Multivariate Analysis Techniques Lecture 2

BABAR Analysis School
SLAC National Laboratory
25th-30th October 2009

Adrian Bevan (a.j.bevan@qmul.ac.uk)

Lecture 2

- In the next 90 minutes we will cover:
  - Putting the previous lecture into context.
  - Decision Trees
    - Binary,
    - Boosted,
    - and Bagged
  - Forests
  - A Quick Tour of TMVA
    - This is just one of many tools available in the HEP community.

Putting the previous lecture into context.

- In the last lecture we discussed the following algorithms:
  - Cut based
  - Fisher discriminant
  - Neural Networks

- These algorithms have been around for a long time, and naturally they have evolved into useful tools.

- But there are more modern ideas that we should think about when trying to classify our events.
  - Some of these more modern algorithms are the topic of today's lectures.

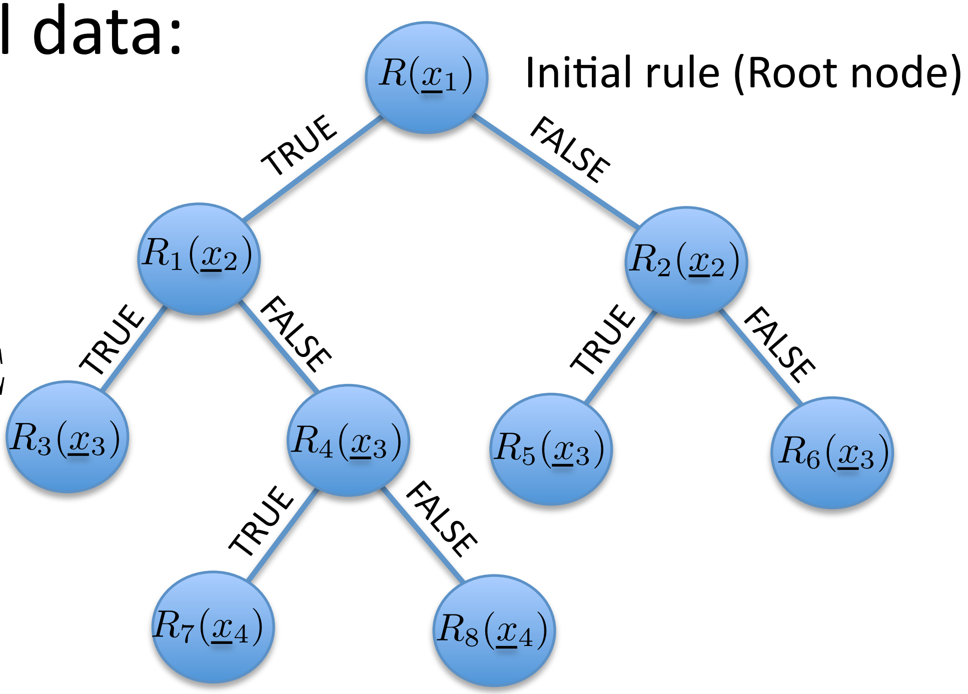
Decision Trees

Decision Trees

- Apply the initial rule to all data:
  - Divide into two classes (with a binary output)

$$R(\underline{x}_1) = \underline{x} > \underline{x}_i \text{ TRUE}$$
$$= \underline{x} < \underline{x}_i \text{ FALSE}$$

  - Each successive layer divides the data further into Signal (class A)/ Background (class B) classifications.

