首先需要登录交大服务器

输入

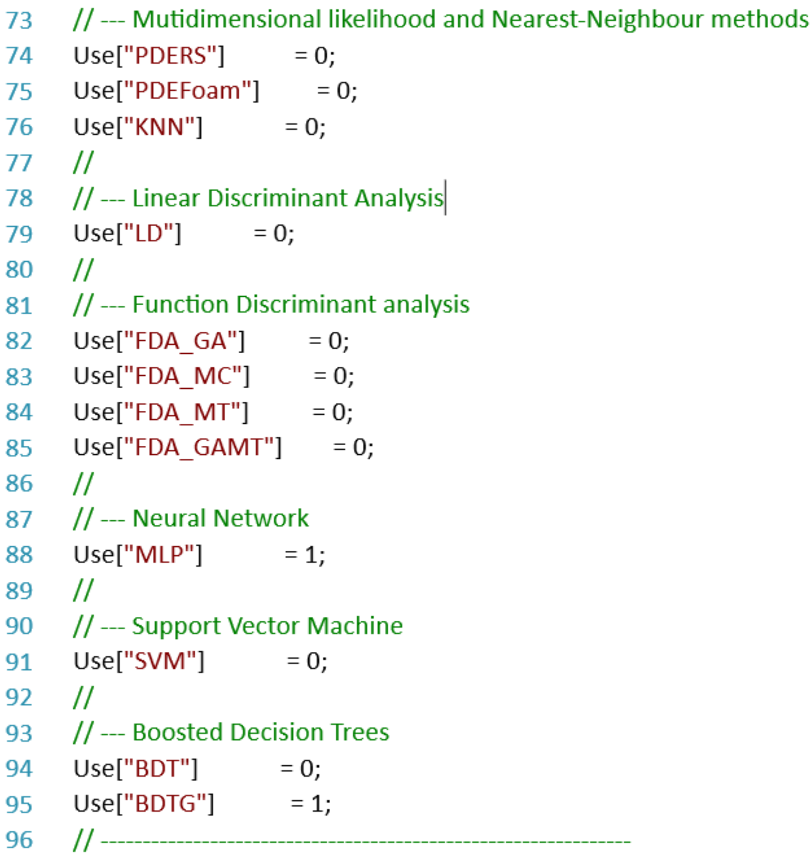
ssh -Y inpactest@bl-1-1.physics.sjtu.edu.cn
password: inpac123456

Step1 Training

1. 首先进入/Erec_tuto/tmva 路径下的 TMVARegression.C

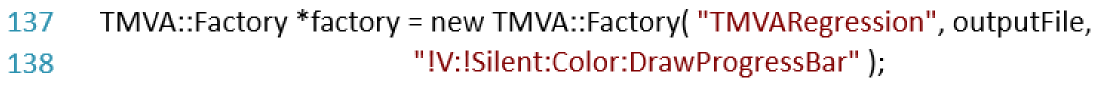
```
cd /Erec_tuto/tmva
vi TMVARegression.C
```

2. 设定 MVA 中所使用的方法。1 表示打开，0 表示关闭。本次所使用的方法为 MLP 和 BDTG 所以请确保 MLP 和 BDTG 方法打开。将程序调整为下图所示。



```
// --- Mutidimensional likelihood and Nearest-Neighbour methods
Use["PDERS"]         = 0;
Use["PDEFoam"]       = 0;
Use["KNN"]           = 0;
//
// --- Linear Discriminant Analysis
Use["LD"]            = 0;
//
// --- Function Discriminant analysis
Use["FDA_GA"]        = 0;
Use["FDA_MC"]        = 0;
Use["FDA_MT"]        = 0;
Use["FDA_GAMT"]      = 0;
//
// --- Neural Network
Use["MLP"]           = 1;
//
// --- Support Vector Machine
Use["SVM"]           = 0;
//
// --- Boosted Decision Trees
Use["BDT"]           = 0;
Use["BDTG"]          = 1;
// ---------------------------------------------------------------
```

3. 创建 factory object. 之后你可以对不同的 factory 选择不同的 TMVA 的方法。如果程序用作回归，请将第一个选项调整为“TMVARegression”，用作分类请调整为“TMVAClassification”。本次重建能量用作回归，所以将程序调整至下图所示。



```
TMVA::Factory *factory = new TMVA::Factory( "TMVARegression", outputFile,
                                            "!V:!Silent:Color:DrawProgressBar" );
```

4. 定义输入变量，这些输入变量将被应用到 MVA 的 training 中。你也可以使用一些变量的组合类似于"3*var1/var2*abs(var3)" 。下图为我们所使用到的 12 个输入变量。

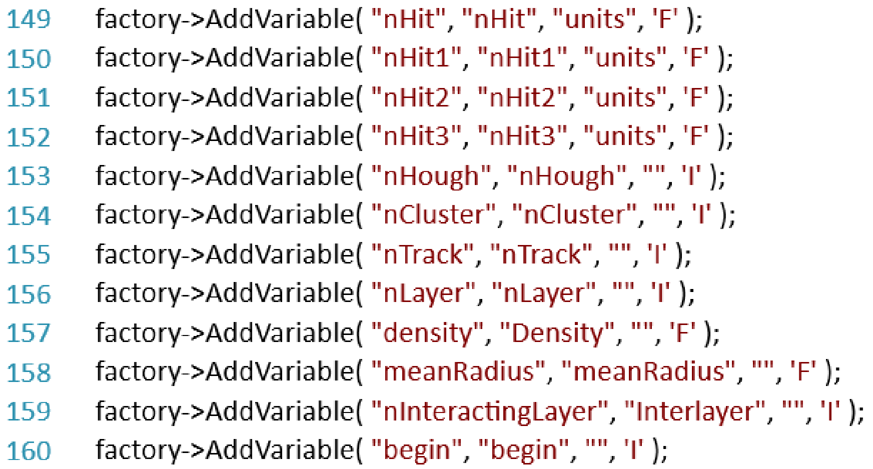
请确保你的程序和下图相同。

-第一个选项是该变量在 root 文件中的名字。

-第二个选项是你希望在输出结果中显示的名字。

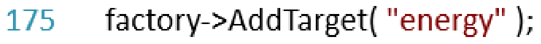
-第三个选项是该变量的单位。

-第四个选项是该变量的数据类型。“I”表示“Int”，“F”表示“Float”……



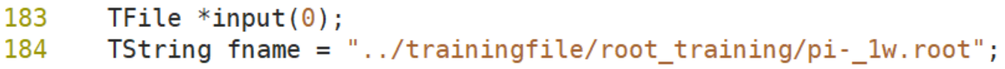
```
factory->AddVariable( "nHit", "nHit", "units", 'F' );
factory->AddVariable( "nHit1", "nHit1", "units", 'F' );
factory->AddVariable( "nHit2", "nHit2", "units", 'F' );
factory->AddVariable( "nHit3", "nHit3", "units", 'F' );
factory->AddVariable( "nHough", "nHough", "", 'I' );
factory->AddVariable( "nCluster", "nCluster", "", 'I' );
factory->AddVariable( "nTrack", "nTrack", "", 'I' );
factory->AddVariable( "nLayer", "nLayer", "", 'I' );
factory->AddVariable( "density", "Density", "", 'F' );
factory->AddVariable( "meanRadius", "meanRadius", "", 'F' );
factory->AddVariable( "nInteractingLayer", "Interlayer", "", 'I' );
factory->AddVariable( "begin", "begin", "", 'I' );
```

5. 定义作为 target 的变量。本次是为了重建能量。因此将 energy 作为程序的 target。



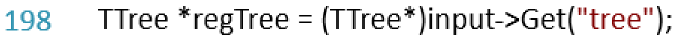
```
factory->AddTarget( "energy" );
```

6. 添加用作 TMVA training 的 MC 数据文件。在路径/Erec_tuto/trainingfile/root_training 下面，我们为你提供了四个 pi-的 root 文件，它们的区别仅在于数据量的不同，分别为 1k，5k，1w，5w。在本程序中我们添加的为数据量为 1w 的文件，但你也可以改变为其他文件，观察数据量不同所导致的结果的差异，当然数据量越大，程序运行的时间就越长。



