nT
ype="-1"/>
          </Node>
          <Node pos="r" depth="2" NCoef="0" IVar="-1" Cut="3.7366983890533447e+00" cType="1" res="-9.9000000000000000e+01" rms="0.0000000000000000e+00" purity="3.8180994987487793e-01" nTyp
se="-1"/>
        </Node>
      </Node>
    </BinaryTree>
  </Weights>
</MethodSetup>
```

xml文件的应用例子： UseXMLOutput.C

UseXMLOutput.C的输出



siguang@pku.edu.cn

UseXMLOutput.C

```
#include <cstdlib>
#include <vector>
#include <iostream>
#include <map>
#include <string>

using namespace TMVA;
void UseOutput(){
// part of codes are from /tmva/test$ emacs TMVAClassificationApplication.C
#ifdef __CINT__
   gROOT->ProcessLine( ".O0" ); // turn off optimization in CINT
#endif
  // This loads the library
   TMVA::Tools::Instance();

   // --- Create the Reader object

   TMVA::Reader *reader = new TMVA::Reader( "!Color:!Silent" );
```

UseXMLOutput.C （续）

```
// Create a set of variables and declare them to the reader
// - the variable names MUST corresponds in name and type to
//   those given in the weight file(s) used
Float_t var1, var2;
Float_t var3, var4;
reader->AddVariable( "var1", &var1 );
reader->AddVariable( "var2", &var2 );
reader->AddVariable( "var3", &var3 );
reader->AddVariable( "var4", &var4 );

//book the method and weightfile
TString methodName = "TMlpANN method"; //MUST exactly
TString weightfile = "weights/TrainTMVA_TMlpANN.weights.xml";
reader->BookMVA( methodName, weightfile );

methodName = "BDT method";
weightfile = "weights/TrainTMVA_BDT.weights.xml";
reader->BookMVA( methodName, weightfile );
```

UseXMLOutput.C （续）

```
//prepare signal and background functions
const Int_t Nx = 4;
TF1 *fsig[Nx];
TF1 *fbkg[Nx];
const Double_t xmin=0;
const Double_t xmax=10;
const Double_t sqPi = sqrt(TMath::Pi());
Int_t N = Nx;

for(Int_t i=0; i<N; i++){
  fsig[i] = new TF1(Form("fsig%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
  fbkg[i] = new TF1(Form("fbkg%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
  Double_t MeanS = i+2.0;
  Double_t MeanB = i+2.8;
  Double_t sigmaS = (i+1.)*0.4;
  Double_t sigmaB = (i+1.)*0.6;
  fsig[i]->SetParameters(1/(sqPi*sigmaS),MeanS,sigmaS);
  fbkg[i]->SetParameters(1/(sqPi*sigmaB),MeanB,sigmaB);
  fsig[i]->SetNpx(10000);
  fbkg[i]->SetNpx(10000);
}
```

UseXMLOutput.C （续）

```
//create data
// Book output histograms
UInt_t nbin = 130;
TH1F* hSigMlp     = new TH1F( "hSigMlp",      "MVA_TMlpANN",       nbin, -1.3, 1.3 );
TH1F* hBkgMlp     = new TH1F( "hBkgMlp",      "MVA_TMlpANN",       nbin, -1.3, 1.3 );

TH1F* hSigBdt     = new TH1F( "hSigBdt",      "MVA_BDT",       nbin, -1.3, 1.3 );
TH1F* hBkgBdt     = new TH1F( "hBkgBdt",      "MVA_TBD",       nbin, -1.3, 1.3 );