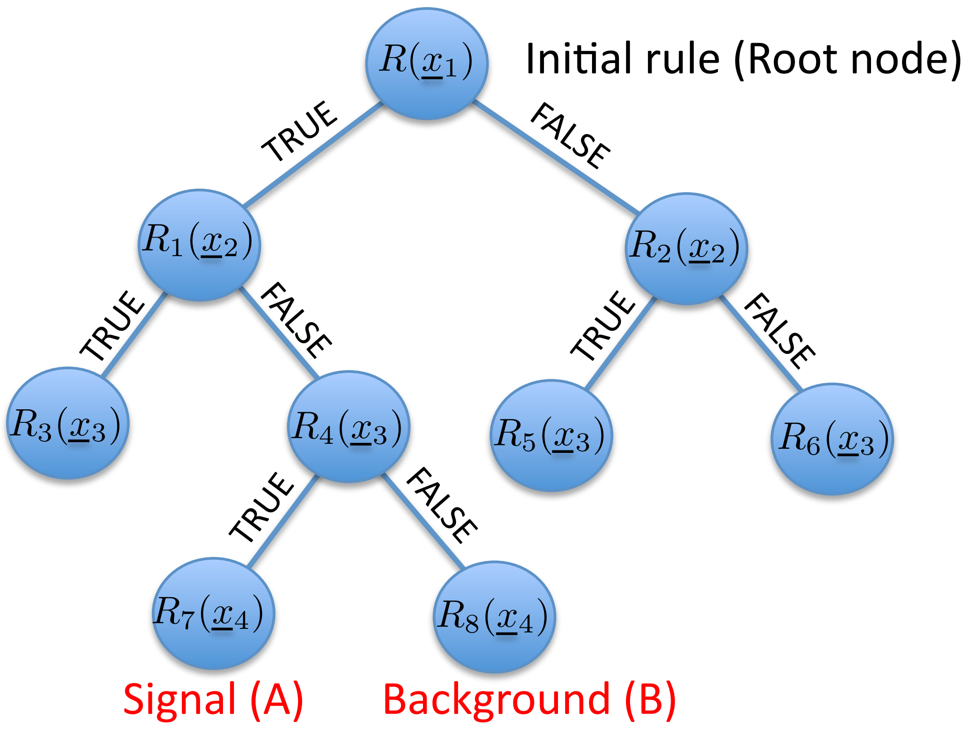


- The classification for a set of cut values will have a classification error.
  - So just as with a NN one can vary the cut values x_i in order to train the tree.

Decision Trees

- What is happening?

- Each nodes uses the sub-set of discriminating variables that give the best separation between classes.

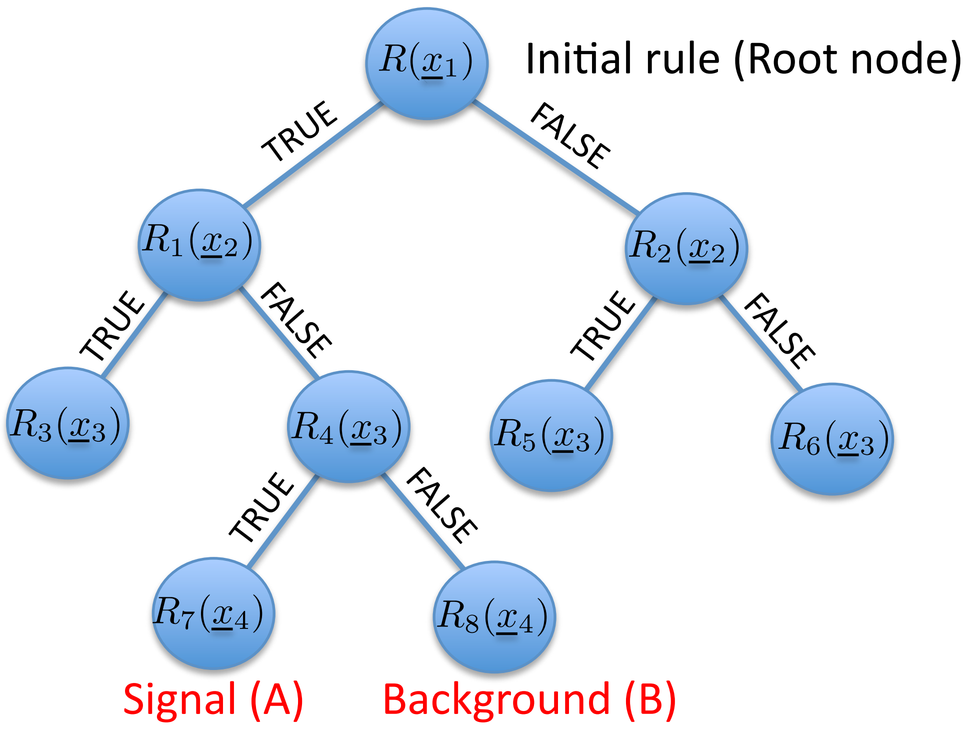


- Some variables may be used by more than one node.

- Other variables may never be used.

Decision Trees

- What is happening?
- The bottom of a tree just looks like a sub-sample of events subjected to a cut based analysis.