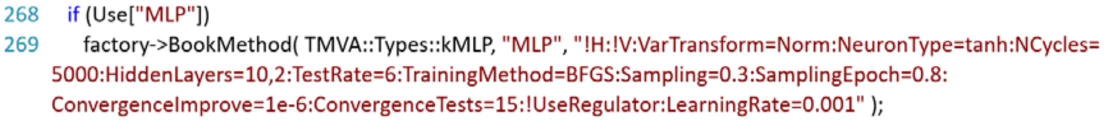
```
TFile *input(0);
TString fname = "../trainingfile/root_training/pi-_1w.root";
```

7. 添加所使用的 root 文件中的 tree。



```
TTree *regTree = (TTree*)input->Get("tree");
```

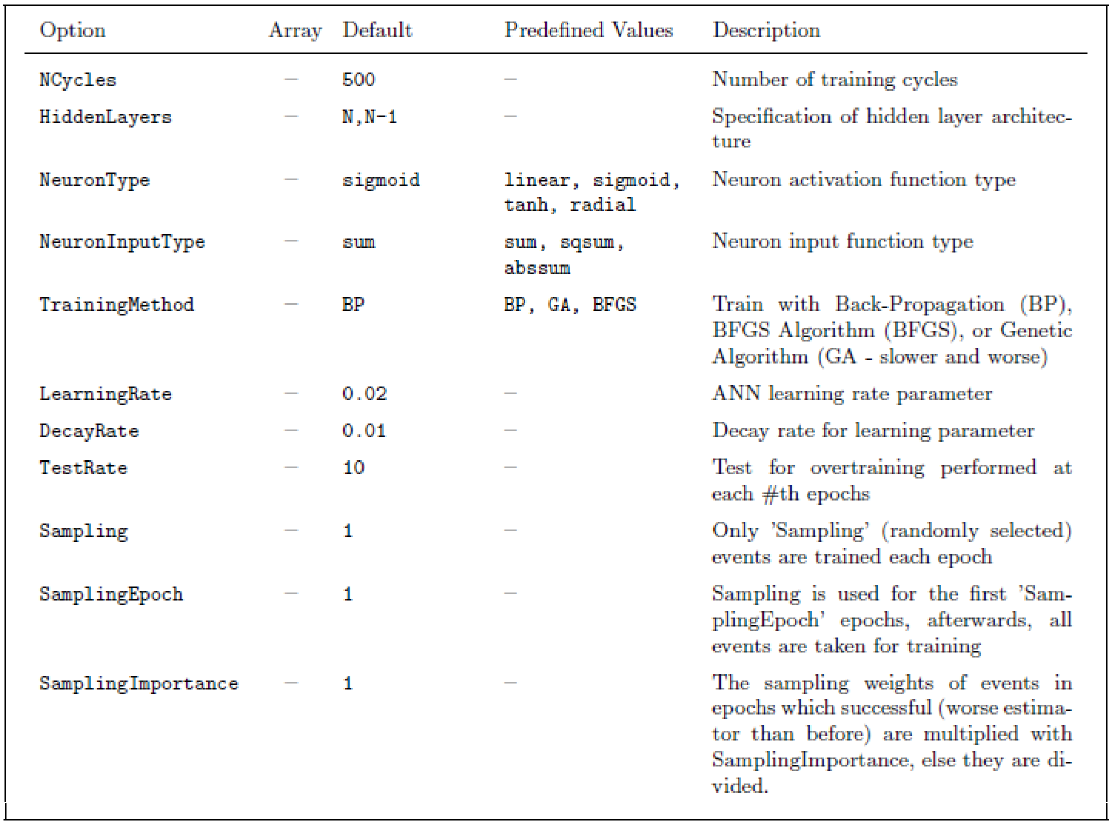
8. 设置 TMVA 中各个方法的参数。本次所示用的方法是 MLP 和 BDTG，因此只需要设置这两个方法的参数即可。下图为本程序中 MLP 方法的设置选项。



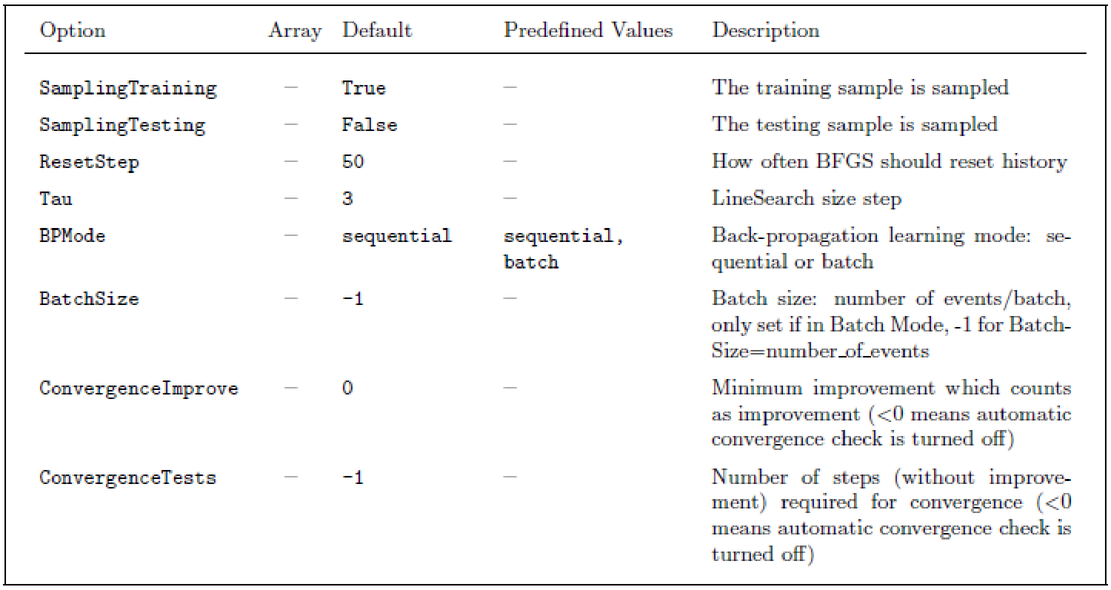
```
if (Use["MLP"])
   factory->BookMethod( TMVA::Types::kMLP, "MLP", "!H:!V:VarTransform=Norm:NeuronType=tanh:NCycles=
5000:HiddenLayers=10,2:TestRate=6:TrainingMethod=BFGS:Sampling=0.3:SamplingEpoch=0.8:
ConvergenceImprove=1e-6:ConvergenceTests=15:!UseRegulator:LearningRate=0.001" );
```

下表中为 MLP 方法中各个参数的意义和解释，你可以根据下表中的信息，对上图中的各个参数进行一定的修改和调试。你也可以参照 TMVA_Users guide 进行进一步了解。

下表中为 BDTG 方法中各个参数的意义和解释，你可以根据下表中的信息，对上图中的各个参数进行一定的修改和调试。你也可以参照 TMVA_Users guide 进行进一步了解。

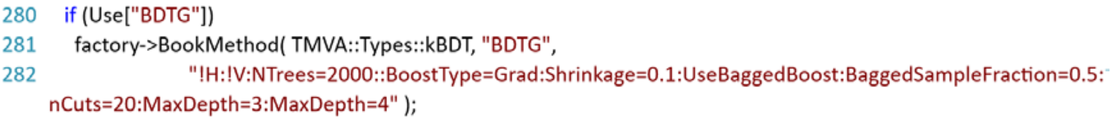


Option	Array	Default	Predefined Values	Description
NCycles	–	500	–	Number of training cycles
HiddenLayers	–	N,N-1	–	Specification of hidden layer architecture
NeuronType	–	sigmoid	linear, sigmoid, tanh, radial	Neuron activation function type
NeuronInputType	–	sum	sum, sqsum, abssum	Neuron input function type
TrainingMethod	–	BP	BP, GA, BFGS	Train with Back-Propagation (BP), BFGS Algorithm (BFGS), or Genetic Algorithm (GA - slower and worse)
LearningRate	–	0.02	–	ANN learning rate parameter
DecayRate	–	0.01	–	Decay rate for learning parameter
TestRate	–	10	–	Test for overtraining performed at each #th epochs
Sampling	–	1	–	Only 'Sampling' (randomly selected) events are trained each epoch
SamplingEpoch	–	1	–	Sampling is used for the first 'SamplingEpoch' epochs, afterwards, all events are taken for training
SamplingImportance	–	1	–	The sampling weights of events in epochs which successful (worse estimator than before) are multiplied with SamplingImportance, else they are divided.