Int_t nSig = 1000;
Int_t nBkg = 1000;
for(Int_t i=0; i<nSig; i++){
  var1 = fsig[0]->GetRandom();
  var2 = fsig[1]->GetRandom();
  var3 = fsig[2]->GetRandom();
  var4 = fsig[3]->GetRandom();
  hSigMlp->Fill(reader->EvaluateMVA( "TMlpANN method"));
  hSigBdt->Fill(reader->EvaluateMVA( "BDT method"));
}
```

UseXMLOutput.C （续）

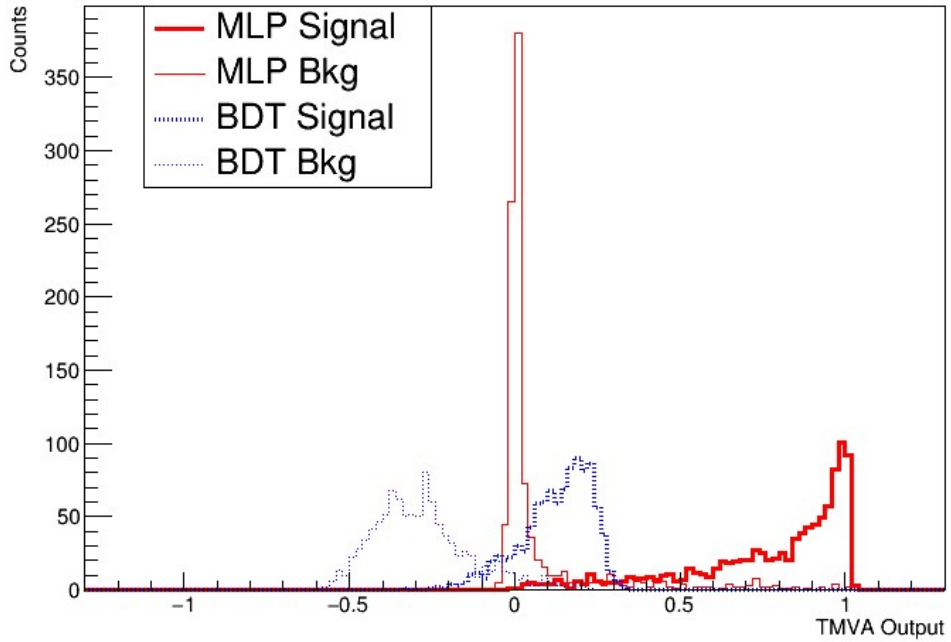
```
for(Int_t i=0; i<nBkg; i++){
   var1 = fbkg[0]->GetRandom();
   var2 = fbkg[1]->GetRandom();
   var3 = fbkg[2]->GetRandom();
   var4 = fbkg[3]->GetRandom();
   hBkgMlp->Fill(reader->EvaluateMVA( "TMlpANN method"));
   hBkgBdt->Fill(reader->EvaluateMVA( "BDT method"));
}

TCanvas *c1  = new TCanvas("c1","");
gPad->SetTopMargin(0.03);
gPad->SetRightMargin(0.03);
THStack *hs = new THStack();
hs->Add(hSigMlp);
hs->Add(hBkgMlp);
hs->Add(hSigBdt);
hs->Add(hBkgBdt);
```

UseXMLOutput.C （续）

```
hSigMlp->SetLineColor(kRed);
hSigMlp->SetLineWidth(3);
hSigMlp->SetLineStyle(1);
hBkgMlp->SetLineColor(kRed);
hBkgMlp->SetLineWidth(1);
hBkgMlp->SetLineStyle(1);

hSigBdt->SetLineColor(kBlue);
hSigBdt->SetLineWidth(3);
hSigBdt->SetLineStyle(3);
hBkgBdt->SetLineColor(kBlue);
hBkgBdt->SetLineWidth(1);
hBkgBdt->SetLineStyle(3);

hs->Draw("nostack");
hs->SetTitle(";TMVA Output; Counts");
hs->GetYaxis()->SetTitleOffset(1.2);
```

UseXMLOutput.C （续）

