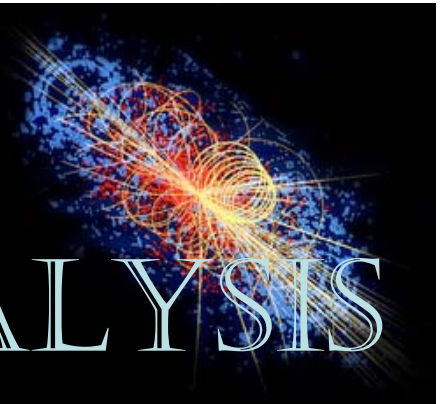


MULTIVARIATE ANALYSIS TUTORIAL

Using neural networks in ROOT
(and other available tools)

Adrian Bevan

YETI January 2007

Uses ROOT 5.12.00

Queen Mary
University of London

OVERVIEW

- 3 tutorials over the next two days:
  - Introduction:
    - Introduction to ROOT.
  - Multi Variate Analysis:
    - Training Neural Networks
    - Tools to calculate fisher discriminants, train neural networks and boosted decision trees.
  - Fitting:

    Fitting in ROOT (1): writing your own PDF to fit to.

    (2): Using TMinuit

    (3): RooFit

Aims of this Tutorial

- Learn how to train an artificial neural network (ANN) using TMultiLayerPerceptron within ROOT

- Be aware of some of the alternative tools that are available within ROOT (e.g. TMVA [Toolkit for Multivariate analysis] and RooNNO).
  - Note that NNO [Neural Network Objects] is also available as a stand alone tool which is independent of ROOT.

TMultiLayerPerceptron

- Need to define inputs
  - What data source for signal and background?
  - What variables do you want to train?
    - What variables are the most important contributions to the final ANN output?
  - What configuration do you want for the Multilayer Perceptron (MLP)?
    - How many nodes to input
    - How many nodes in the hidden layer(s)
    - How many nodes to output
- How do you know if you have the best signal/background discrimination from your chosen configuration?
- What can you do next?

See Chapter 5 of the ROOT 5.12 User Guide for more details

EXAMPLE: MLP in $B^0 \to a_1 \rho$ Search

- Large background from $e^+e^- \to qq$ (q=u,d,s,c,) events.
- Small signal (not observed yet, BR ~10^{-5} - 10^{-6}).
- In $e^+e^- \to \Upsilon(4S) \to BB$ events we can use energy flow variables calculated in the centre of mass of the other B-meson in the event to discriminate between BB events and qq background.
  - Typically use GEANT 4 based MC (with appropriate calibrations from data if necessary) for signal training sample.
  - data accumulated below the $\Upsilon(4S)$ resonance for the background training sample.

EXAMPLE: MLP in $B^0 \to a_1\rho$ Search

- What variables are usable?
  - Here is a list of some of the variables one can use:

VARIABLE DEFINITIONS

VARIABLE NAMES

- The Legendre monomials, L_0, L_2, where these are split into sums over the rest of the event for neutral and charged particles; $L_{0,n}, L_{2,n}$ and $L_{0,c}, L_{2,c}$. These monomials were the result of an investigation with respect to improving the Fisher discriminant used in the analysis of h^+h^-. The full documentation of this study can be found in [26]. The definition of the monomials is

$$L_0 = \sum_{ROE} p_i \quad (8)$$

$$L_2 = \sum_{ROE} p_i |cos(\theta_i)|^2 \quad (9)$$

where the sum is over the rest of the event, p_i is the particle momentum and θ_i is the angle of the particle direction relative to the thrust axis of the B candidate. The previous iteration of this analysis included a Fisher discriminant resulting from these four input variables.

L0
L2
lgdr0P1c
lgdr0P1n
lgdr2P1c
lgdr2P1n

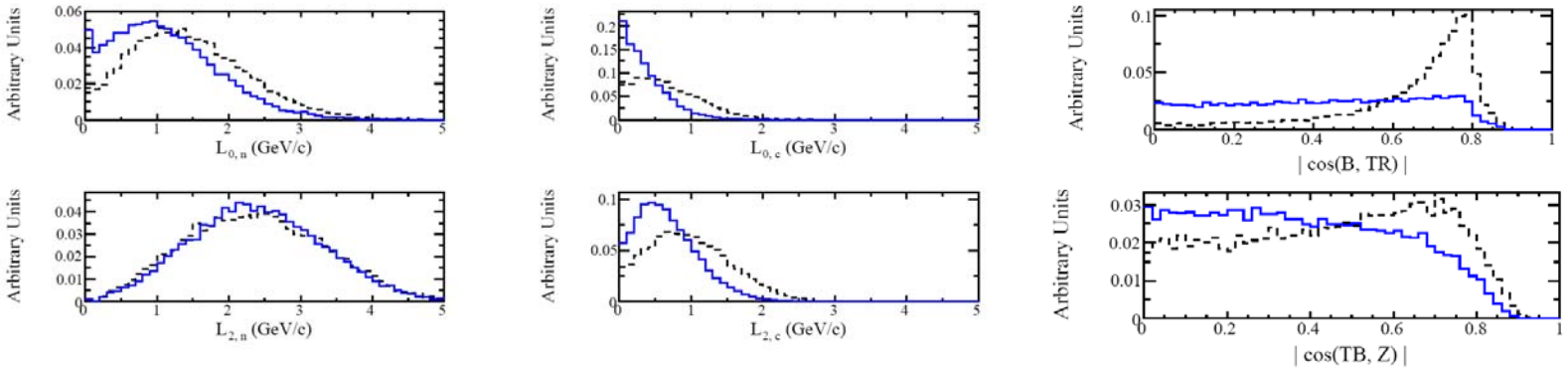
- $|cos(B,TR)|$, the absolute value of the cosine of the angle between the the B thrust and the thrust of the rest of the event. This variable is strongly peaked at unity for $q\overline{q}$ events. $B\overline{B}$ events are more isotropic as the B mesons are produced close to the kinematic threshold.
- $|cos(TB,Z)|$, the absolute value of the cosine of the angle between the B thrust and the z axis.