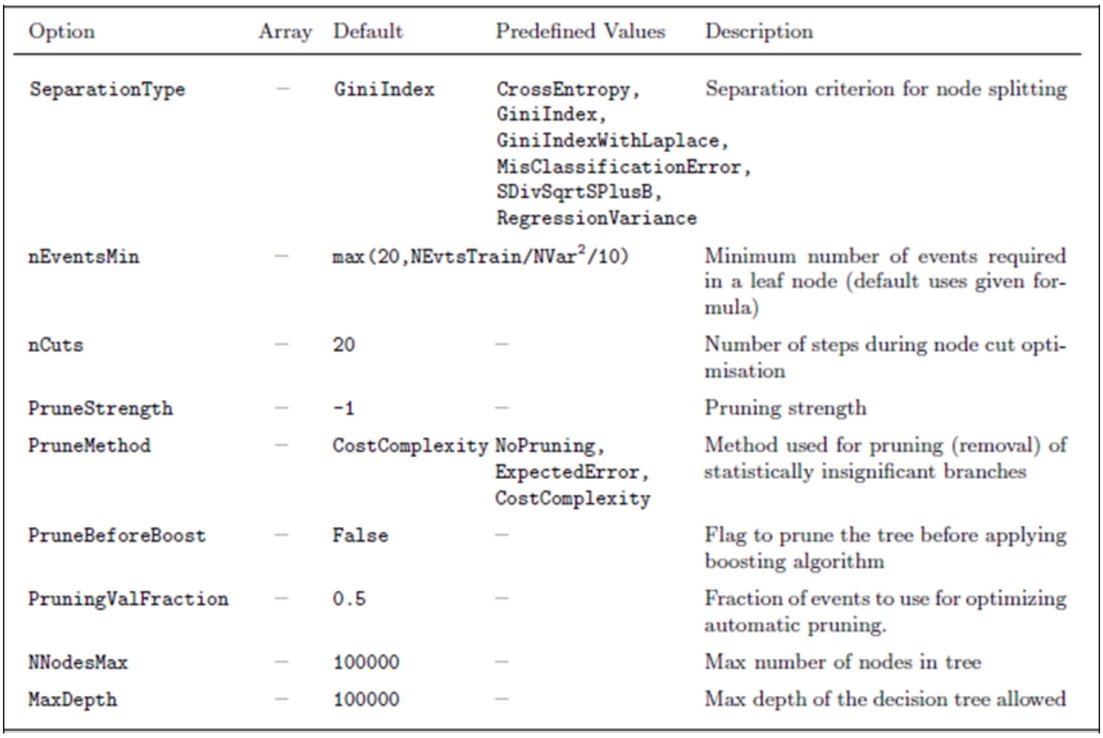
Option	Array	Default	Predefined Values	Description
SamplingTraining	–	True	–	The training sample is sampled
SamplingTesting	–	False	–	The testing sample is sampled
ResetStep	–	50	–	How often BFGS should reset history
Tau	–	3	–	LineSearch size step
BPMode	–	sequential	sequential, batch	Back-propagation learning mode: sequential or batch
BatchSize	–	-1	–	Batch size: number of events/batch, only set if in Batch Mode, -1 for BatchSize=number_of_events
ConvergenceImprove	–	0	–	Minimum improvement which counts as improvement (<0 means automatic convergence check is turned off)
ConvergenceTests	–	-1	–	Number of steps (without improvement) required for convergence (<0 means automatic convergence check is turned off)

下图为本程序中 BDTG 方法的设置选项。

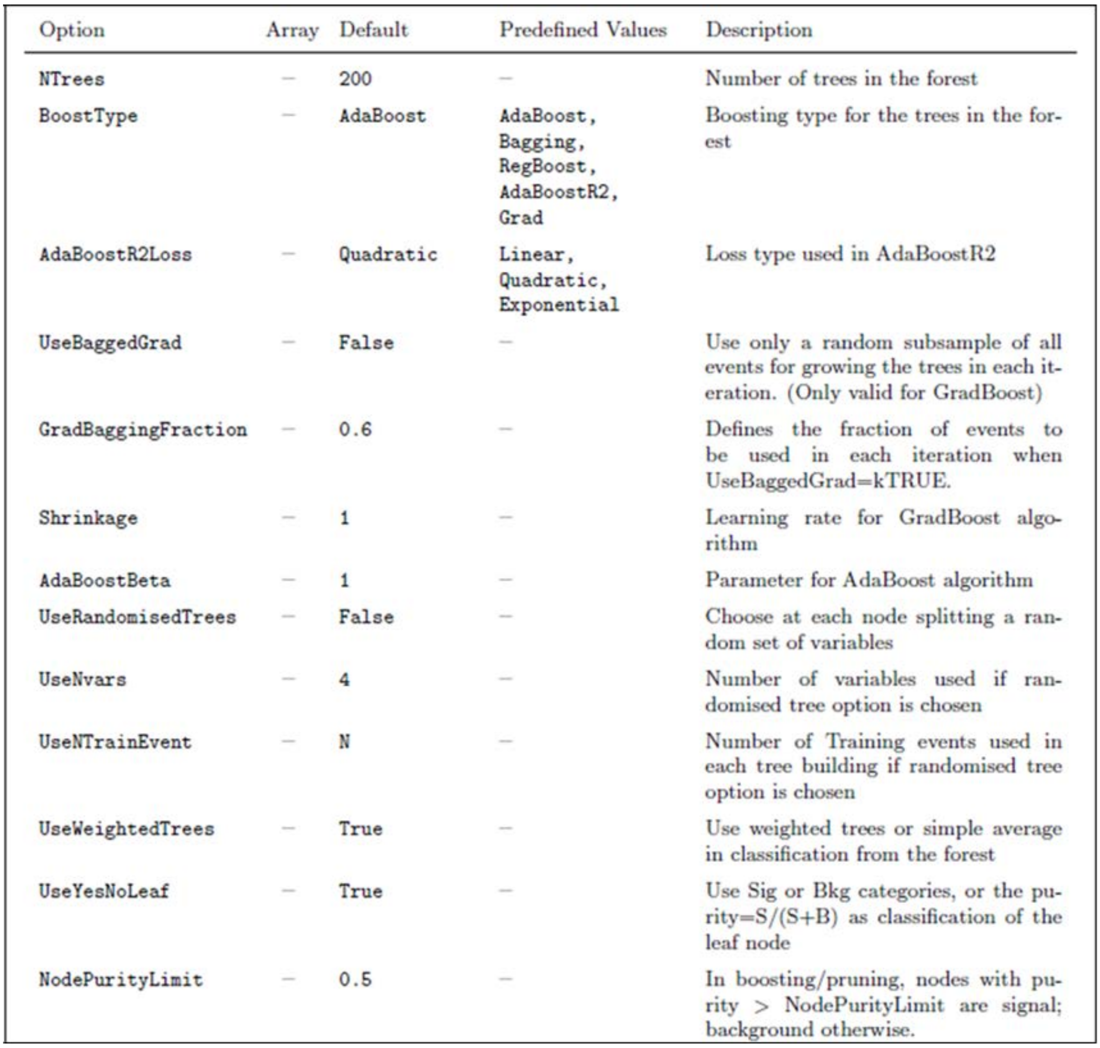


```
 if (Use["BDTG"])
   factory->BookMethod( TMVA::Types::kBDT, "BDTG",
              "!H:!V:NTrees=2000::BoostType=Grad:Shrinkage=0.1:UseBaggedBoost:BaggedSampleFraction=0.5:
nCuts=20:MaxDepth=3:MaxDepth=4" );
```

下表中为 BDTG 方法中各个参数的意义和解释，你可以根据下表中的信息，对上图中的各个参数进行一定的修改和调试。你也可以参照 TMVA_Users guide 进行进一步了解。