Initial rule (Root node)

Signal (A)

Background (B)

- There are many bottom levels to the tree:
  - ... so there are many signal / background regions defined by the algorithm.

Decision Trees

- Binary Decision Tree has the following pros/cons:

- Pros:
  - Easy to understand and interpret.
  - More flexibility in the algorithm when trying to separate classes of events.
    - Able to obtain better separation between classes of events than a simple cut-based approach.

- Cons:
  - Instability with respect to statistical fluctuations in the training sample.

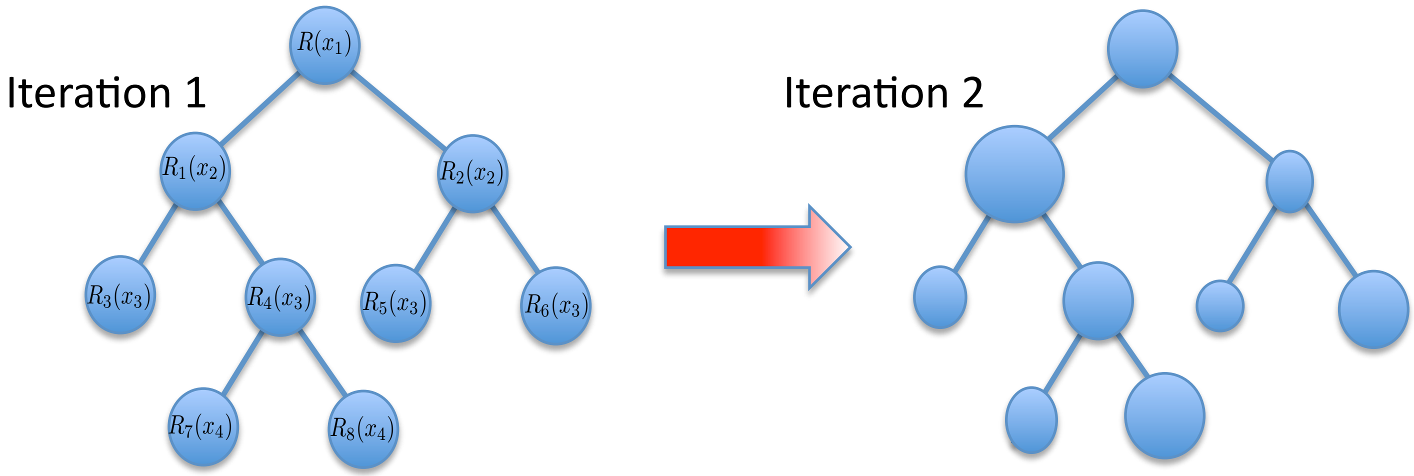
- It is possible to improve upon the binary decision tree algorithm to try and overcome the instability or susceptibility of overtraining.

Boosted Decision Trees

- At each stage in training there may be some mis-classification of events (error rate).
  - Assign a greater event weight α to mis-classified events in the next training iteration.

$$\alpha = \frac{1-\epsilon}{\epsilon} \qquad\qquad \epsilon = \text{error rate}$$

  - Re-weight whole sample so that the sum of weights remains the same, then iterate.



