

粒子物理与核物理实验中的 数据分析

杨振伟 王喆
清华大学

第六讲：ROOT在数据分析中的应用(3)

本讲要点

- 直方图的操作
- 直方图的运算
 - 加减乘除: Add, Divide, ...
 - 归一化: Scale
- ROOT中直方图拟合
 - `h1->Fit()`;
- Graph的拟合
- 神经网络分析例子
- 练习画一些测试统计量

直方图归一化(1)

<http://root.cern.ch/root/html522/TH1.html#TH1:Scale>

直方图的归一化

```
void TH1::Scale(Double_t c1, Option_t *option)
```

默认c1=1，把直方图每个区间的值(BinContent)乘以c1

假设 `sum=h1->Integral()`

`h1->Scale(c1)`之后，

$$h1->Integral() = c1 * sum$$

不加参数时，`h1->Scale()` 没有变化(默认c1=1)

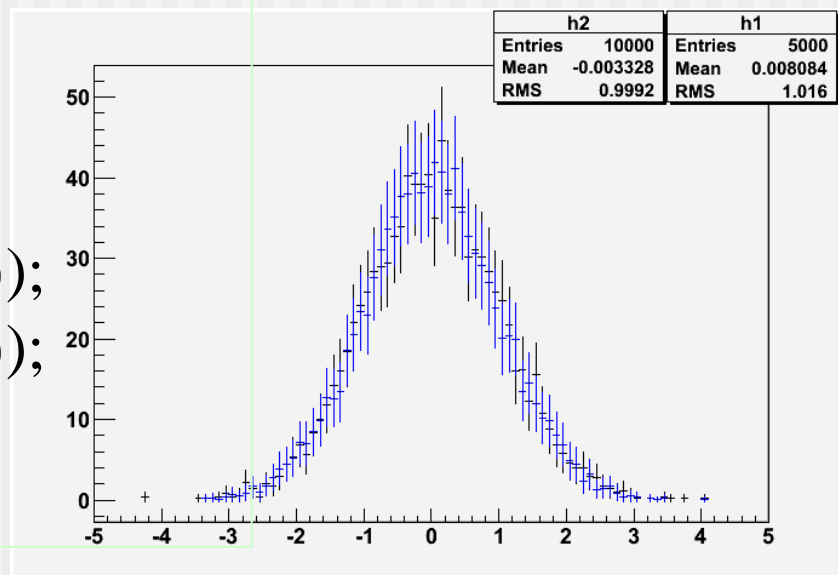
"归一化"后，不仅BinContent变化了， BinError也变化了

直方图的归一化(2)

归一化常用于比较两种分布，找出区别。

所以，将2个直方图归一化到积分相同进行比较才直观。

```
root[0]TH1F *h1=new TH1F("h1","",100,-5,5);
root[1]TH1F *h2=new TH1F("h2","",100,-5,5);
root[2]h1->FillRandom("gaus",5000);
root[3]h2->FillRandom("gaus",10000);
root[4]float norm=1000;
root[5]h1->Scale(norm/h1->Integral());
root[6]h2->Scale(norm/h2->Integral());
root[7]h1->Draw("e");
root[8]h2->Draw("esames");
```



注意Draw()函数的选项

"归一化"之后，h1或h2->Integral()==norm
在同一张图上可以看出比较2个分布的差别。

直方图四则运算(1)

重要提示:

1. 对直方图进行四则运算操作, 一定要想明白运算的意义
比如两个直方图的相加与两个随机变量的卷积有什么区别
2. 两个直方图的四则运算, 区间大小和区间数相同才有意义
四则运算"加减乘除"分别对应

统计量(BinContent)的相加、相减、相乘、相除

3. 如果需要正确处理统计误差, 需要在对ROOT脚本中
调用TH1的某个静态成员函数, 即

TH1::SetDefaultSumw2();

不一定一直正确!

```
void SetDefaultSumw2(Bool_t sumw2 = kTRUE)  
//static function. When this static function is called with  
sumw2=kTRUE, all new histograms will automatically  
activate the storage of the sum of squares of errors,  
ie TH1::Sumw2 is automatically called.
```

直方图的四则运算(2)

相加： 常用于相同实验的数据叠加， 增加统计量。

.....

```
root[1]TH1::SetDefaultSumw2();
```

```
root[1]TH1F *h3=new TH1F(*h1);
```

```
root[2]h3->Add(h1,h2,a,b);
```