cosThetaT

cosBthr

EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

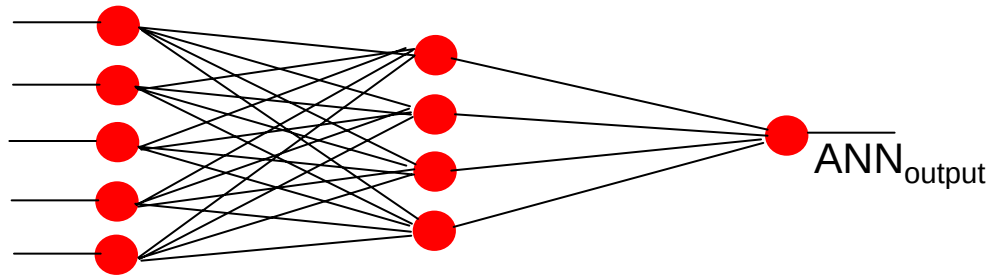
- Variable distributions for signal (solid) and background (dashed)



- Correlation coefficients between the variables show that while they are correlated, there is addition information when adding them all.
- Try a simple MLP configuration:

N : N-1 : 1

where the hidden layer always has two or more nodes.



EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- The macro `mlpTest.cc` is set up in order to use the `signal.root` and `background.root` data files.
  - Need to define a training sample
    - A TTree with equal amounts of signal (type=1), and background (type=0).
  - Need to select a network configuration
    - List the variables to use in the network, and then the MLP configuration. e.g. for L0 and L2 we want

      `L0,L2:2:2:1`
  - Need to specify how to select the events to use for training, and the events to use for testing the training.
    - Want to check that we don't tune our ANN to statistical fluctuations in the training sample.
    - Usually take alternate events for training/testing

EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

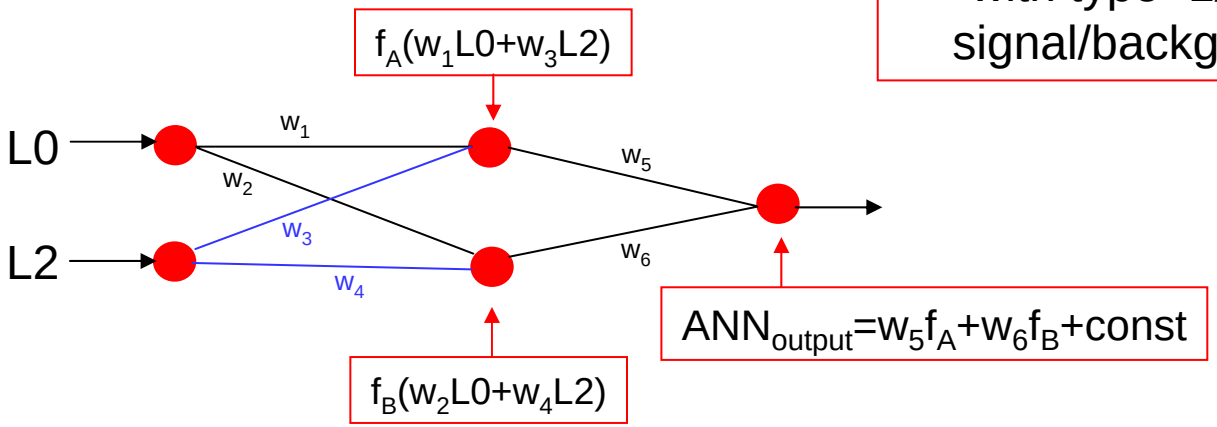
- Make a TMultiLayerPerceptron object

TMultiLayerPerceptron mlp("L0,L2:2:2:1", TrainingSample, "Entry$%2", "Entry$/2");

A comma separated list of variables to Use in the ANN, followed by a colon Separated list of the number of input, hidden and output nodes.

A TTree training sample with type=1/0 for signal/background

Selection rules to talk alternate events to train and to test.



EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- Make a TMultiLayerPerceptron object

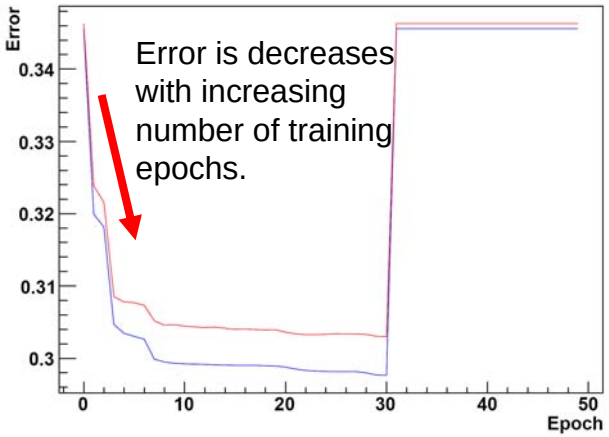
TMultiLayerPerceptron mlp("L0,L2:2:2:1", TrainingSample, "Entry$%2", "Entry$/2");

- Train the MLP

mlp.Train(50, "text,graph,update=5");

The number of training cycles to run (how many loops through the data defined by the TTree TrainingSample)

Every 5 cycles update information on the progress of the training. When the training is done the error on the type selection will become constant.



The error for signal (red) and background (blue) identification for the first 30 epochs of training.