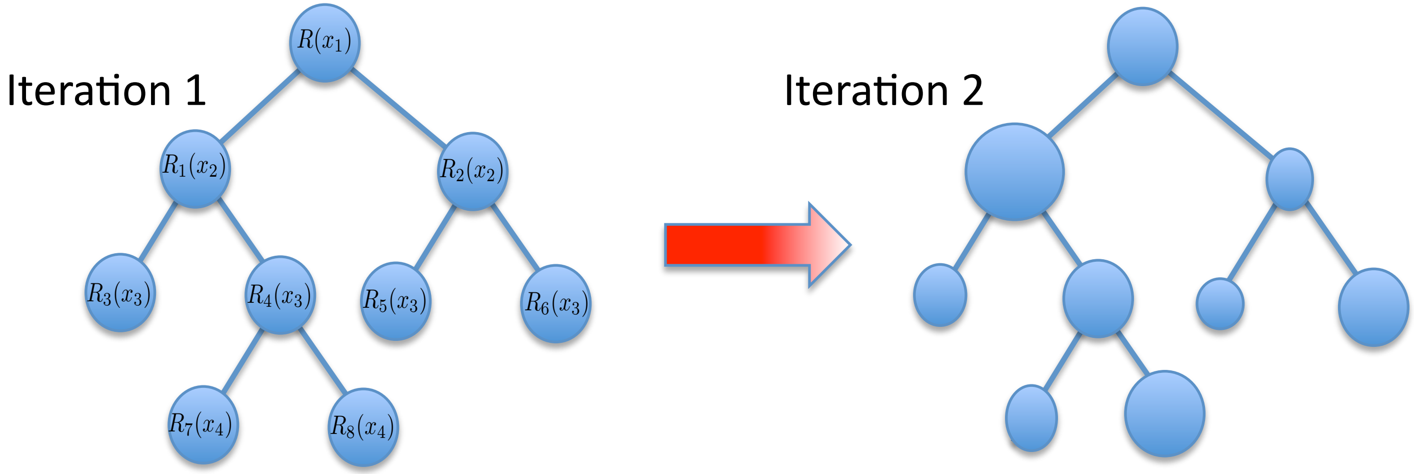
By re-weighting mis-classified events by α the aim is to reduce the error rate of the trained tree, compared with an un-boosted algorithm.

Boosted Decision Trees

- At each stage in training there may be some mis-classification of events (error rate).
  - Assign a greater event weight α to mis-classified events in the next training iteration.

$$\alpha = \frac{1-\epsilon}{\epsilon} \qquad\qquad \epsilon = \text{error rate}$$

  - Re-weight whole sample so that the sum of weights remains the same, then iterate.



Iteration 1

Iteration 2

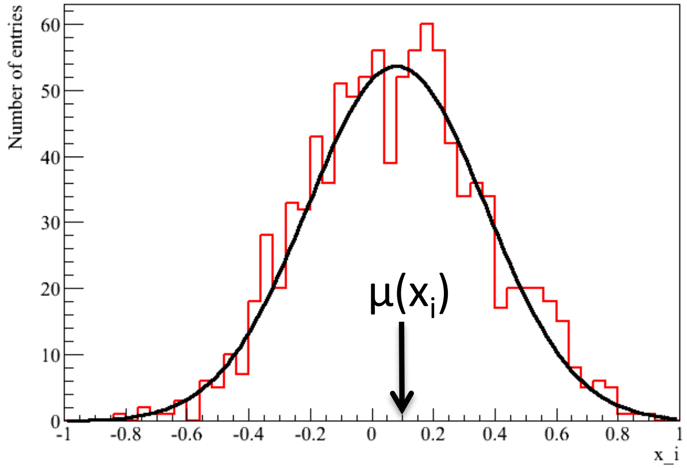
The resulting Boosted Decision Tree tends to be more stable than a normal Decision Tree.

Bagged Decision Trees

- Aim: To improve the stability of a Decision Tree algorithm.
- Solution: Sample the training data used to determine the solution.
- Take the average solution of a number of re-sampled solutions.
  - This re-sampling removes the problem of fine tuning on statistical fluctuations.

Like choosing the mean value of a cut at each level.

Again, the results tend to be more stable than just using a decision tree.



Forests

- A given decision tree may not be stable, so instead we can grow a forest.
  - For a forest, the classification of an event e_i in sample A or B is determined as the dominant result of all of the tree classifications for that event.

    e.g. In a forest of 100 trees, if there are 80 classifications of type A, and 20 of type B, the event e_i is considered to be of type A.

  - Trees in a forest use a common training sample, and are typically boosted.

Randomized Trees / Pruning

- Randomized Trees:
  - A finesse of the previous algorithms is to grow trees using a random sub-set of variables at each node.
- Pruning:
  - Some nodes may be insignificant in obtaining the final tree.
  - It is best to train the tree, then remove the insignificant nodes after the fact.
  - Pruning starts at the bottom and works upward.

Decision Trees

- The ranking of an input variable in a decision tree is derived from:
  - The number of times that it is used in one of the nodes of the tree.
  - Weighting based on the square-class-separation .

- Some final remarks:
  - Insensitive to weak discriminating variables.
  - Simple to understand (compared to a NN for example)
  - Best theoretical performance for a decision tree is inferior to neural networks.