Option	Array	Default	Predefined Values	Description
SeparationType	–	GiniIndex	CrossEntropy, GiniIndex, GiniIndexWithLaplace, MisClassificationError, SDivSqrtSPlusB, RegressionVariance	Separation criterion for node splitting
nEventsMin	–	max(20,NEvtsTrain/NVar^2/10)		Minimum number of events required in a leaf node (default uses given formula)
nCuts	–	20	–	Number of steps during node cut optimisation
PruneStrength	–	-1	–	Pruning strength
PruneMethod	–	CostComplexity	NoPruning, ExpectedError, CostComplexity	Method used for pruning (removal) of statistically insignificant branches
PruneBeforeBoost	–	False	–	Flag to prune the tree before applying boosting algorithm
PruningValFraction	–	0.5	–	Fraction of events to use for optimizing automatic pruning.
NNodesMax	–	100000	–	Max number of nodes in tree
MaxDepth	–	100000	–	Max depth of the decision tree allowed

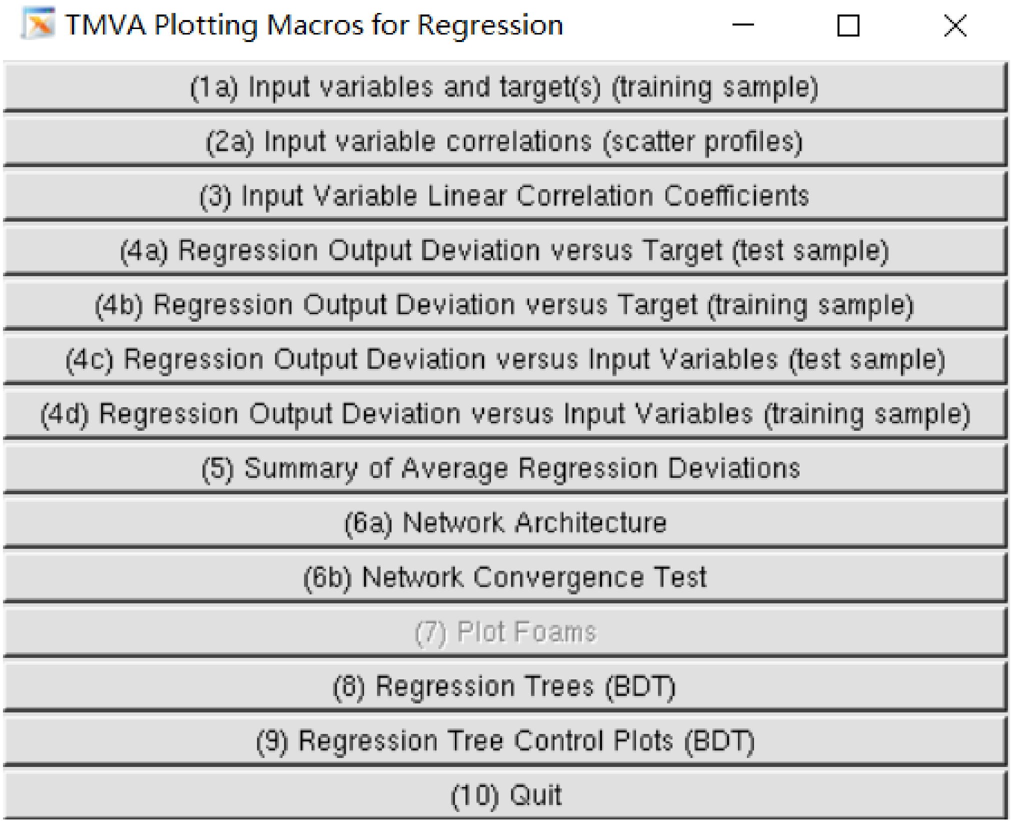


Option	Array	Default	Predefined Values	Description
NTrees	–	200	–	Number of trees in the forest
BoostType	–	AdaBoost	AdaBoost, Bagging, RegBoost, AdaBoostR2, Grad	Boosting type for the trees in the forest
AdaBoostR2Loss	–	Quadratic	Linear, Quadratic, Exponential	Loss type used in AdaBoostR2
UseBaggedGrad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

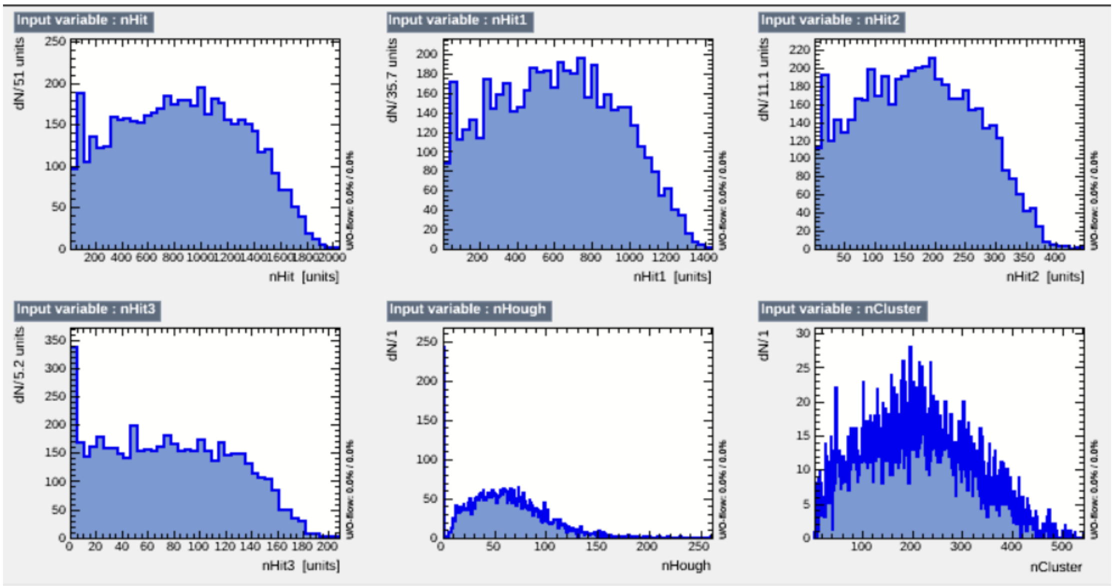
9. 到此为止能量重建 training 部分的程序已设定完毕，退出程序，运行程序即可。

```
root TMVARegression.C
```

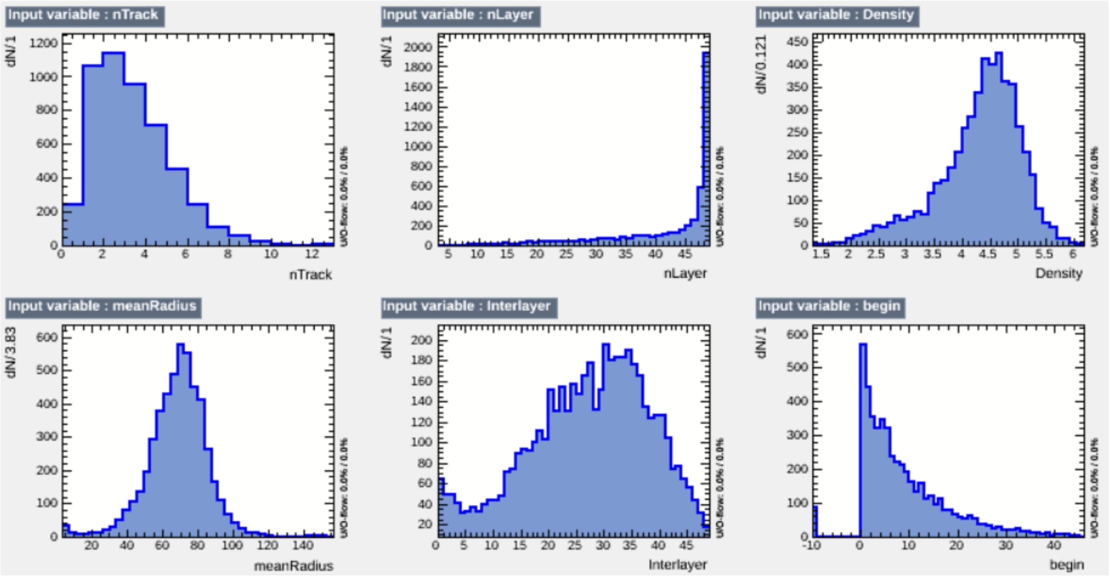
10. 输出结果。当程序运行结束之后会生成一个控制面板。你可以点击相应的选项来查看不同的输出结果。你需要主要了解的选项如下。