结果： h3的BinContent被 $a*h1+b*h2$ 替换， 一般 $a=b=1$

相减： 常用于从实验测量的分布中扣除本底。

.....

```
root[1]TH1F *h3=new TH1F(*h1);
```

```
root[2]h3->Sumw2();//也可在定义h3前TH1::SetDefaultSumw2();
```

```
root[3]h3->Add(h1,h2,a,-b);
```

结果： h3的BinContent被 $a*h1+b*h2$ 替换， 一般 $a=-b=1$

误差： $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2} = \sqrt{n_1 + n_2}$ (假设h1和h2独立)

直方图的四则运算 (3)

相除

常用于效率的计算。

```
root[1]TH1F *h3=new TH1F(*h1);  
root[2]h3->Sumw2();  
root[3]h3->Divide(h1,h2,a,b);  
root[4]h3->Divide(h1,h2,a,b);
```

$$\sigma = \frac{n_1}{n_2} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}} \quad (\text{h1和h2独立})$$

思考：如果h1和h2不独立，怎么办？比如h1包含于h2

```
root[4]h3->Divide(h1,h2,a,b,"B");
```

$$\sigma = \sqrt{\frac{\frac{n_1}{n_2} (1 - \frac{n_1}{n_2})}{n_2}}$$

二项分布误差


相乘

常用于对分布进行诸如效率等的修正。

```
root>TH1F *h3=new TH1F(*h1);  
root>h3->Sumw2();  
root>h3->Multiply(h1,h2,a,b);
```

$$\sigma = n_1 n_2 \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

直方图四则运算的误差处理

 虽然ROOT都提供了较完善的一维直方图运算功能，但对最终结果的误差一定要仔细检查。很多情况下，用户需要从图中读出各频数数值与误差值，并确认运算无误。

自己来进行更复杂的操作，或自己设定误差：

GetBinContent (nbin)

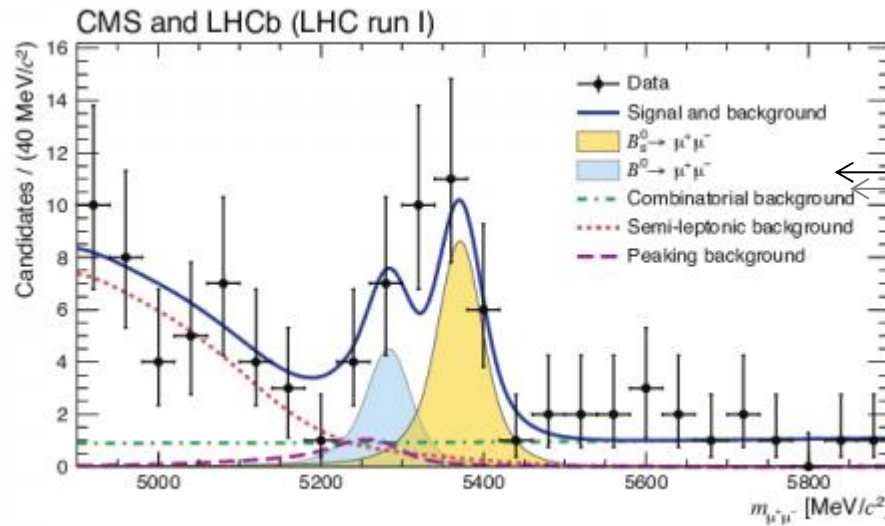
GetBinError (nbin)

SetBinContent (nbin)

SetBinError (nbin)