EXAMPLE: MLP in $B^0 \to a_1 \rho$ Search

- Make a TMultiLayerPerceptron object

```
TMultiLayerPerceptron mlp("L0,L2:2:2:1", TrainingSample, "Entry$%2", "Entry$/2");
```

- Train the MLP

```
mlp.Train(50, "text,graph,update=5");
```

- Look at the result of the training

```
TMLPAnalyzer ana(mlp);
ana.GatherInformations();
ana.CheckNetwork();
ana.DrawDInputs()
mlp.Draw()
etc.
```

Show the relative weight of variables for the MLP after training

Draw a schematic of the MLP showing all input, hidden and output nodes.

EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- Make a TMultiLayerPerceptron object

```
TMultiLayerPerceptron mlp("L0,L2:2:2:1", TrainingSample, "Entry$%2", "Entry$/2");
```

- Train the MLP

```
mlp.Train(50, "text,graph,update=5");
```

- Look at the result of the training

```
TMLPAnalyzer ana(mlp);
ana.GatherInformations();
ana.CheckNetwork();
ana.DrawDInputs()
mlp.Draw()
etc.
```

- Is the output sensible?

A lot of the time common sense is sufficient to tell you if there is a problem with the training or not.

EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- Make a TMultiLayerPerceptron object

```
TMultiLayerPerceptron mlp("L0,L2:2:2:1", TrainingSample, "Entry$%2", "Entry$/2");
```

- Train the MLP

```
mlp.Train(50, "text,graph,update=5");
```

- Look at the result of the training

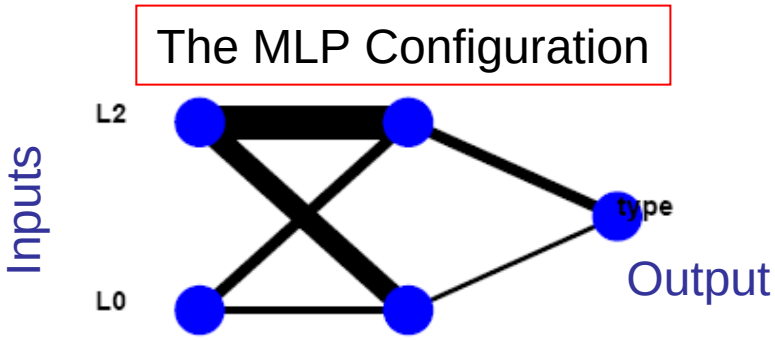
```
TMLPAnalyzer ana(mlp);
ana.GatherInformations();
ana.CheckNetwork();
ana.DrawDInputs()
mlp.Draw()
etc.
```

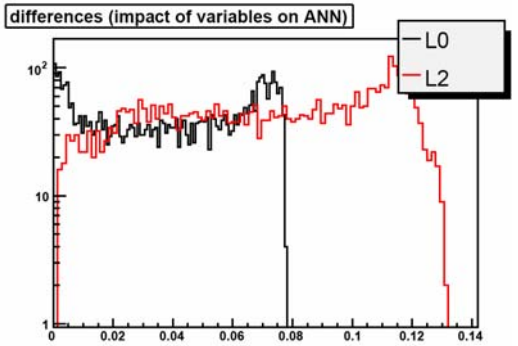
- Is the output sensible?
- Write out code to calculate ANN_{output} for you

```
mlp.Export("myCode");
```

EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- What are the strongest contributions to the ANN output: Consider using L0 and L2 only:
  - Both outputs of the TMultiLayerPerceptron indicate that L2 is a more powerful discriminator than L0.

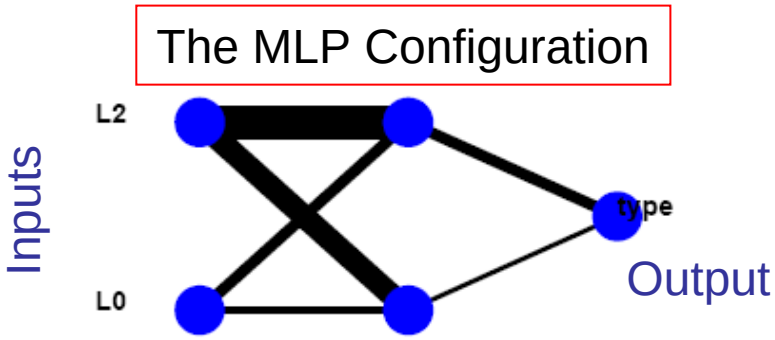


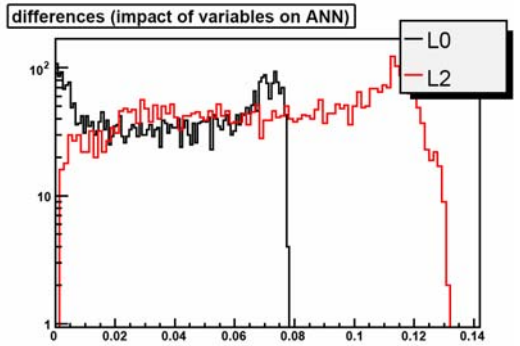


Relative weights in the ANN

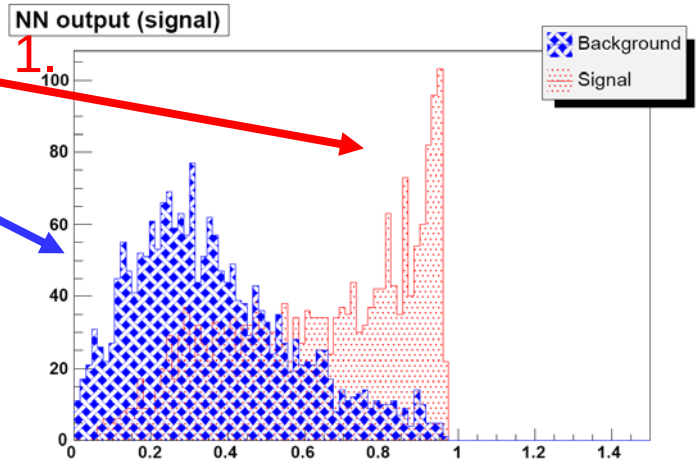
EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- What are the strongest contributions to the ANN output: Consider using L0 and L2 only:
  - Both outputs of the TMultiLayerPerceptron indicate that L2 is a more powerful discriminator than L0.