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

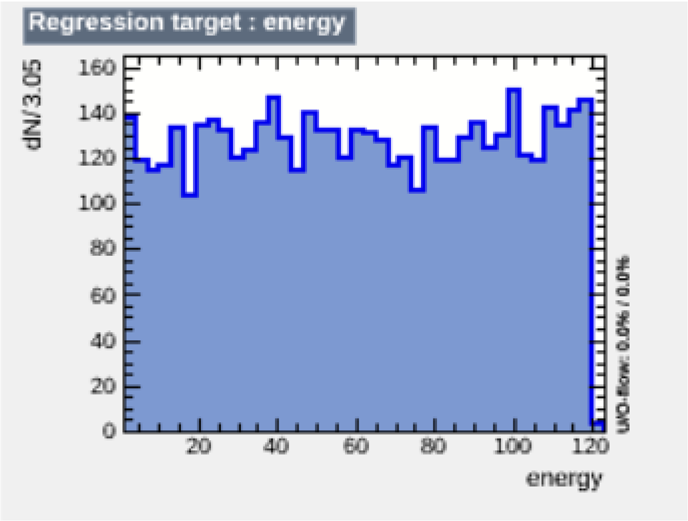
1） 1a 表示输入 variables 和 target 的分布。



（input variables）

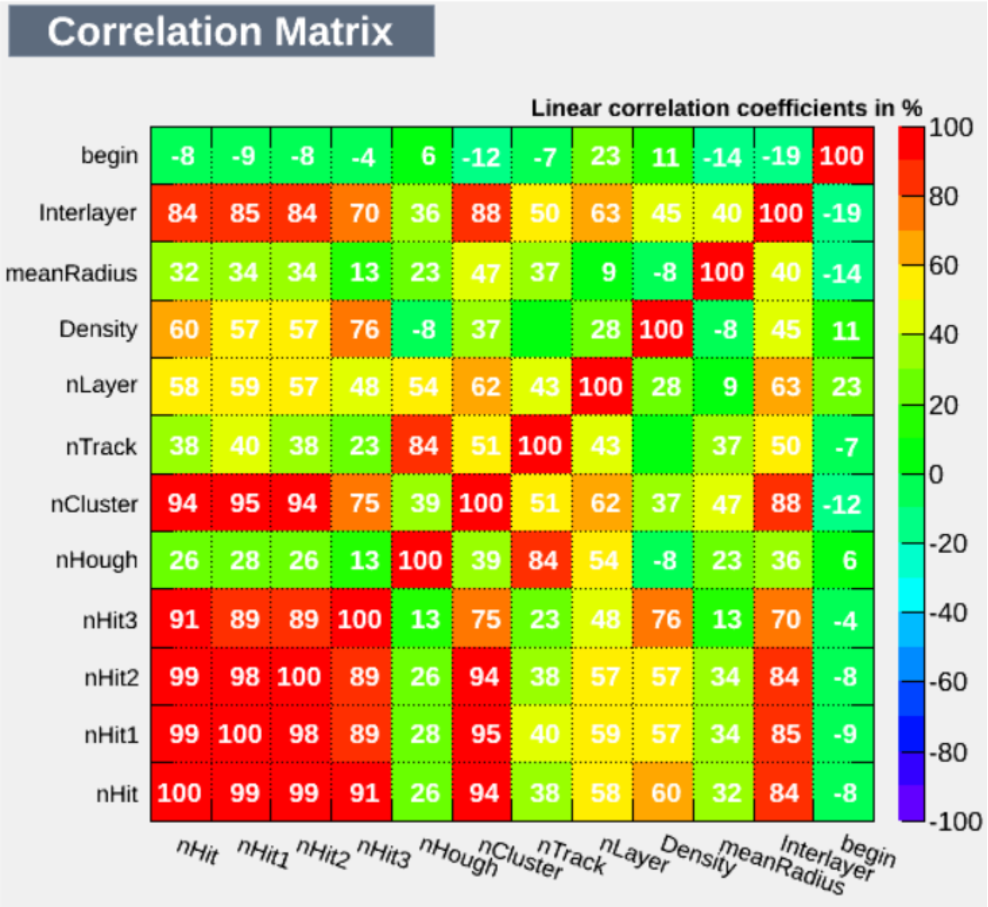


（input variables）



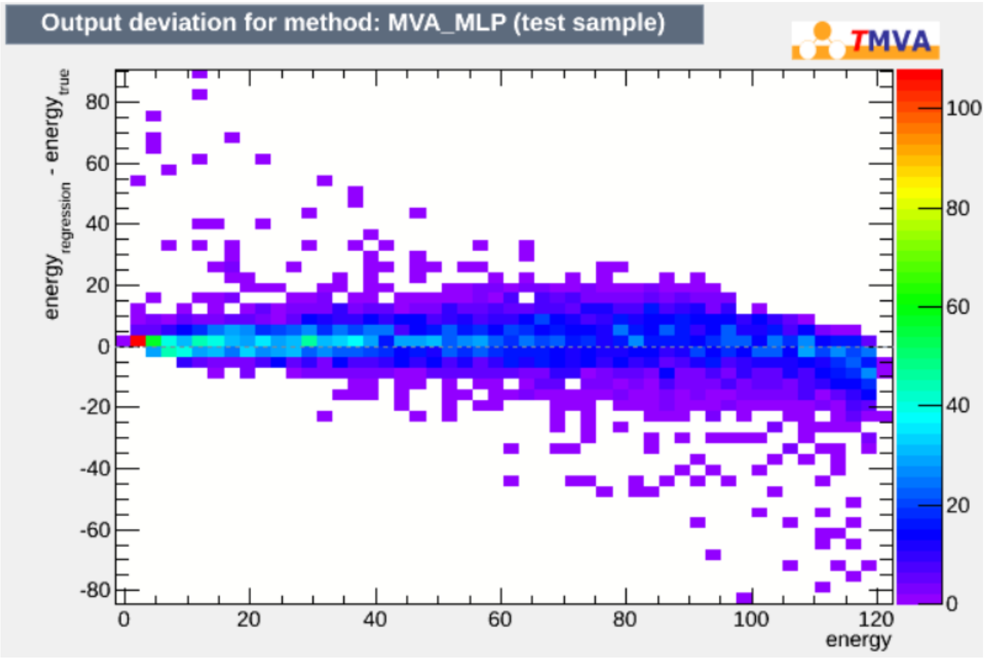
（target）

2） 3a 为输入各变量之间的相关度

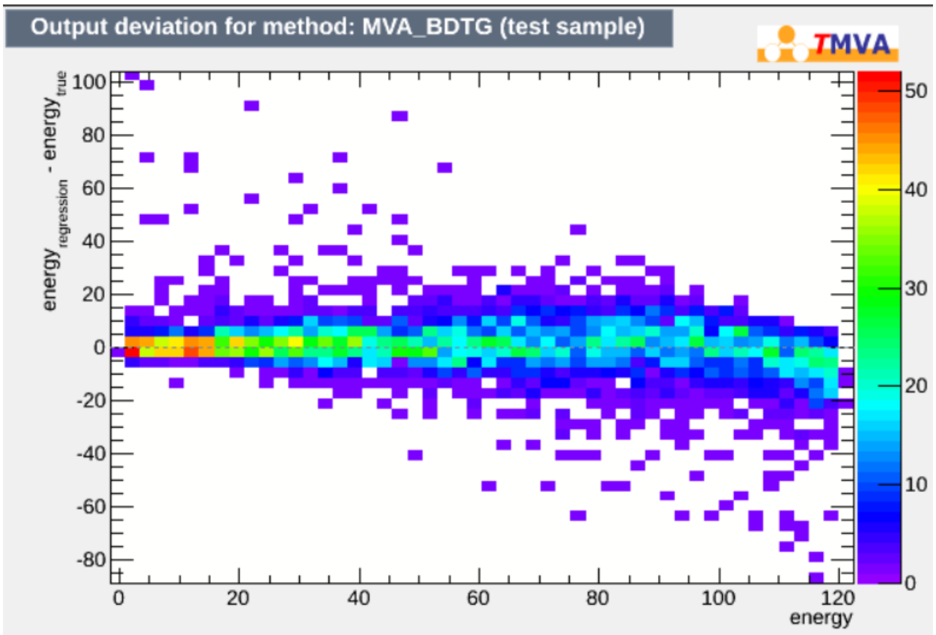


	nHit	nHit1	nHit2	nHit3	nHough	nCluster	nTrack	nLayer	Density	meanRadius	Interlayer	begin
begin	-8	-9	-8	-4	6	-12	-7	23	11	-14	-19	100
Interlayer	84	85	84	70	36	88	50	63	45	40	100	-19
meanRadius	32	34	34	13	23	47	37	9	-8	100	40	-14
Density	60	57	57	76	-8	37		28	100	-8	45	11