```
    hs->GetYaxis()->SetTitleOffset(1.2);
    TLegend *lg = new TLegend(0.16,0.7,0.45,0.97);
    lg->AddEntry(hSigMlp,"MLP Signal","L");
    lg->AddEntry(hBkgMlp,"MLP Bkg","L");
    lg->AddEntry(hSigBdt,"BDT Signal","L");
    lg->AddEntry(hBkgBdt,"BDT Bkg","L");
    lg->Draw();
}
```

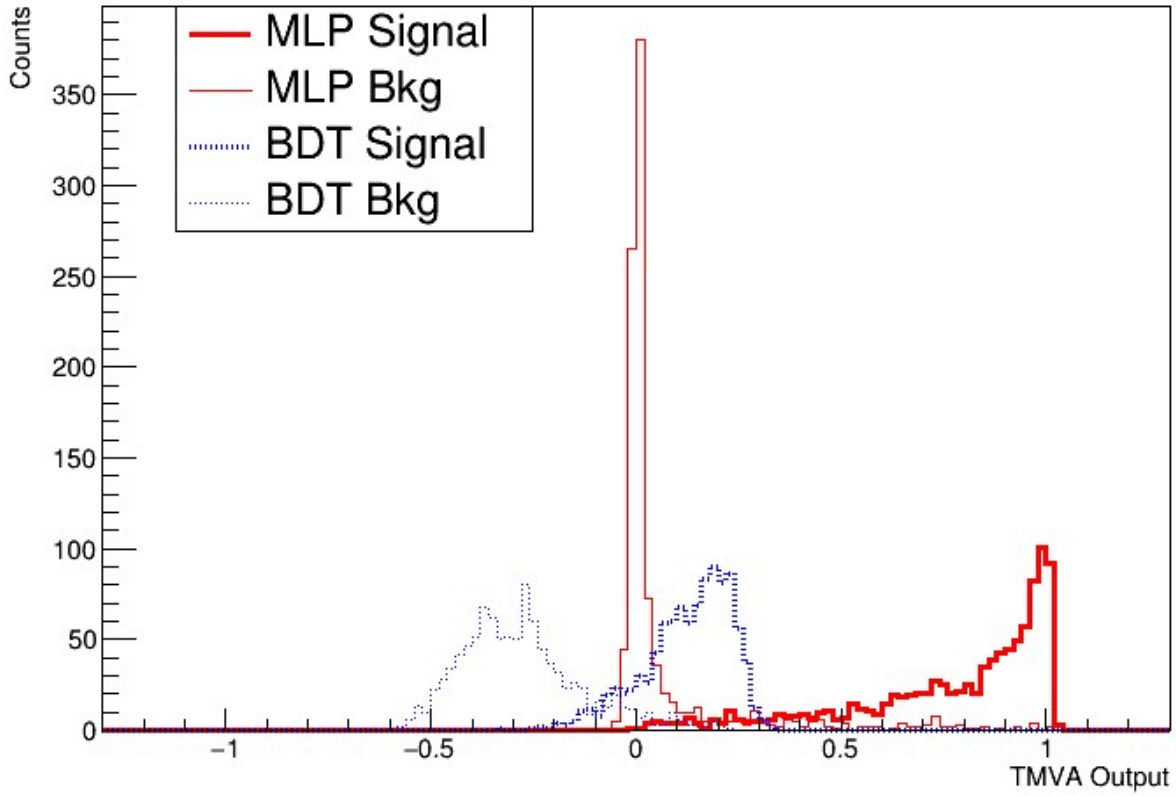


siguang@pku.edu.cn

TMVA的训练结果的应用：用C++文件测试训练结果

UseCppOutput.C输出

利用TMVA输出的C++文件进行测试



展示出怎么使用TMVA训练的类

siguang@pku.edu.cn

编译TMVA的C++输出成库文件

在运行TMVA训练程序的目录下（含有**weights**子目录的文件夹！），启动root，运行：

[] .L weights/TrainTMVA_TMlpANN.cxx++

生成TrainTMVA_TMlpANN_cxx.so （weights内）

注意：因为TrainTMVA_TMlpANN.cxx文件中含有#include “weights/TrainTMVA_TMlpANN.h”，如在weights子目录内运行，找不到.h头文件。报错如下：

/media/sf_code/chapter3/tmva/weights/./TrainTMVA_TMlpANN.cxx:1:39: fatal error: weights/TrainTMVA_TMlpANN.h: No such file or directory

#include "weights/TrainTMVA_TMlpANN.h"

故必须在weights上级目录运行！

[].L weights/TrainTMVA_BDT.class.C++

生成：TrainTMVA_BDT.class_C.so

UseCppOutput.C 代码（续）