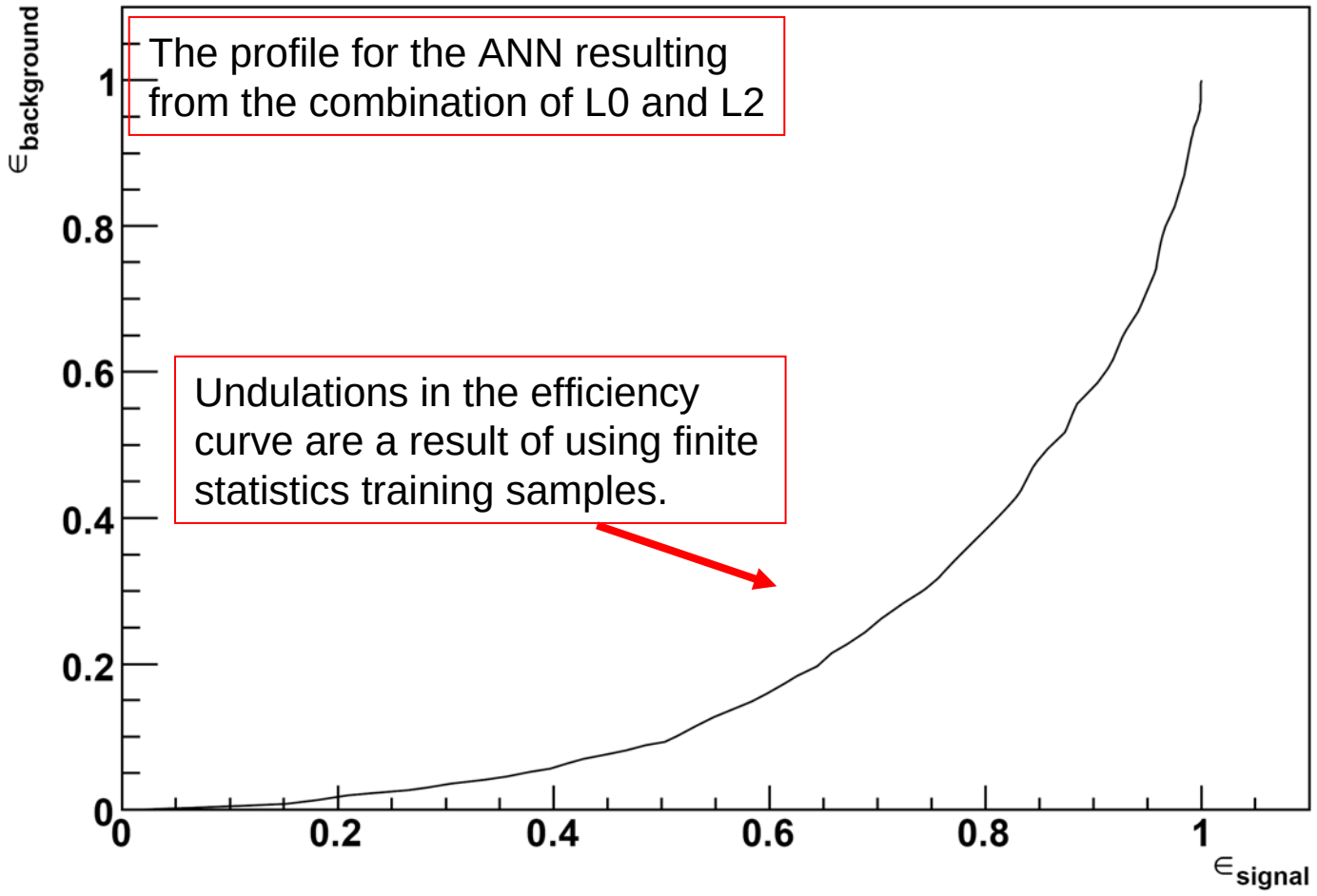


- What does the ANN output distribution look like?
  - The signal distribution peaks toward 1.
  - Background peaks toward 0.



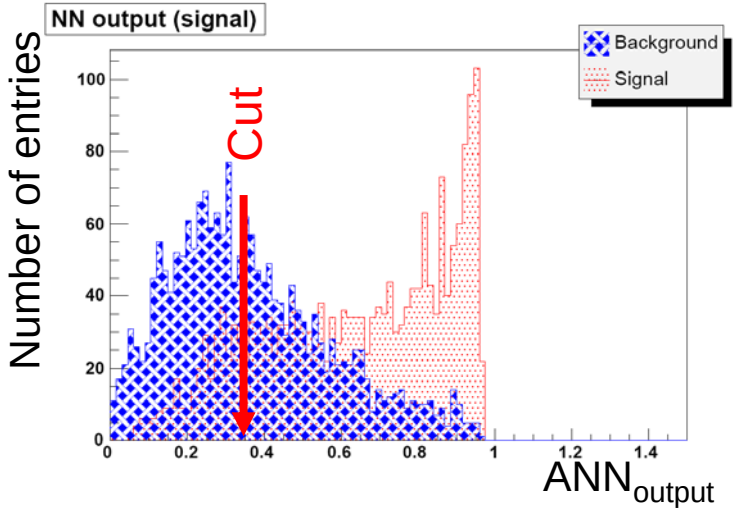
EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