nLayer	58	59	57	48	54	62	43	100	28	9	63	23
nTrack	38	40	38	23	84	51	100	43		37	50	-7
nCluster	94	95	94	75	39	100	51	62	37	47	88	-12
nHough	26	28	26	13	100	39	84	54	-8	23	36	6
nHit3	91	89	89	100	13	75	23	48	76	13	70	-4
nHit2	99	98	100	89	26	94	38	57	57	34	84	-8
nHit1	99	100	98	89	28	95	40	59	57	34	85	-9
nHit	100	99	99	91	26	94	38	58	60	32	84	-8

3） 4a 为能量重建 test 的结果。纵轴为重建能量-真实能量，横轴为真实能量。

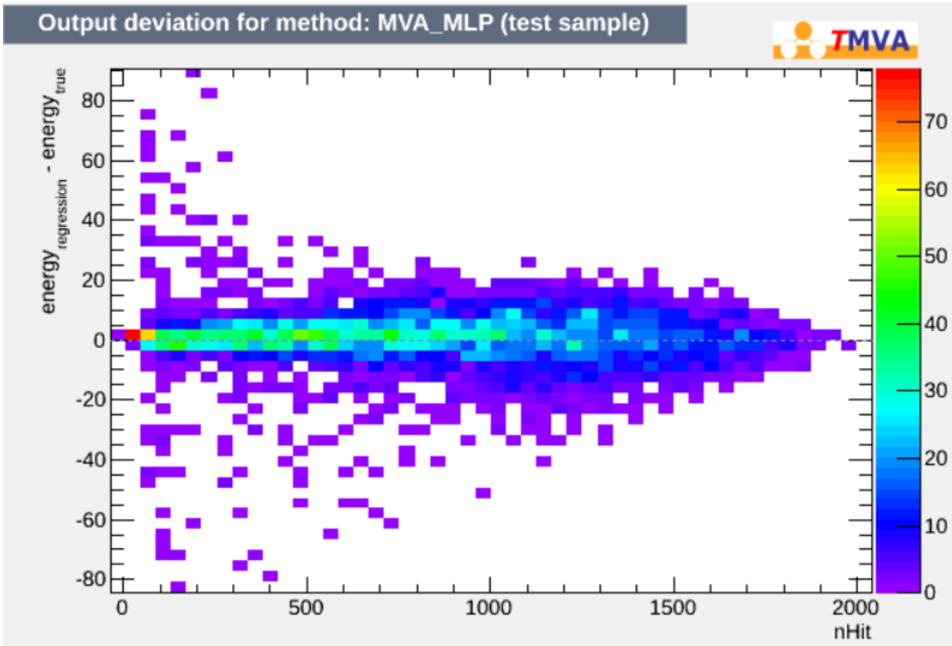


（MLP 方法 test 结果）

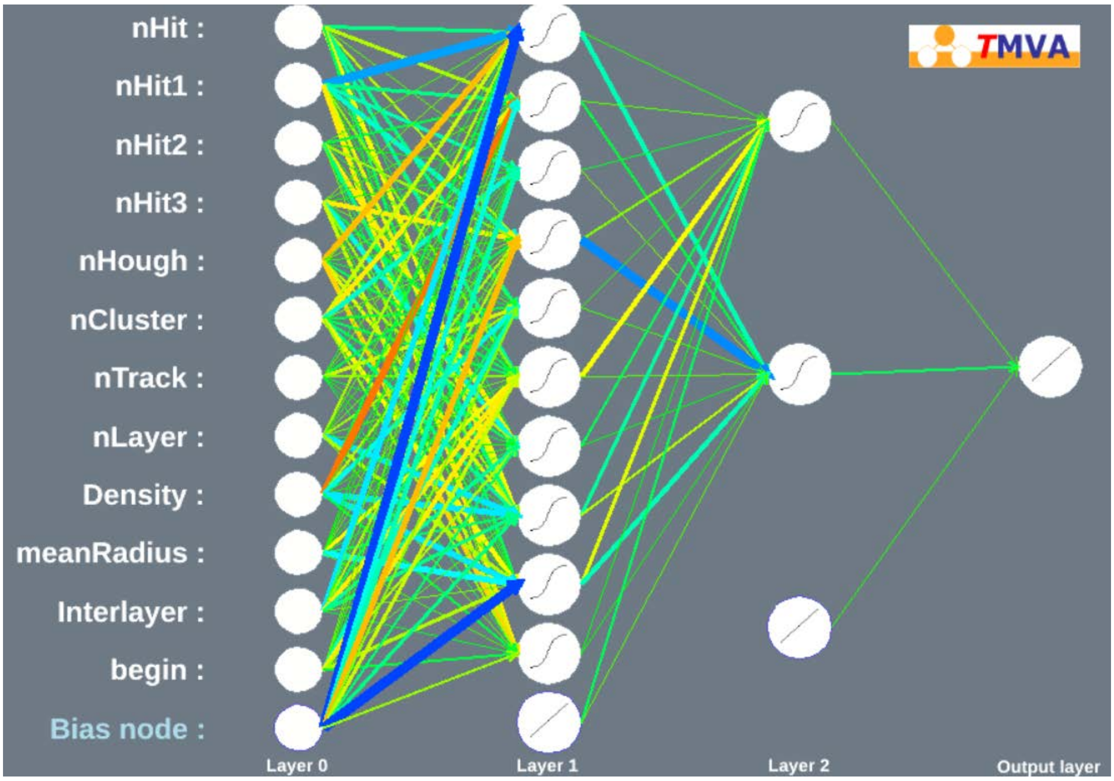


（BDTG 方法 test 结果）

4） 4c 也是能量重建 test 的结果，但是横轴更换为各个不同的变量，你可以通过 4c 展示 fig 中的分布判断不同变量对能量重建的影响。例如：nHit 对能量重建的影响，如图所示：