TMVA

A Quick Tour of TMVA

- TMVA = Toolkit For Multivariate Analysis
  - See the TMVA web page for more details, and a user guide:
    - http://tmva.sourceforge.net/
  - TMVA are the following people

TMVA

arXiv:physics/0703039

A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss,
M. Backes, T. Carli, O. Cohen, A. Christov, D. Dannheim, K. Danielowski,
S. Henrot-Versille, M. Jachowski, K. Kraszewski, A. Krasznahorkay Jr.,
M. Kruk, Y. Mahalalel, R. Ospanov, X. Prudent, A. Robert, D. Schouten,
F. Tegenfeldt, A. Voigt, K. Voss, M. Wolter, A. Zemla

- Note*:*
  - *I am not a contributor to this project – I am a user who finds this a nice interface to a large number of algorithms.*

TMVA

- The source code is available for download from the sourceforge web page.
- TMVA can also be pre-compiled as an option when building ROOT.
  - TMVA libraries are available in the versions of ROOT used at SLAC.
  - To load TMVA, start ROOT in the usual way:

```
root -l
```

(use `bbrroot` in a BaBar release)

  - then load the shared library into memory:

```
root[0] gSystem->Load("libTMVA.so")
```

- Now you're ready to use TMVA!

What algorithms are in TMVA

- The algorithms available include:

Cuts
Likelihood
HMatrix
Fisher
MLP
CFMlpANN
TMlpANN
BDT
RuleFit
SVM
BayesClassifier
Committee
MaxMethod

These algorithms are described in more detail in the TMVA user guide. For example there are several NN algorithms available for use – each one a slightly different variant from the next.

Note:

- All of the algorithms have a common interface!

- Any combination of algorithms can be used at one time. This means that the user can compare the results quickly.

- This convenience makes TMVA a useful toolkit for Particle Physics Analysis.

How do I use TMVA

- The following steps should be followed in order to use TMVA:
  - Book the classifiers you want to train.
  - Train the classifiers.
  - Inspect the results obtained and compare.

- Having done this, you can go back and compute a classifier that you want to use in your analysis.

  - Note: Fisher coefficients are computed using scaled input variables by default in TMVA. Use TMVA to compute the Fisher rather than trying to compute this in your own code to avoid any surprises.

How do I use TMVA

- The following steps should be followed in order to use TMVA:
  - Book the classifiers you want to train.
  - Train the classifiers.
  - Inspect

Warning: BaBar releases (analysis-51) use a very old version of TMVA. There is an example macro for this, and conceptually the use of TMVA is no different between the latest version and that available in ROOt 5.14. But the syntax is different. Don't confuse the two!

- Having do mpute a classifier lysis.

  - Note: Fisher coefficients are computed using scaled input variables by default in TMVA. Use TMVA to compute the Fisher rather than trying to compute this in your own code to avoid any surprises.

TMVA: Preparing data

- You need to prepare a tree of signal and background events.
- Then prepare the TMVA Factory:

Specify the output file (useful output will be stored here)

```
TFile * fout = new TFile("TMVA.root", "RECREATE");
TMVA::Factory factory("BAS09-TMVA-ANALYSIS", fout, "");
```

Add the signal and background trees to the factory

```
factory.AddSignalTree( t_signal, 1.0);
factory.AddBackgroundTree( t_bg, 1.0 );
```

Specify the variables to use in the classifiers you want.
These variables should be in the signal/background trees.

```
factory.AddVariable("bCosTBTR", 'F');
factory.AddVariable("sumPtR", 'F');
factory.AddVariable("lgdr0P1n", 'F');
.
.
.
```

TMVA: Booking Classifiers

- Similar syntax for all types of classifier:

Make sure that any training/validation data splits required are specified. In the examples, 50% of the data will be used as a training sample, and 50% as a validation sample.

```
factory.PrepareTrainingAndTestTree(theCut, 5000, 5000, 5000, 5000);
```

Then book the type or types of classifier that you want to run.

```
factory.BookMethod(TMVA::Types::kFisher, "Fisher");
factory.BookMethod(TMVA::Types::kMLP, "MLP");
factory.BookMethod(TMVA::Types::kBDT, "BDT");
```

TMVA: Training Classifiers

- This step is the same, irrespective of how many classifiers that you're using:

Train the classifier methods that you've booked.

```
factory.TrainAllMethods();
```

Having trained the methods can be applied to the specified validation sample for further inspection. The validation information is included in the `TestTree` written to the output file.