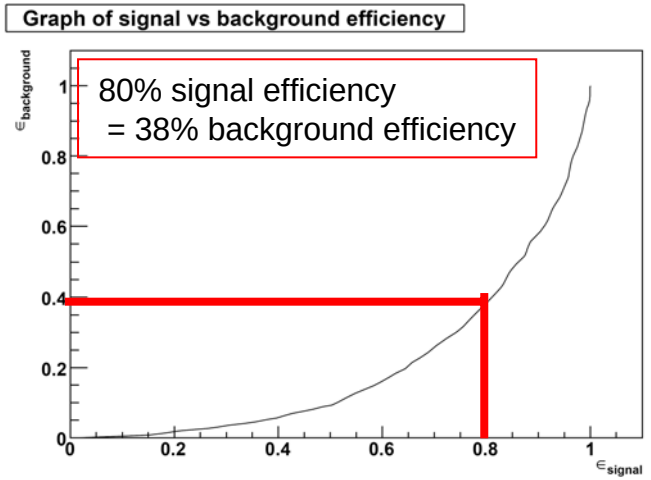
- If we have a choice of ANNs to use in an analysis. How do we compare them?
  - Use the efficiency curve of signal vs. background



EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- How do we use the ANN now that we have decided on our optimal configuration?
  - Can cut on the ANN output (loose if you want to fit to the data, optimally if you have a cut based analysis).

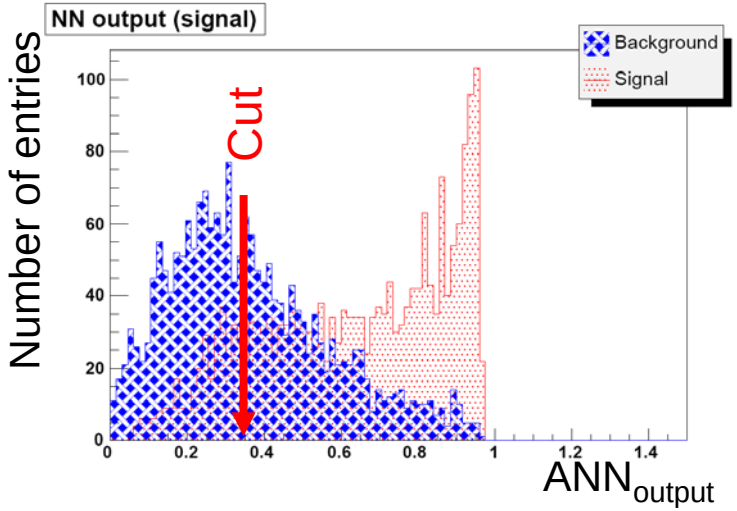


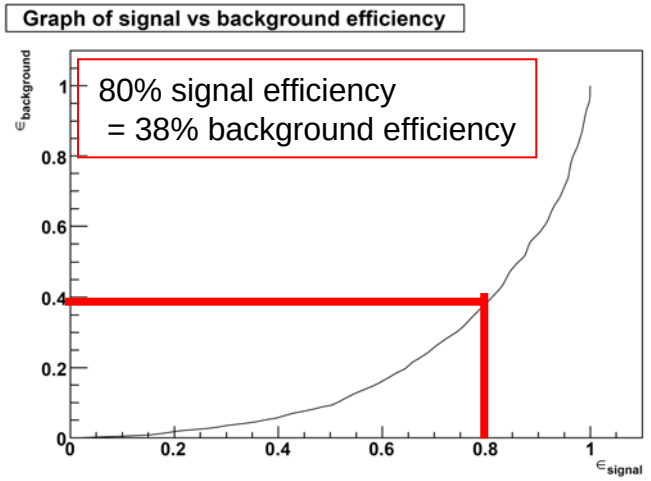


Why might we want to cut on ANN_{output} if we are fitting data?
→ Reduce the number of events going into the fit
→ Simplify the fit (esp. if this is multi-dimensional)

EXAMPLE: MLP in $B^0 \rightarrow a_1 \rho$ Search

- How do we use the ANN now that we have decided on our optimal configuration?
  - Can cut on the ANN output (loose if you want to fit to the data, optimally if you have a cut based analysis).





  - Include the ANN output in a fit.
    - We might like to consider transforming the output to parameterise the ANN shape more easily (or just use a non-parametric PDF; i.e. a Histogram to do this job for us).

EXERCISES: MLP in $B^0 \rightarrow a_1 \rho$ Search

1. Run the macro `mlpTest.cc` to train different ANNs
   - L0+L2
   - L0+L2+cosThetaT
   - L0+L2+cosThetaT+cosBthr
   - To run the macro enter the following commands into ROOT

```
gSystem->Load("libMLP.so")
.x mlpTest.cc
```

- What is the strongest contribution to each of the networks?

EXERCISES: MLP in $B^0 \rightarrow a_1 \rho$ Search

1. Run the macro `mlpTest.cc` to train different ANNs
   - L0+L2
   - L0+L2+cosThetaT
   - L0+L2+cosThetaT+cosBthr
   - To run the macro enter the following commands into ROOT

```
gSystem->Load("libMLP.so")
.x mlpTest.cc
```

- What is the strongest contribution to each of the networks?
- What is the best network configuration?