5） 6a 为 MLP 的结构图，你可以在步骤 8 中调整这个结构。

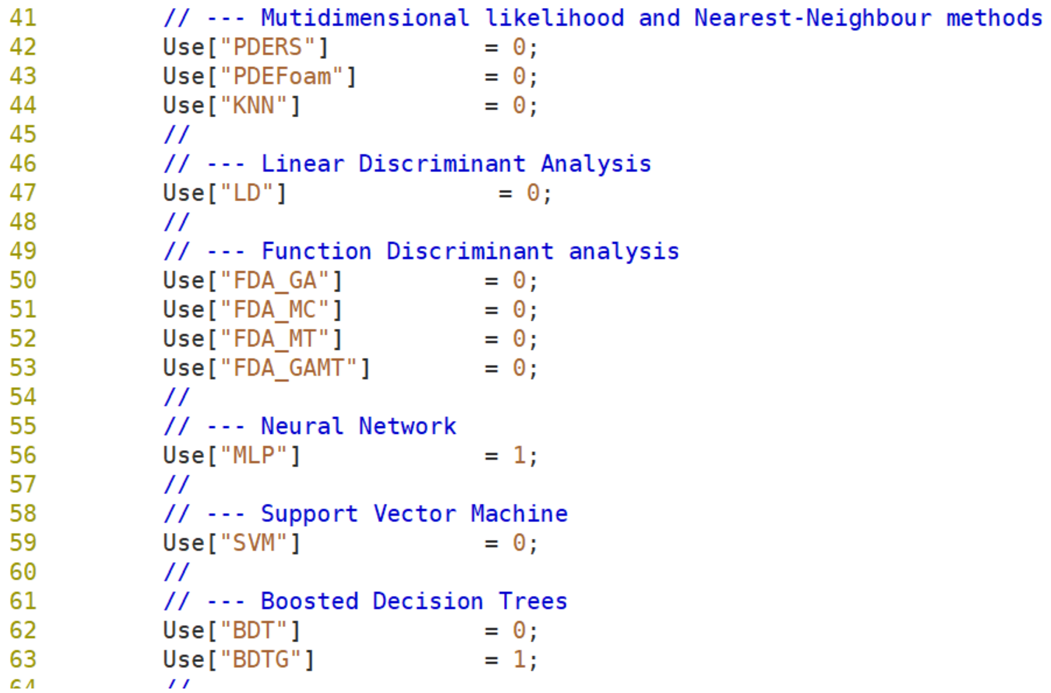


Step2 Application

1. 首先进入/Erec_tuto/tmva 路径下的 TMVARegressionApplication.C

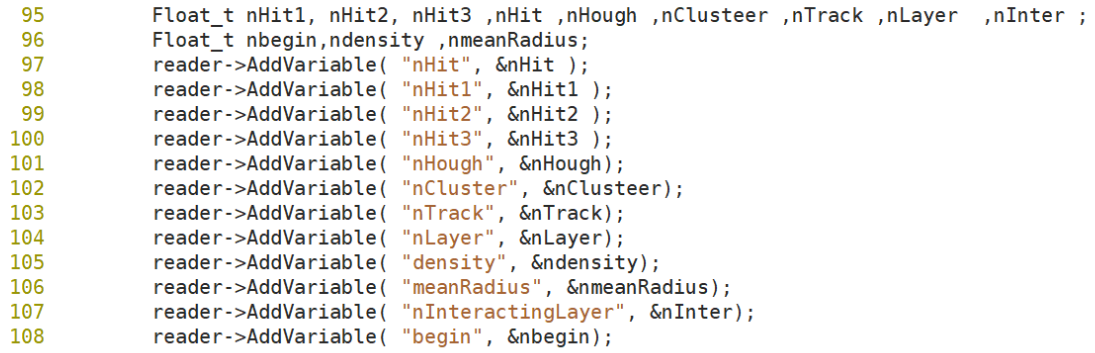
```
cd /Erec_tuto/tmva
vi TMVARegressionApplication.C
```

2. 设定 MVA 中所使用的方法。1 表示打开，0 表示关闭。本次所使用的方法为 MLP 和 BDTG 所以请确保 MLP 和 BDTG 方法打开。将程序调整为下图所示。



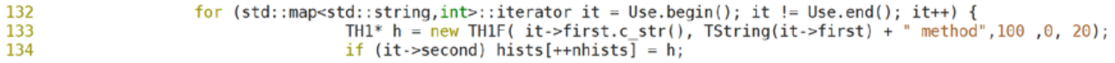
```
        // --- Mutidimensional likelihood and Nearest-Neighbour methods
        Use["PDERS"]           = 0;
        Use["PDEFoam"]         = 0;
        Use["KNN"]             = 0;
        //
        // --- Linear Discriminant Analysis
        Use["LD"]               = 0;
        //
        // --- Function Discriminant analysis
        Use["FDA_GA"]          = 0;
        Use["FDA_MC"]          = 0;
        Use["FDA_MT"]          = 0;
        Use["FDA_GAMT"]        = 0;
        //
        // --- Neural Network
        Use["MLP"]             = 1;
        //
        // --- Support Vector Machine
        Use["SVM"]             = 0;
        //
        // --- Boosted Decision Trees
        Use["BDT"]             = 0;
        Use["BDTG"]            = 1;
        //
```

3. 添加变量，这些变量**必须**和 TMVARegression.C 中的变量相同



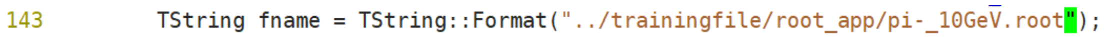
```
        Float_t nHit1, nHit2, nHit3 ,nHit ,nHough ,nClusteer ,nTrack ,nLayer  ,nInter ;
        Float_t nbegin,ndensity ,nmeanRadius;
        reader->AddVariable( "nHit", &nHit );
        reader->AddVariable( "nHit1", &nHit1 );
        reader->AddVariable( "nHit2", &nHit2 );
        reader->AddVariable( "nHit3", &nHit3 );
        reader->AddVariable( "nHough", &nHough);
        reader->AddVariable( "nCluster", &nClusteer);
        reader->AddVariable( "nTrack", &nTrack);
        reader->AddVariable( "nLayer", &nLayer);
        reader->AddVariable( "density", &ndensity);
        reader->AddVariable( "meanRadius", &nmeanRadius);
        reader->AddVariable( "nInteractingLayer", &nInter);
        reader->AddVariable( "begin", &nbegin);
```

4. 设置重建结束之后输出重建能量分布的参数。其中 100,0,20 分别表示 bin 的数量，histgram 的能量范围，我们将其设定为 0,20。当你重建的能量在此区域之外时可以适当调整该值。



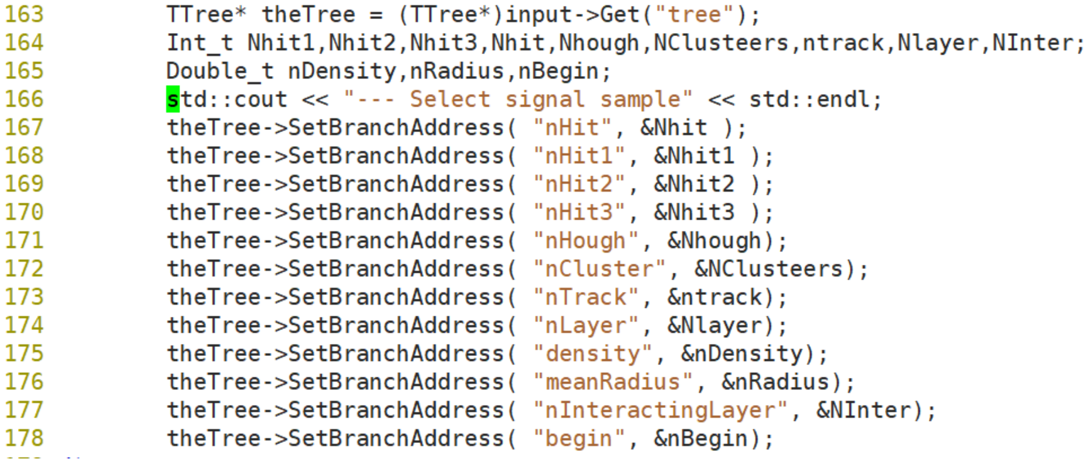
```
        for (std::map<std::string,int>::iterator it = Use.begin(); it != Use.end(); it++) {
                TH1* h = new TH1F( it->first.c_str(), TString(it->first) + " method",100 ,0, 20);
                if (it->second) hists[++nhists] = h;
```

5. 添加需要重建能量的 MC/data samples 的路径，我们在路径 Erec_tuto/trainingfile/root_app 下为你准备了 8 个 pi-的 MC root 文件，这些 root 文件的区别在于它们拥有不同的分立的能量。分别是 10,20,30……80GeV，你可以修改下面这一行代码，尝试对不同能量的 root 文件进行能量重建。



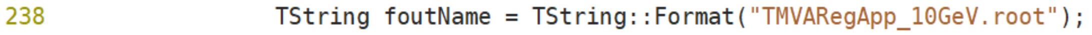
```
        TString fname = TString::Format("../trainingfile/root_app/pi-_10GeV.root");
```

6. 读取 root 中的 tree 和 variables，每一个 variable 都需要与之前所填写的一一对应。



```
        TTree* theTree = (TTree*)input->Get("tree");
        Int_t Nhit1,Nhit2,Nhit3,Nhit,Nhough,NClusteers,ntrack,Nlayer,NInter;
        Double_t nDensity,nRadius,nBegin;
        std::cout << "--- Select signal sample" << std::endl;
        theTree->SetBranchAddress( "nHit", &Nhit );
        theTree->SetBranchAddress( "nHit1", &Nhit1 );
        theTree->SetBranchAddress( "nHit2", &Nhit2 );
        theTree->SetBranchAddress( "nHit3", &Nhit3 );
        theTree->SetBranchAddress( "nHough", &Nhough);
        theTree->SetBranchAddress( "nCluster", &NClusteers);
        theTree->SetBranchAddress( "nTrack", &ntrack);
        theTree->SetBranchAddress( "nLayer", &Nlayer);
        theTree->SetBranchAddress( "density", &nDensity);
        theTree->SetBranchAddress( "meanRadius", &nRadius);
        theTree->SetBranchAddress( "nInteractingLayer", &NInter);
        theTree->SetBranchAddress( "begin", &nBegin);
```