```
#include "weights/TrainTMVA_BDT.class_C.so"
#include "weights/TrainTMVA_TMlpANN_cxx.so"
#include "weights/TrainTMVA_TMlpANN.h"


void UseCppOutput(){

   //prepare ReadBDT
   std::vector<std::string> theInputVars;
   theInputVars.push_back("var1");
   theInputVars.push_back("var2");
   theInputVars.push_back("var3");
   theInputVars.push_back("var4");
   ReadBDT *rBDT = new ReadBDT(theInputVars);

   //prepare Read Mlp
   TrainTMVA_TMlpANN *rMlp = new  TrainTMVA_TMlpANN();
```

解释运行的条件下
可直接include .so可以！

UseCppOutput.C 代码（续）

测试数据函数准备与xml代码同，可以忽略

```
//prepare signal and background functions
const Int_t Nx = 4;
TF1 *fsig[Nx];
TF1 *fbkg[Nx];
const Double_t xmin=0;
const Double_t xmax=10;
const Double_t sqPi = sqrt(TMath::Pi());
Int_t N = Nx;

for(Int_t i=0; i<N; i++){
  fsig[i] = new TF1(Form("fsig%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
  fbkg[i] = new TF1(Form("fbkg%d",i),"[0]*exp(-(x-[1])*(x-[1])/[2]/[2])",xmin,xmax);
  Double_t MeanS = i+2.0;
  Double_t MeanB = i+2.8;
  Double_t sigmaS = (i+1.)*0.4;
  Double_t sigmaB = (i+1.)*0.6;
  fsig[i]->SetParameters(1/(sqPi*sigmaS),MeanS,sigmaS);
  fbkg[i]->SetParameters(1/(sqPi*sigmaB),MeanB,sigmaB);
  fsig[i]->SetNpx(10000);
  fbkg[i]->SetNpx(10000);
}
```

UseCppOutput.C 代码（续）

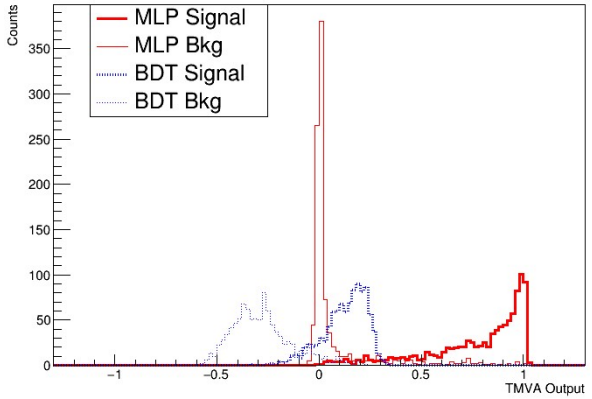
```
// output histograms
UInt_t nbin = 130;
TH1F* hSigMlp    = new TH1F( "hSigMlp",      "MVA_TMlpANN",      nbin, -1.3, 1.3 );
TH1F* hBkgMlp    = new TH1F( "hBkgMlp",      "MVA_TMlpANN",      nbin, -1.3, 1.3 );

TH1F* hSigBdt    = new TH1F( "hSigBdt",      "MVA_BDT",      nbin, -1.3, 1.3 );
TH1F* hBkgBdt    = new TH1F( "hBkgBdt",      "MVA_TBD",      nbin, -1.3, 1.3 );

Int_t nSig = 1000;
Int_t nBkg = 1000;
std::vector<double> vVar;
Double_t var1,var2,var3,var4;
for(Int_t i=0; i<nSig; i++){
  var1 = fsig[0]->GetRandom();
  var2 = fsig[1]->GetRandom();
  var3 = fsig[2]->GetRandom();
  var4 = fsig[3]->GetRandom();
  vVar.clear();
  vVar.push_back(var1);
  vVar.push_back(var2);
  vVar.push_back(var3);
  vVar.push_back(var4);
  hSigMlp->Fill(rMlp->Value(0,var1,var2,var3,var4));
  hSigBdt->Fill(rBDT->GetMvaValue(vVar));
}
```

UseCppOutput.C 代码（续）