好多时候误差不是根号n那么简单！

直方图画图的一些技巧



TLegend

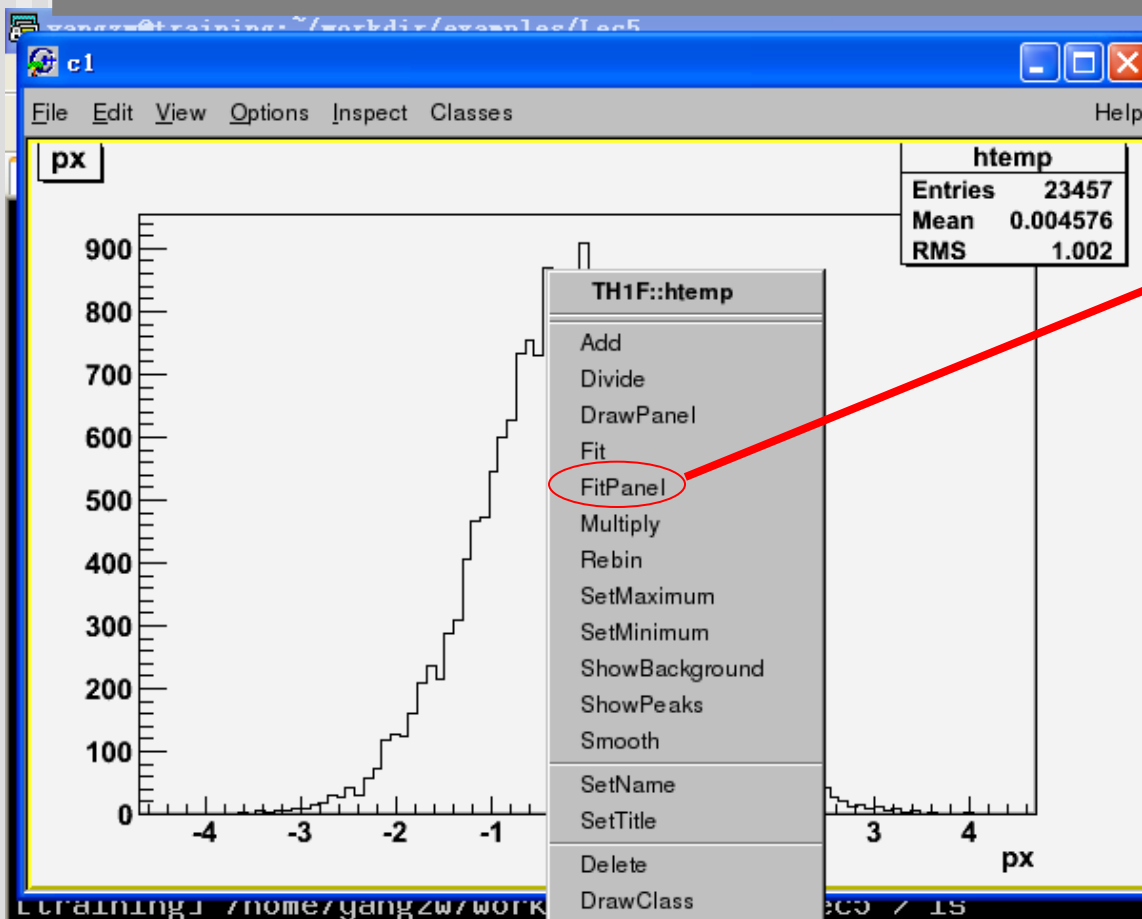
添加图标及
描述

```
hSim->GetXaxis()->SetTitle("Visible Energy [MeV]");  
hSim->GetYaxis()->SetTitle("Entries / 0.01 MeV");  
hSim->GetXaxis()->SetRangeUser(0.1,20);  
hSim->SetTitle(" Simulation and Fit Result");  
hSig->SetLineColor(kRed);  
hSig->SetLineWidth(2);  
hSig->Rebin(10);  
hSig->Draw("same");
```