7. 定义一个新的 root 文件，并将重建能量的分布存储到这个 root 文件中。



```
                TString foutName = TString::Format("TMVARegApp_10GeV.root");
```

8. 退出文件，并运行该文件即可用之前 training 过的软件包对新的数据进行能量重建。

```
root TMVARegressionApplication.C
```

9. 运行完第 7 步之后会产生一个新的 tmva 目录下产生一个新的 root 文件，也就是第 6 步中所定义的“TMVARegApp_10GeV.root”。

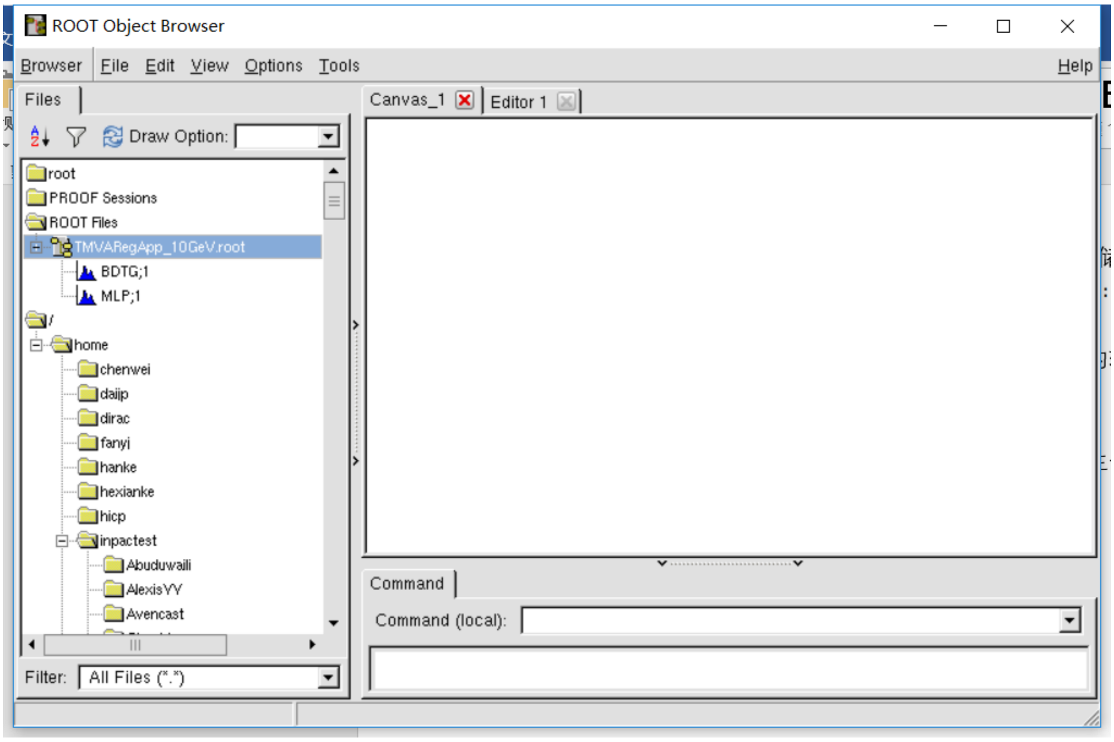
打开这个 root 文件。

```
root TMVARegressionApp_10GeV.C
```

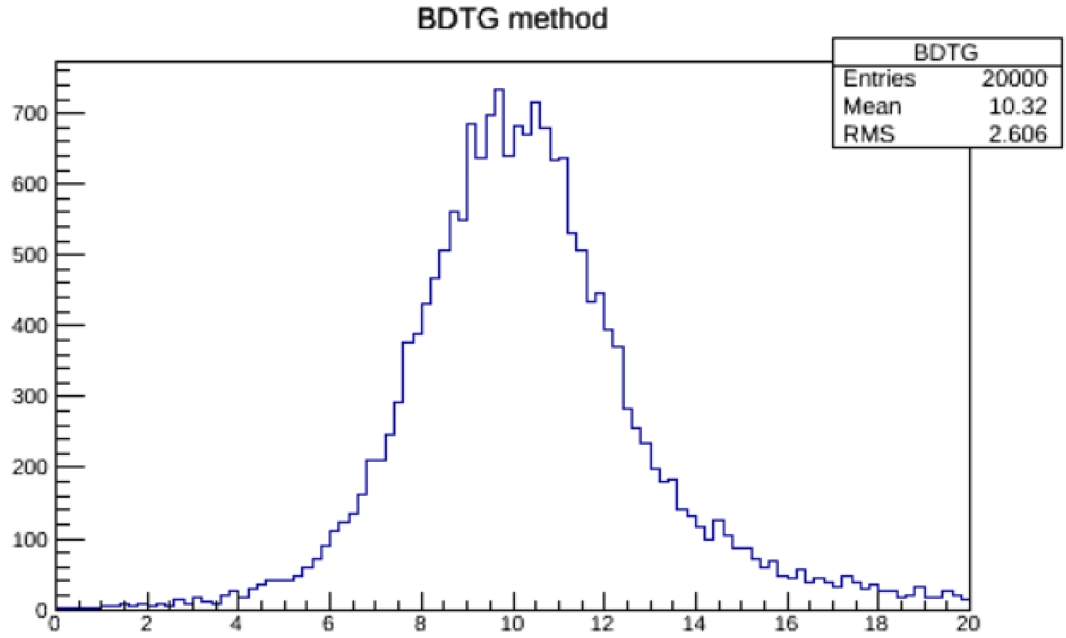
输入

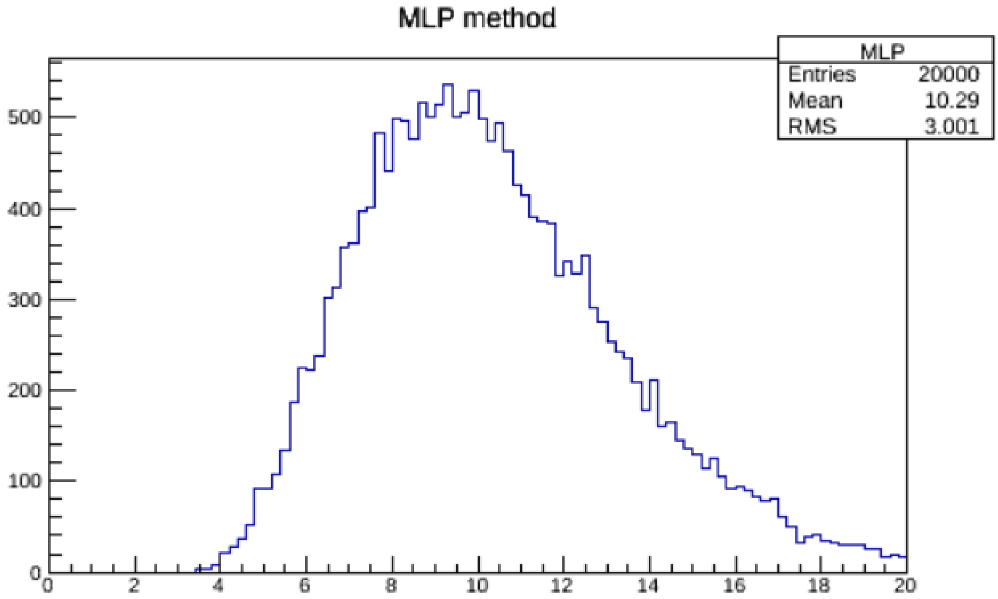
```
root [1] TBrowser a
```

会进入浏览界面



双击“BDTG；1”和“MLP；1”两个图标即可得到能量重建的结果。


BDTG method
BDTG
Entries 20000
Mean 10.32
RMS 2.606


MLP method
MLP
Entries 20000
Mean 10.29
RMS 3.001