```
TCanvas *c1  = new TCanvas("c1","");
gPad->SetTopMargin(0.03);
gPad->SetRightMargin(0.03);
THStack *hs = new THStack();
hs->Add(hSigMlp);
hs->Add(hBkgMlp);
hs->Add(hSigBdt);
hs->Add(hBkgBdt);
hSigMlp->SetLineColor(kRed);
hSigMlp->SetLineWidth(3);
hSigMlp->SetLineStyle(1);
hBkgMlp->SetLineColor(kRed);
hBkgMlp->SetLineWidth(1);
hBkgMlp->SetLineStyle(1);

hSigBdt->SetLineColor(kBlue);
hSigBdt->SetLineWidth(3);
hSigBdt->SetLineStyle(3);
hBkgBdt->SetLineColor(kBlue);
hBkgBdt->SetLineWidth(1);
hBkgBdt->SetLineStyle(3);
```

画图代码与xml文件的应用同，可以忽略

```
  hs->Draw("nostack");
  hs->SetTitle(";TMVA Output; Counts");
  hs->GetYaxis()->SetTitleOffset(1.2);
  TLegend *lg = new TLegend(0.16,0.7,0.45,0.97);
  lg->AddEntry(hSigMlp,"MLP Signal","L");
  lg->AddEntry(hBkgMlp,"MLP Bkg","L");
  lg->AddEntry(hSigBdt,"BDT Signal","L");
  lg->AddEntry(hBkgBdt,"BDT Bkg","L");
  lg->Draw();
}
```



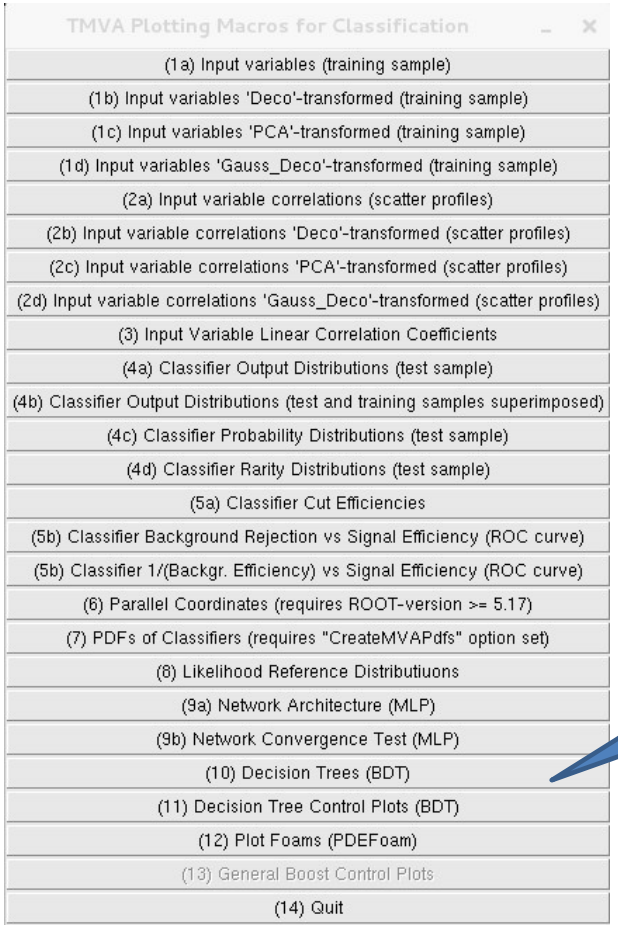