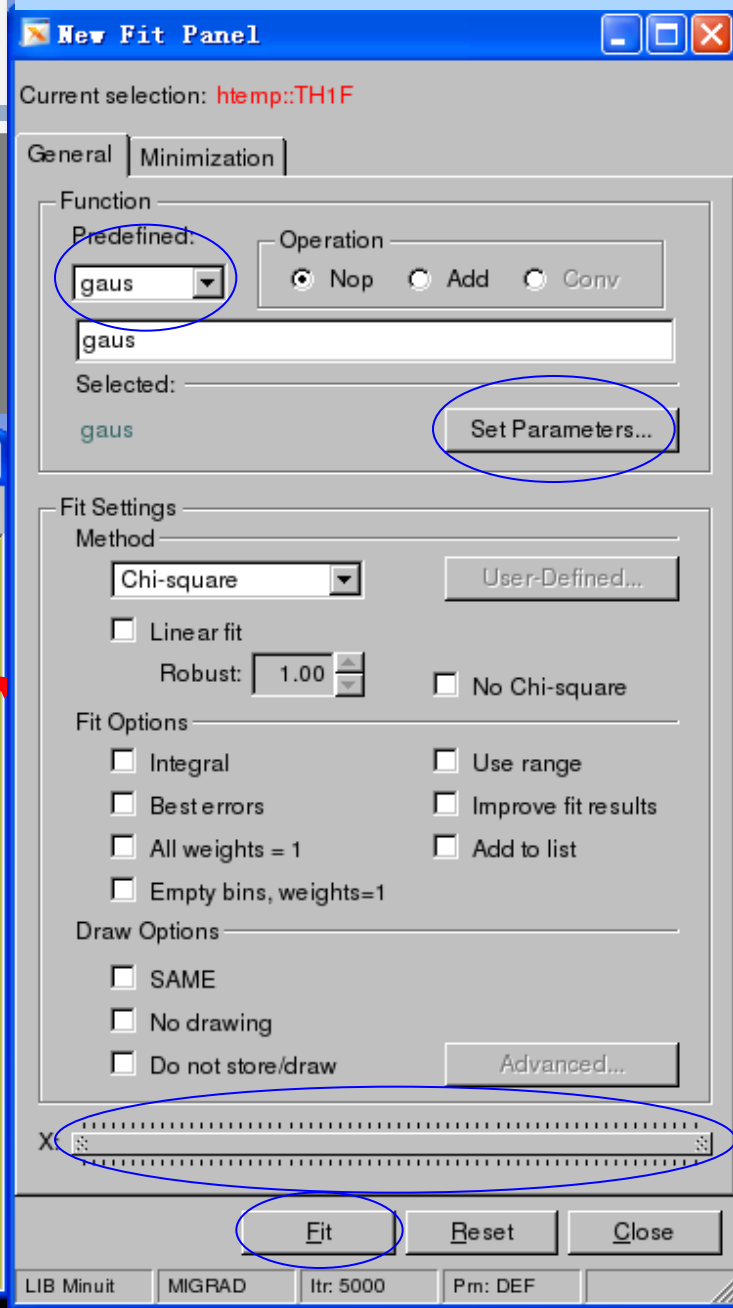
让直方图更清
晰，明确，正
确

拟合直方图(1)

将鼠标放到直方图上，右键，出现直方图操作选项，选择**FitPanel**，可以在**FitPanel**中选择拟合的各个选项，比如用什么**函数**拟合，拟合的**区间**，等等。