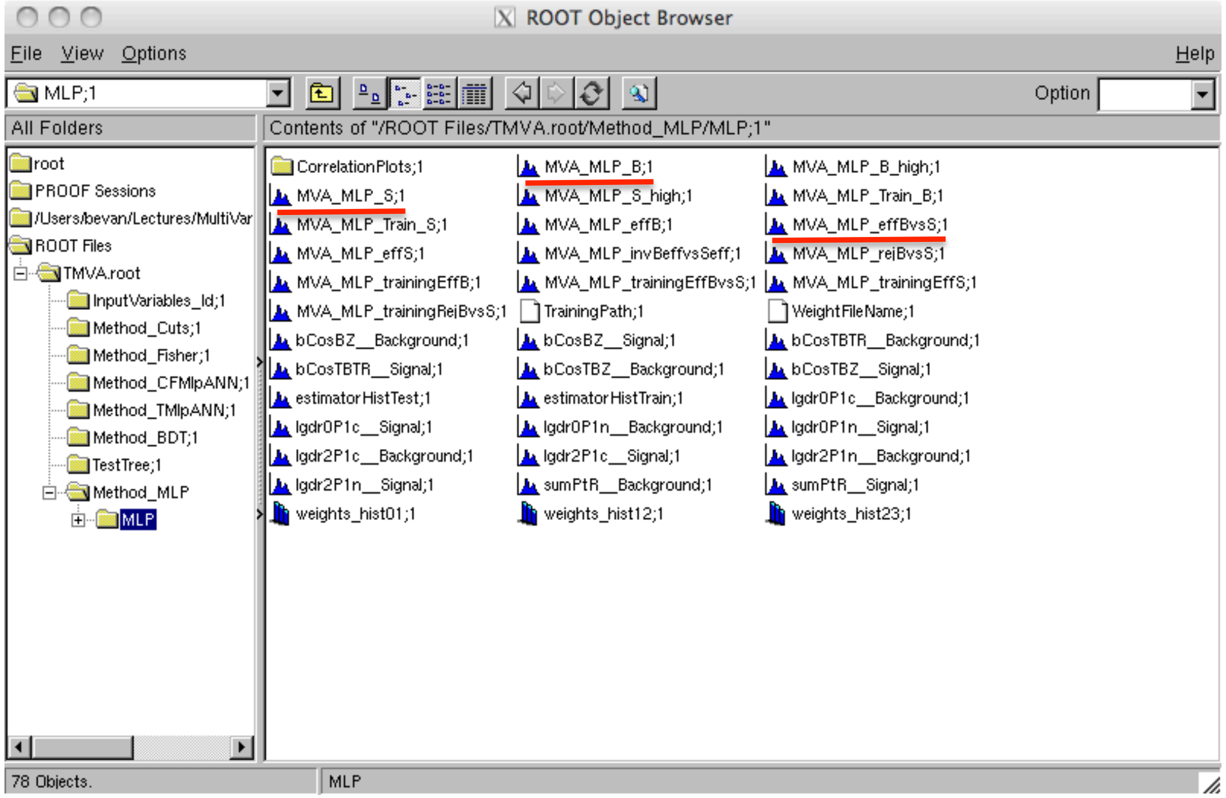
```
factory.TestAllMethods();
```

Simple evaluation of the performance of methods: compute correlations and rank the variables (see stdout printout [keep a log file of the stdout for inspection!] .

```
factory.EvaluateAllMethods();
```

TMVA: Inspecting Results

- You should have a ROOT file (`TMVA.root` in this case) and a log file to inspect.



```
root
PROOF Sessions
/Users/bevan/Lectures/MultiVar
ROOT Files
  TMVA.root
    InputVariables_Id;1
    Method_Cuts;1
    Method_Fisher;1
    Method_CFMlpANN;1
    Method_TMlpANN;1
    Method_BDT;1
    TestTree;1
    Method_MLP
      MLP
```