siguang@pku.edu.cn

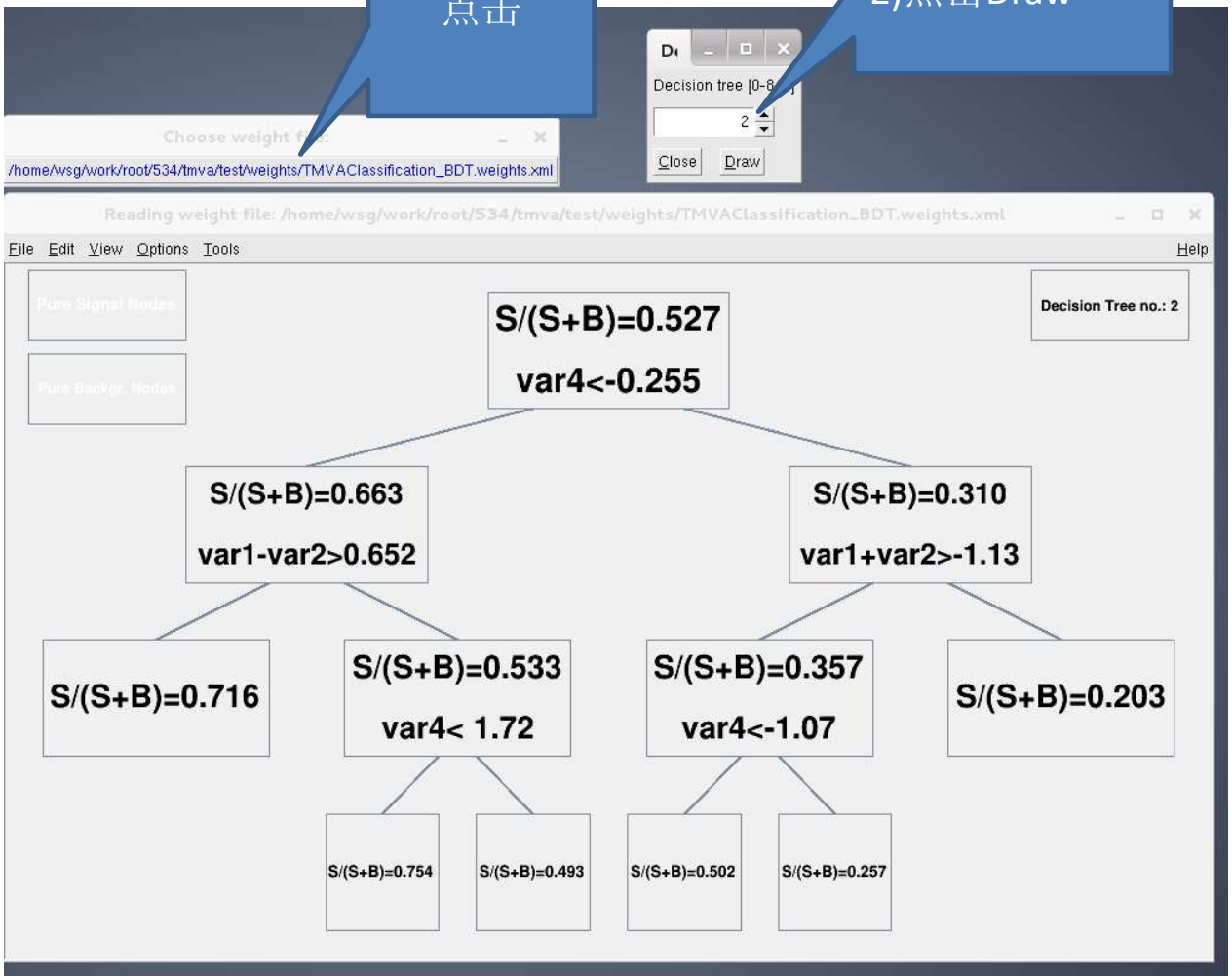
TMVA软件包结果分析程序

运行：

>root TMVAClassification.C

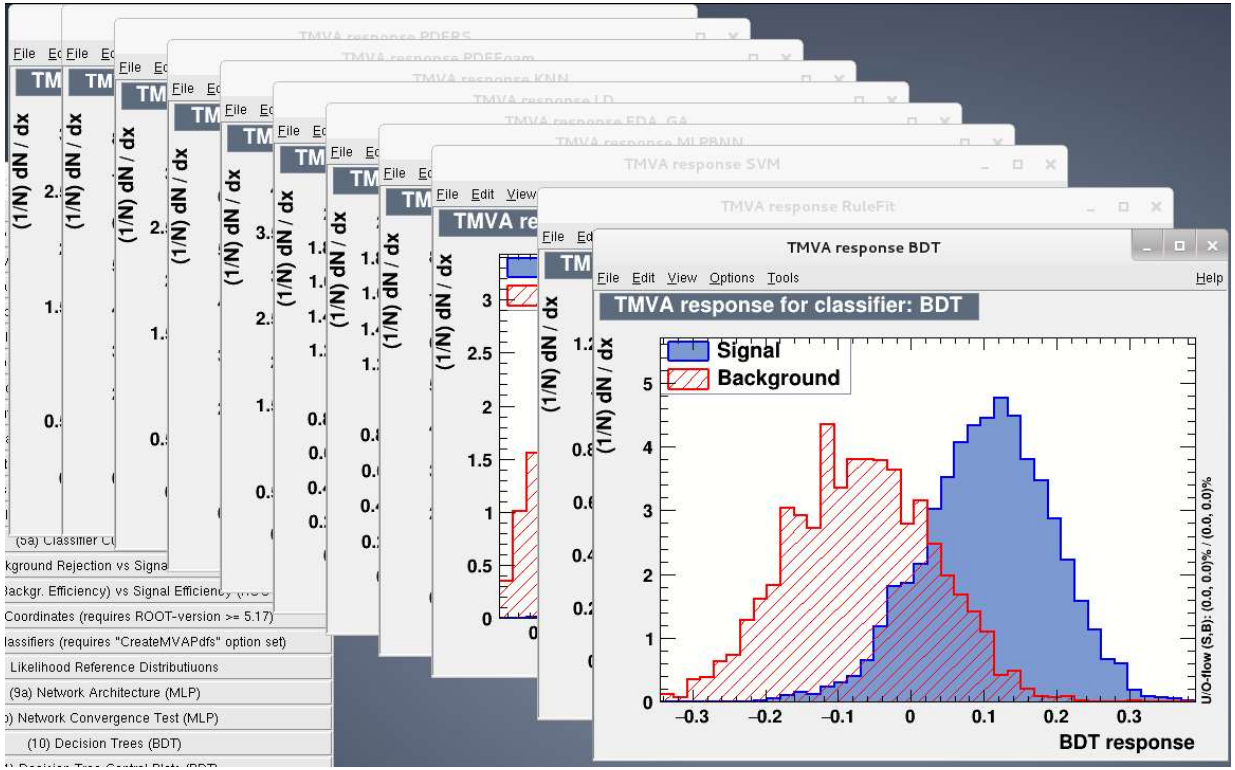
会弹出右边的窗口（TMVAGui.C文件运行结果），点击文字将运行不同的代码





siguang@pku.edu.cn

点击（4a）画出各种算法对信号与本底的分布图



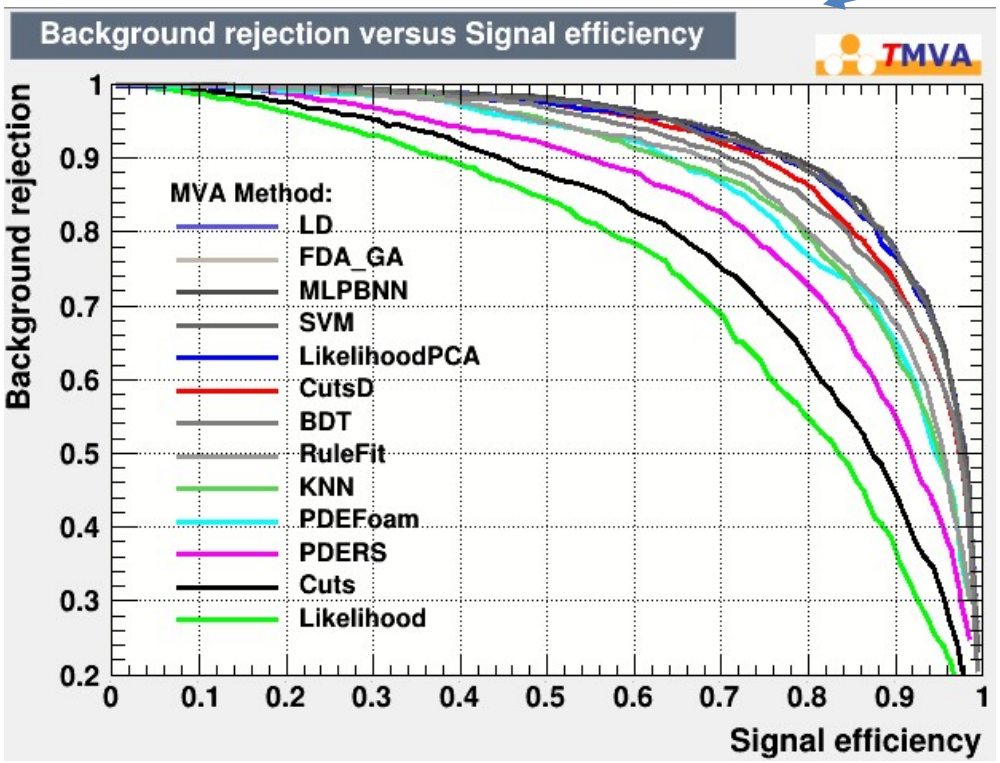
可以打开TMVAGui.C，找出来画图的程序文件名进行参考(mvas.C)：

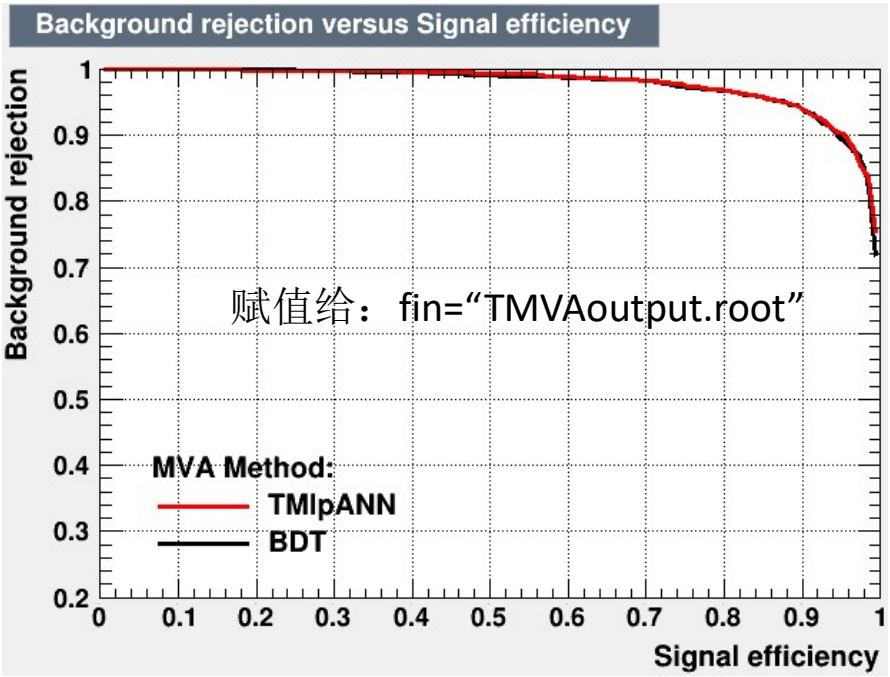
```
ActionButton( cbar,
              title,
              Form( ".x mvas.C(\"%s\",1)", fName ),
              "Plots the probability of each classifier for the test data (macro mvas.C(...,1))",
              buttonType, defaultRequiredClassifier );
```

点击(5b)TMVA的训练结果识别信号与压制本底的效率

$ROOTSYS/tmva/test/efficiencies.C

```
void efficiencies( TString fin = "TMVA.root",
```





左图：运行**$ROOTSYS/tmva/test/TMVAClassification.C**生成**TMVA.root**，然后运行**efficiencies.C**

右图：运行本讲座程序**TrainTMVA.C**，生成**TMVAoutput.root**,然后运行**efficiencies.C**，注意输入的**root**文件名差异