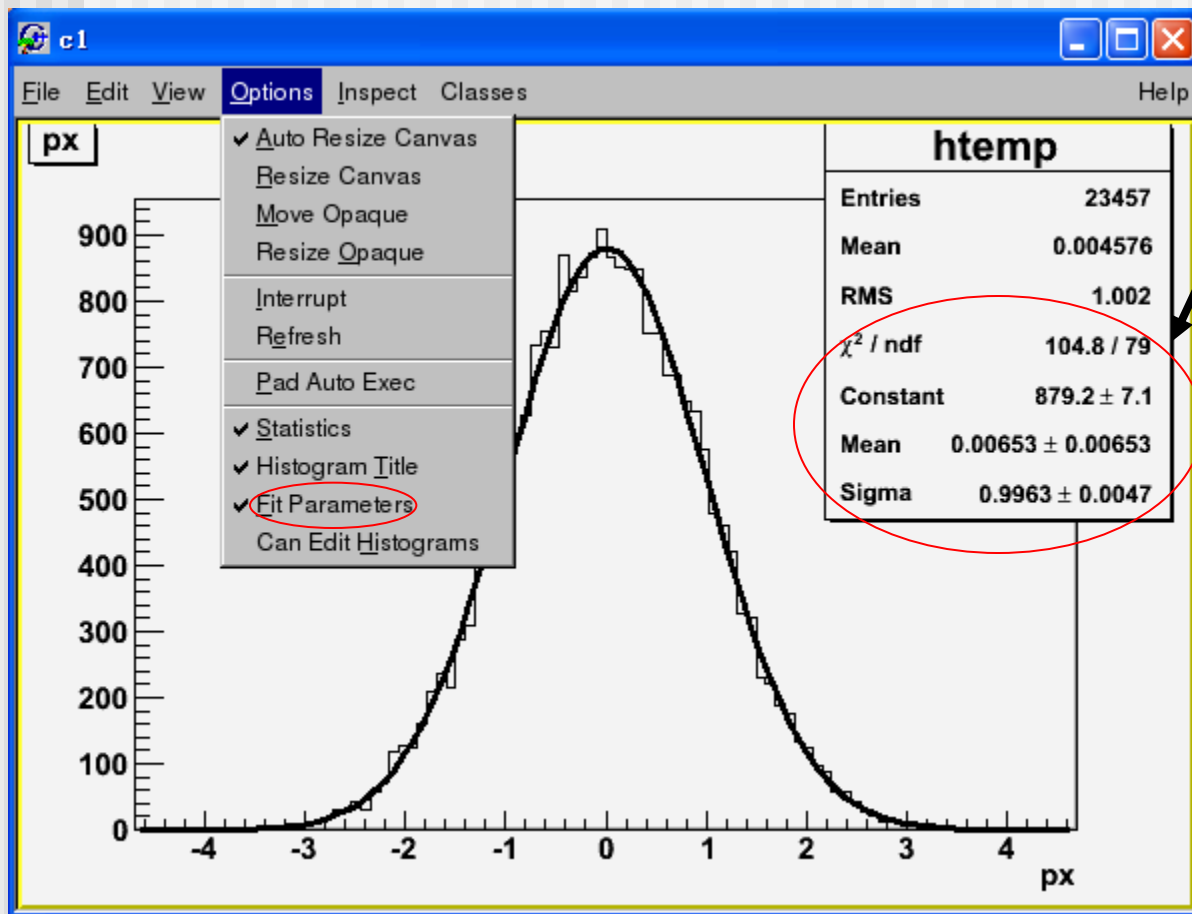


Context sensitive menu



拟合直方图(2)

用默认的高斯拟合，并在Options菜单中选上Fit Parameters选项，可以看到拟合的结果。



拟合结果给出了高斯分布的3个参数：常系数、均值、均方差，以及拟合的好坏 χ^2/ndf

并不推荐这种拟合方式：

- 1) 不适合自定义函数拟合
- 2) 不适合批处理

拟合直方图(3) 简单函数 .../Lec5/ex51.C

```
hpx->Fit("gaus");  
hpx->Fit("gaus","",",-3,3);
```

自定义拟合函数

```
TF1 *fcn = new TF1("fcn","gaus",-3,3);  
hpx->Fit(fcn,"R");
```

gStyle->SetOptFit();//设置拟合选项

拟合前往往需要给出合理的参数初值

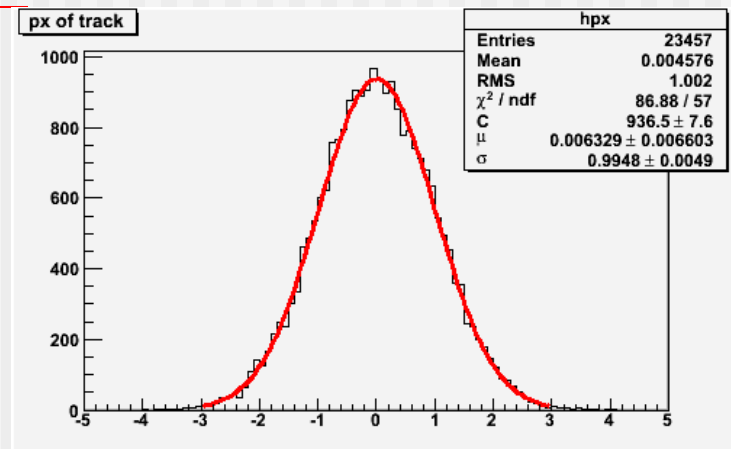
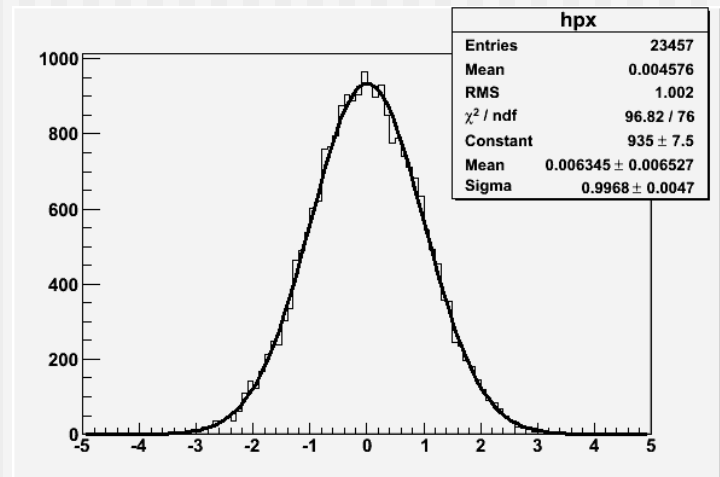
fcn->SetParameters(500,mean,sigma);