EXERCISES: MLP in $B^0 \rightarrow a_1 \rho$ Search

1. Run the macro `mlpTest.cc` to train different ANNs
   - L0+L2
   - L0+L2+cosThetaT
   - L0+L2+cosThetaT+cosBthr
   - To run the macro enter the following commands into ROOT

```
gSystem->Load("libMLP.so")
.x mlpTest.cc
```

- What is the strongest contribution to each of the networks?
- What is the best network configuration?
  - Hint: open the files nn/*.root retrieve and overlay the TGraphs called efficiency_graph.

```
TGraph * a = (TGraph*)fa.Get("efficiency_graph");
 … etc. …
a->Draw("Ac");
b->SetLineColor(kRed); b->Draw("same");
C->SetLineColor(kGreen); c->Draw("same");
```

Other MVA Tools

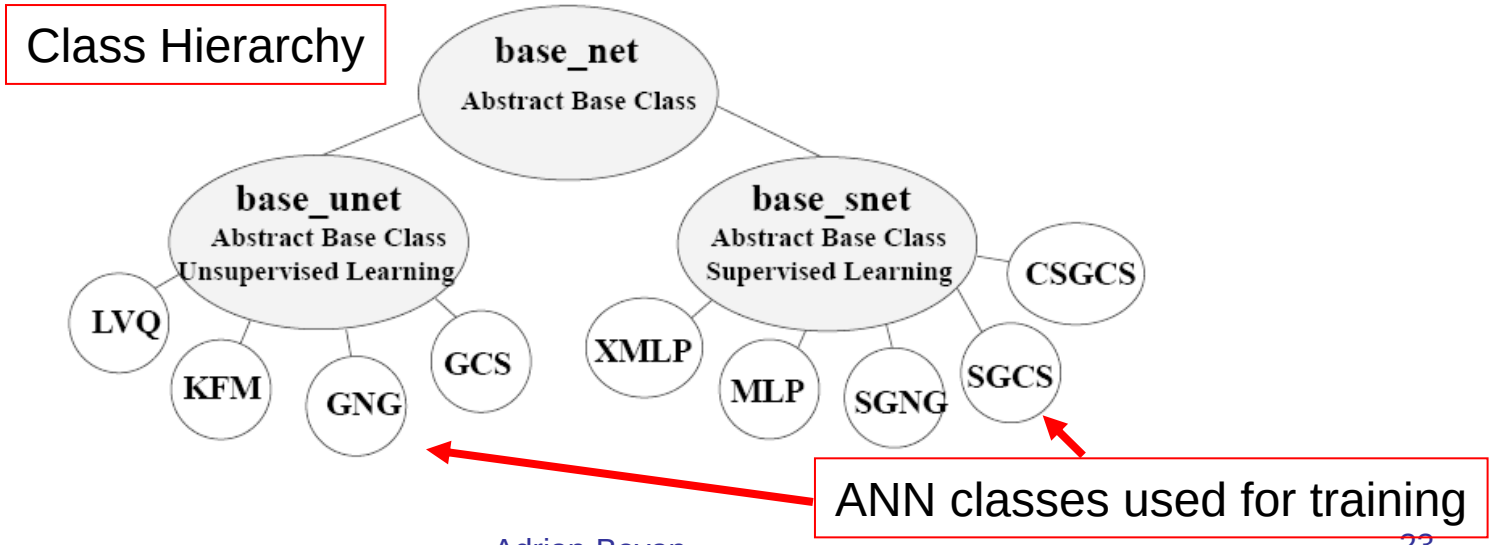
- There are other MVA tools available, and not all of them require ROOT. Some of these are:
  - Neural Network Objects: RooNNO (NNO)
    - This is an object orientated set of classes that can be used to train an ANN. Versions of NNO exist within ROOT and as a standalone set of code.
  - Toolkit for Multivariate Analysis: TMVA
    - This toolkit is available within ROOT, and it can be used to calculate fisher discriminants, train ANN and boosted decision trees.

RooNNO / NNO

Root based / stand alone

- C++ based artificial neural network training algorithm.
- Includes several different types of ANN algorithms.
  - The simplest class is the Multi-Layer Perceptron (MLP).
  - Also has more sophisticated models for training.

See website for more details
http://www.ep1.ruhr-uni-bochum.de/~marcel/nno.html
The proceedings of the 5th AIHENP Workshop, Lausanne (1996) gives more detail.



TMVA

- Various multivariate analysis tools:
  - Cut optimisation
  - Likelihood estimator
  - H-matrix
  - Fisher
  - ANN
  - Decision trees
  - Rule Fit

Concentrate on this

See website for more details
http://tmva.sourceforge.net/

- Pros:
  - Many different algorithms available to compare etc. Available in ROOT (since v5.11/03).
- Cons:
  - No User guide yet (in progress – due by the end of January 2007), still some development in progress (Genetic Algorithms).

Example: Fisher Discriminant

- Is using an ANN overkill for my problem?
- Do you feel like the ANN is a black box and want a more transparent discriminant?
  - You can use a Fisher discriminant to check the signal vs. background efficiency performance against a NN.

$$F = \sum_i w_i x_i + C$$

This is a linear weighting of the input variables, with a single output variable.

Example: Fisher Discriminant

- Easy to set up using TMVA: see `tmva_fisher.cc`
  - Start from `tmva/test/TMVAnalysis.C` & modify

```
gROOT->SetStyle("Plain");
gSystem->Load("libMLP");
gSystem->Load("libTMVA.so");

// Set up the output file and TMVA toolkit
TFile f("fisher.root", "RECREATE");
TMVA::Factory * factory = new TMVA::Factory("TMVAanalysis", &f, "");
```

Load MLP and TMVA libraries

Make a TMVA Factory

A name for your analysis

An output file to write to

Example: Fisher Discriminant

- Easy to set up using TMVA: see `tmva_fisher.cc`
  - Start from `tmva/test/TMVAnalysis.C` & modify

```
gROOT->SetStyle("Plain");
gSystem->Load("libMLP");
gSystem->Load("libTMVA.so");

// Set up the output file and TMVA toolkit
TFile f("fisher.root", "RECREATE");
TMVA::Factory * factory = new TMVA::Factory("TMVAanalysis", &f, "");

// Get data to use in calculating the MVA
TFile fsignal("signal.root");
TFile fbg("background.root");
TTree * tsignal = (TTree*)fsignal.Get("data");
TTree * tbg     = (TTree*)fbg.Get("data");
```