拟合后取出拟合得到的参数

Double_t mypar[3];

fcn->GetParameters(&mypar[0]);

fcn->GetChisquare();



运行: root -l

root [0] .L ex51.C

root [1] ex51r()

root [2] ex51r2()

用自定义的函数拟合直方图

拟合直方图(3) 复杂函数

.../Lec5/ex52.C

共振峰(Breit-Wigner分布)加上二次函数本底的拟合(一共6个参数)

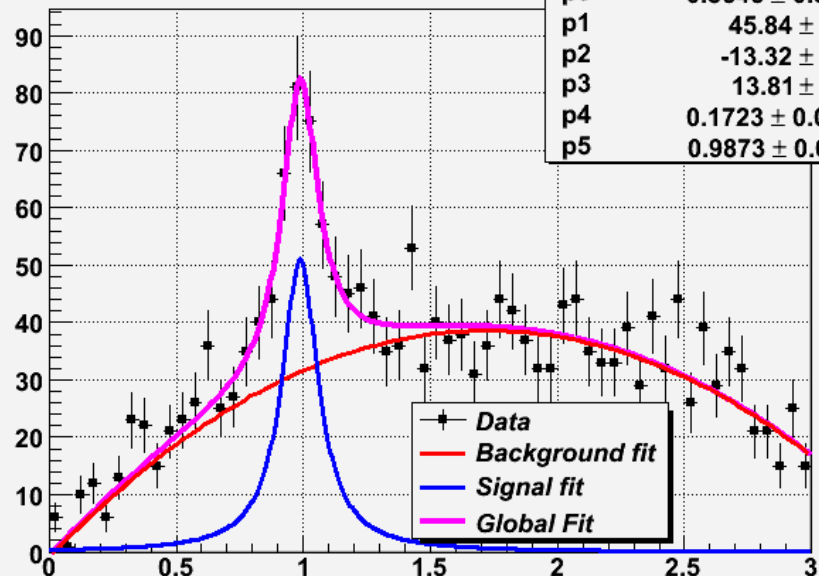
这是下面例子的简化版:

`$ROOTSYS/tutorials/fit/FittingDemo.C`

先自定义本底函数(background)和共振峰函数(lorentianPeak), 再定义这两个函数的和为拟合函数:fitFunction

利用fitFunction定义TF1函数

Lorentzian Peak on Quadratic Bkgd



```
TF1 *fitFcn = new TF1("fitFcn",fitFunction,0,3,6);
```

这里指定函数区间为0-3, 6个参数

`fitFcn->SetParameter(4, 0.2);` 为某个参数设初值(width)

`fitFcn->SetParLimits(5, 0.6,1.4);` 为某参数设置取值范围

运行: `root -l`
`root [0] .L ex52.C`

注意TLegend的使用

拟合任意图形

.../Lec5/graphfit.C

当直方图的形式已经不能满足你的需求，
怎么处理任意测量结果呢？

```
const Int_t n1 = 10;  
Double_t x1[] = {-0.1,0.05,0.25,0.3  
Double_t y1[] = {-1,2.9,5.6,7.4,9,9.  
Double_t ex1[] = {.05,.1,.07,.07,.04,  
Double_t ey1[] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
```

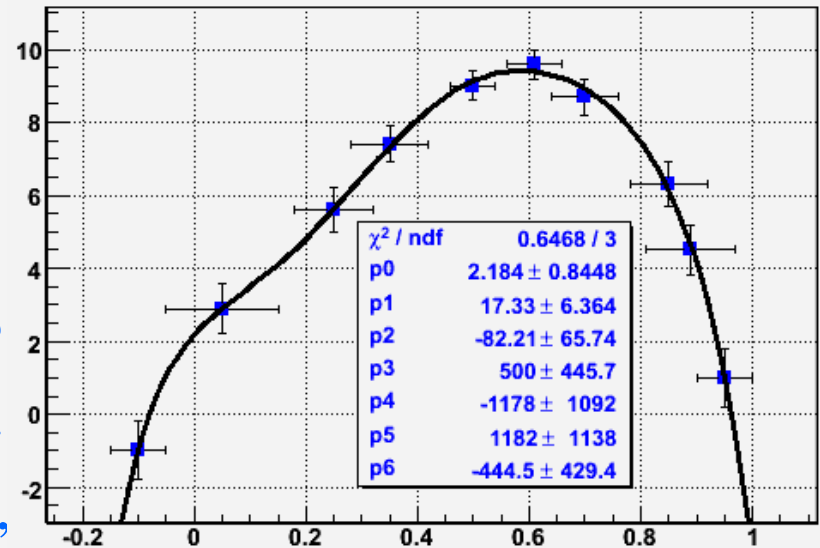
```
TGraphErrors *gr1 = new TGraphErrors(n1,x1,y1,ex1,ey1);
```

```
gr1->Fit("pol6","q");
```

```
gr1->Draw("Ap");
```

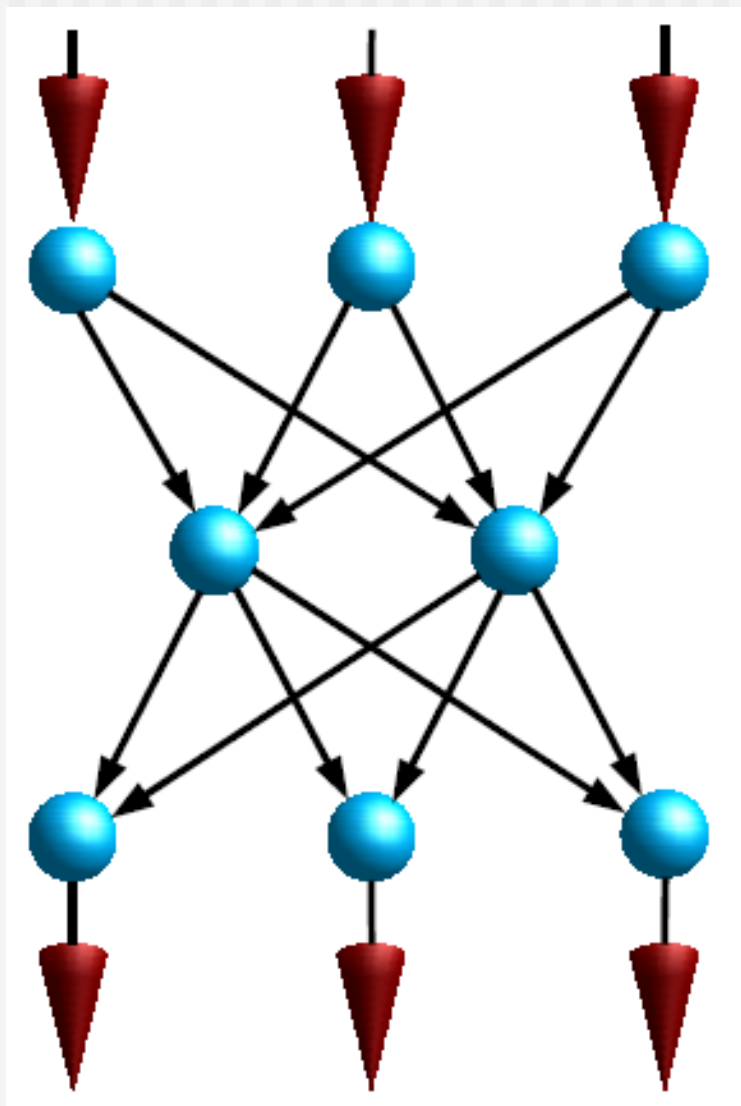
画出来，一定要加Ap

用一个6次多项式拟合



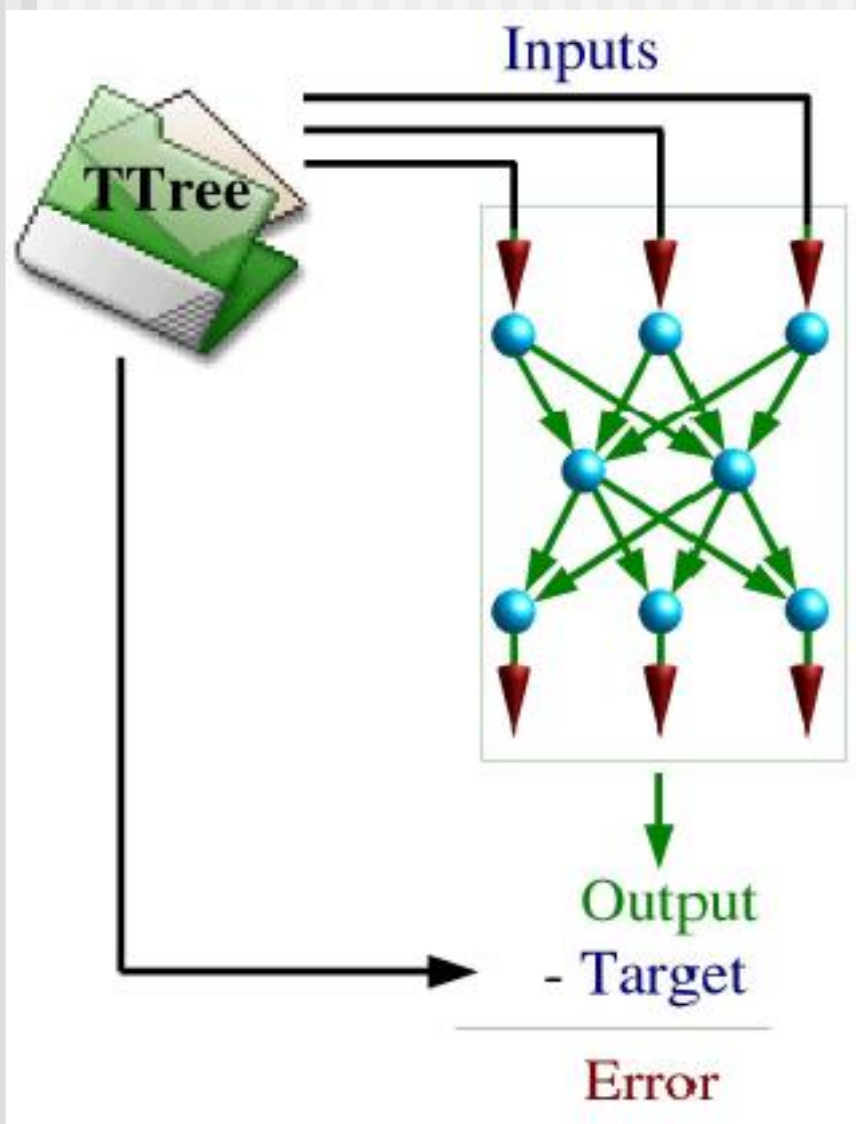
生成一个TGraph: 定义它的数据量，
各个数据点的值和误差

神经网络分析（一）



1. 输入层，有多个变量可以用来鉴别，用户设置
2. 中间层，隐藏层，层数和节点数由用户设置
3. 输出层，在做事例鉴别的时候，一般只设定一个输出节点

神经网络分析（二）



Root中的实现

- 有一个学习过程，使最终的变换结果与指定值之间的误差最小

$$\Delta = \frac{1}{2} \sqrt{\sum_i \Delta_i^2}$$

- 使用者输入一个TTree，用来神经网络的学习。它包含多个分支，是可供分析的变量，并有一个分支或几个用来甄别是各种假设的目标值。在一个输出神经元的时候，信号常用1来表示，而本底用0。

神经网络分析（三）

分析root的例子：tutorials/mlp/mlpHiggs.C

```
TMultiLayerPerceptron *mlp =  
    new TMultiLayerPerceptron("@msumf,@ptsumf,@acolin:5:3:type",  
                               "ptsumf",simu,"Entry$%2","(Entry$+1)%2");
```

- 用来做学习的Tree是simu
- msumf, ptsumf, acolin, acopl分别是用来做Higgs判选的变量
- type是用来判选的变量，1或0
- 有两个隐藏层，分别为5，3个节点
- 每个事例的权重由ptsumf指定
- 最后事例号为偶数的用来做学习，事例号为奇数的做演示

神经网络分析（四）

```
mlp->Train(ntrain, "text,graph,update=10");  
mlp->Export("test","python");
```

- 开始神经网络的自学习，共ntrain次，而且每10次显示一下总误差的估计情况
- 最后输出python的函数形式，共以后使用，也可以是C++的函数。

作业及课后自行体会内容， 反馈

- 产生1000个有100个事例的高斯分布，
 $\mu=0$, $\sigma=1$
 1. 用高斯函数来拟合，全部参数自由，记录每次的拟合结果， μ , σ , chi^2/ndf
 - 画上述三个变量的分布
 2. 仍用高斯函数拟合，固定 $\mu=0.05$, σ 自由，记录得到的 σ , chi^2/ndf
 - 画上述两个变量的分布
- 对比上述两种拟合的 σ , chi^2/ndf 的分布，画在一起