Load MLP and TMVA libraries

Make a TMVA Factory

Get the signal and background Training samples ($B^0 \rightarrow a_1 \rho$ again)

Example: Fisher Discriminant

- Easy to set up using TMVA: see `tmva_fisher.cc`
  - Start from `tmva/test/TMVAnalysis.C` & modify

```
gROOT->SetStyle("Plain");
gSystem->Load("libMLP");
gSystem->Load("libTMVA.so");

// Set up the output file and TMVA toolkit
TFile f("fisher.root", "RECREATE");
TMVA::Factory * factory = new TMVA::Factory("TMVAanalysis", &f, "");

// Get data to use in calculating the MVA
TFile fsignal("signal.root");
TFile fbg("background.root");
TTree * tsignal = (TTree*)fsignal.Get("data");
TTree * tbg     = (TTree*)fbg.Get("data");

cout << "have signal tree     "<< tsignal << endl;
cout << "have background tree "<< tbg << endl;

// Set up the input data trees to use with TMVA
factory->SetInputTrees( tsignal, tbg);
```

Load MLP and TMVA libraries

Make a TMVA Factory

Get the signal and background Training samples ($B^0 \rightarrow a_1 \rho$ again)

Set the signal and background trees to use in the MVA

Example: Fisher Discriminant

- Easy to set up using TMVA: see `tmva_fisher.cc`
  - Start from `tmva/test/TMVAnalysis.C` & modify

```
gROOT->SetStyle("Plain");
gSystem->Load("libMLP");
gSystem->Load("libTMVA.so");

// Set up the output file and TMVA toolkit
TFile f("fisher.root", "RECREATE");
TMVA::Factory * factory = new TMVA::Factory("TMVAanalysis", &f, "");

// Get data to use in calculating the MVA
TFile fsignal("signal.root");
TFile fbg("background.root");
TTree * tsignal = (TTree*)fsignal.Get("data");
TTree * tbg     = (TTree*)fbg.Get("data");

cout << "have signal tree     "<< tsignal << endl;
cout << "have background tree "<< tbg << endl;

// Set up the input data trees to use with TMVA
factory->SetInputTrees( tsignal, tbg);

// now define the variables to use in the MVA
vector<TString>* inputVars = new vector<TString>;
inputVars->push_back("lgdr0P1n");
inputVars->push_back("lgdr2P1n");
inputVars->push_back("lgdr0P1c");
inputVars->push_back("lgdr2P1c");
factory->SetInputVariables( inputVars );
```

Load MLP and TMVA libraries

Make a TMVA Factory

Get the signal and background Training samples ($B^0 \rightarrow a_1 \rho$ again)

Set the signal and background trees to use in the MVA

Select the variables to use in the MVA (use same variable names as in the TTree)

Example: Fisher Discriminant

- The rest of the macro sets up the TMVA factory, calculates the fisher & makes the output plots.

```
// Apply additional cuts on the signal and background sample.
// Assumptions on size of training and testing sample:
//    a) equal number of signal and background events is used for training
//    b) any numbers of signal and background events are used for testing
//    c) an explicit syntax can violate a)
// more Documentation with the Factory class
TCut mycut = "";
factory->PrepareTrainingAndTestTree( mycut, 1000, 1000 );


// Book the MVA methods you like to investigate.
//  - here we just select the Fisher method
factory->BookMethod( TMVA::Types::Fisher, "Fisher" );
```

Set the MVA methods to use (just Fisher here) & set training parameters

Example: Fisher Discriminant

- The rest of the macro sets up the TMVA factory, calculates the fisher & makes the output plots.

```
// Apply additional cuts on the signal and background sample.
// Assumptions on size of training and testing sample:
//    a) equal number of signal and background events is used for training
//    b) any numbers of signal and background events are used for testing
//    c) an explicit syntax can violate a)
// more Documentation with the Factory class
TCut mycut = "";
factory->PrepareTrainingAndTestTree( mycut, 1000, 1000 );


// Book the MVA methods you like to investigate.
//  - here we just select the Fisher method
factory->BookMethod( TMVA::Types::Fisher, "Fisher" );

// Train MVAs.
factory->TrainAllMethods();
```

Set the MVA methods to use (just Fisher here) & set training parameters

Run the training

Example: Fisher Discriminant

- The rest of the macro sets up the TMVA factory, calculates the fisher & makes the output plots.

```
// Apply additional cuts on the signal and background sample.
// Assumptions on size of training and testing sample:
//    a) equal number of signal and background events is used for training
//    b) any numbers of signal and background events are used for testing
//    c) an explicit syntax can violate a)
// more Documentation with the Factory class
TCut mycut = "";
factory->PrepareTrainingAndTestTree( mycut, 1000, 1000 );


// Book the MVA methods you like to investigate.
//  - here we just select the Fisher method
factory->BookMethod( TMVA::Types::Fisher, "Fisher" );

// Train MVAs.
factory->TrainAllMethods();


// Test MVAs.
factory->TestAllMethods();


// Evaluate variables.
factory->EvaluateAllVariables();

// Evaluate MVAs
factory->EvaluateAllMethods();

// Save the output.
f.Close();
```

Set the MVA methods to use (just Fisher here) & set training parameters

Run the training

Analyse results (make plots and add to the file f)

Example: Fisher Discriminant

- TMVA tabulates your coefficients when you run it.

```
--- TMVA::MethodFisher: ranked output (top variable is best ranked)
------------------------------------------------------------------
--- Variable   :   Coefficient:   Discr. power:
------------------------------------------------------------------
--- lgdr2P1n   :   -2.667         0.1018
--- lgdr2P1c   :   -2.626         0.0544
--- lgdr0P1n   :   +2.129         0.0372
--- lgdr0P1c   :   +2.085         0.0015
--- (offset)   :   -0.713         0
```

- And the Fisher is easy to calculate and use:

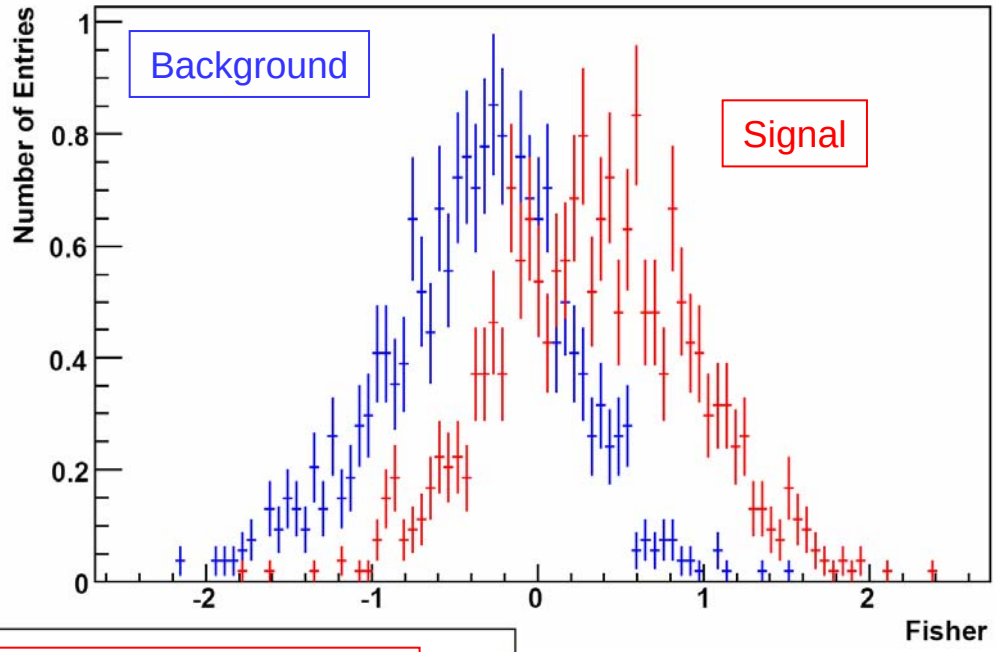
$$F = \sum_i w_i x_i + C$$
$$= 2.129 lgdr0P1n + 2.085 lgdr0P1c$$
$$-2.667 lgdr2P1n - 2.626 lgdr2P1c - 0.713$$

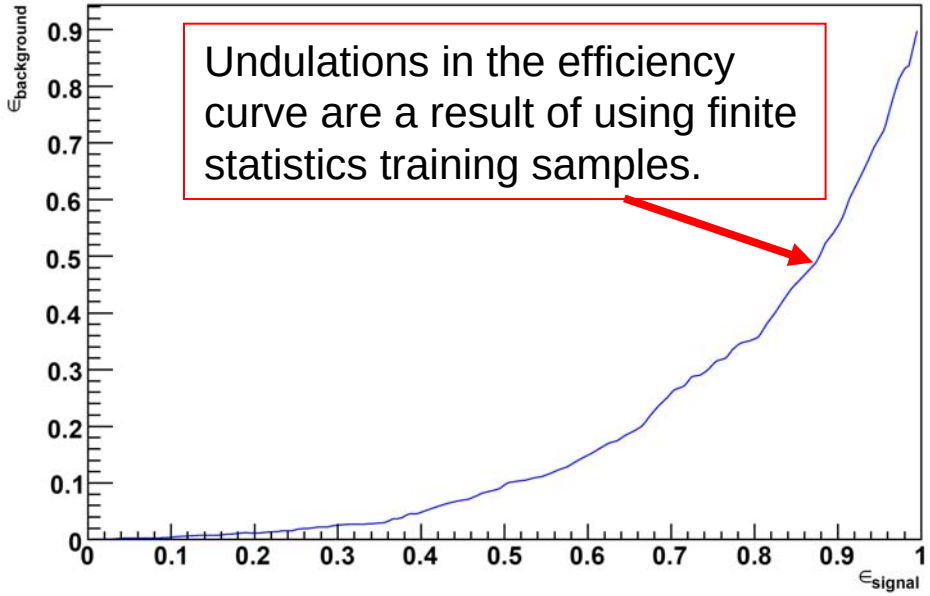
Example: Fisher Discriminant

- Inputs:

  lgdr0P1c, lgdr0P1n,
  lgdr2P1c, lgdr2P1n

- Gaussian-like distribution (easy to parameterise for a fit)





Exercises: Fisher Discriminant

1. Run the macro `tmva_fisher.cc` to produce an output fisher for the four variables previously shown. Look at the root file produced: `fisher.root`

2. Calculate two new Fishers by adding `cosThetaT` and `cosBthr` one at a time.
   - Compare the outputs – what is the best Fisher discriminant?
   - What is the strongest discriminating variable between signal and background

Exercises: Fisher Discriminant

1. Run the macro `tmva_fisher.cc` to produce an output fisher for the four variables previously shown. Look at the root file produced: `fisher.root`

2. Calculate two new Fishers by adding `cosThetaT` and `cosBthr` one at a time.
   - Compare the outputs – what is the best Fisher discriminant?
   - What is the strongest discriminating variable between signal and background

3. Compare the best Fisher discriminant, with your best ANN from earlier. Which is more optimal – the ANN or the Fisher?

Summary

- You have been introduced to
  - examples of using artificial neural networks for developing background fighting techniques.
  - Fisher discriminants as an effective (but usually a little less performant) and simpler alternative to ANNs.
- These techniques become increasingly useful when trying to extract signals from large backgrounds.
- More information can be found at the following URLs

http://root.cern.ch
http://tmva.sourceforge.net/
http://www.ep1.ruhr-uni-bochum.de/~marcel/nno.html