Contents of "/ROOT Files/TMVA.root/Method_MLP/MLP;1"

```
CorrelationPlots;1            MVA_MLP_B;1                   MVA_MLP_B_high;1
MVA_MLP_S;1                   MVA_MLP_S_high;1              MVA_MLP_Train_B;1
MVA_MLP_Train_S;1             MVA_MLP_effB;1                MVA_MLP_effBvsS;1
MVA_MLP_effS;1                MVA_MLP_invBeffvsSeff;1       MVA_MLP_rejBvsS;1
MVA_MLP_trainingEffB;1        MVA_MLP_trainingEffBvsS;1     MVA_MLP_trainingEffS;1
MVA_MLP_trainingRejBvsS;1     TrainingPath;1                WeightFileName;1
bCosBZ__Background;1          bCosBZ__Signal;1              bCosTBTR__Background;1
bCosTBTR__Signal;1            bCosTBZ__Background;1         bCosTBZ__Signal;1
estimatorHistTest;1           estimatorHistTrain;1          lgdr0P1c__Background;1
lgdr0P1c__Signal;1            lgdr0P1n__Background;1        lgdr0P1n__Signal;1
lgdr2P1c__Background;1        lgdr2P1c__Signal;1            lgdr2P1n__Background;1
lgdr2P1n__Signal;1            sumPtR__Background;1          sumPtR__Signal;1
weights_hist01;1              weights_hist12;1              weights_hist23;1
```

78 Objects.   MLP

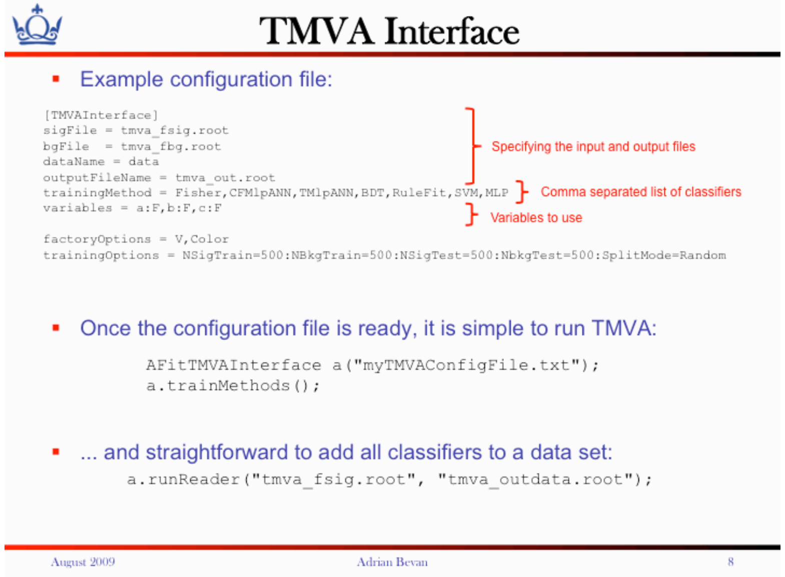
For each method booked, you should find a directory that contains histograms that quantify the performance of the classifier.

The output distributions and ε_S vs. ε_B plot are a good place to start trying to understand the computed classifier.

More information is given in the TMVA user guide on how to inspect and evaluate the output on the training process.

A final word on TMVA

- The AFit package (like RooRarFit this is a higher level wrapper for RooFit) has an interface to TMVA:
  - http://pprc.qmul.ac.uk/~bevan/afit/
  - AFitTMVAInterface was written to:
    - Provide a simple text file driven interface to TMVA.
    - Having trained classifiers, to also be able to simply add those variables to a RooDataSet so that they can be included in a fit.



TMVA Interface

- Example configuration file:

```
[TMVAInterface]
sigFile = tmva_fsig.root
bgFile  = tmva_fbg.root
dataName = data
outputFileName = tmva_out.root
trainingMethod = Fisher,CFMlpANN,TMlpANN,BDT,RuleFit,SVM,MLP
variables = a:F,b:F,c:F

factoryOptions = V,Color
trainingOptions = NSigTrain=500:NBkgTrain=500:NSigTest=500:NbkgTest=500:SplitMode=Random
```

Specifying the input and output files

Comma separated list of classifiers

Variables to use

- Once the configuration file is ready, it is simple to run TMVA:

```
AFitTMVAInterface a("myTMVAConfigFile.txt");
a.trainMethods();
```

- ... and straightforward to add all classifiers to a data set:

```
a.runReader("tmva_fsig.root", "tmva_outdata.root");
```

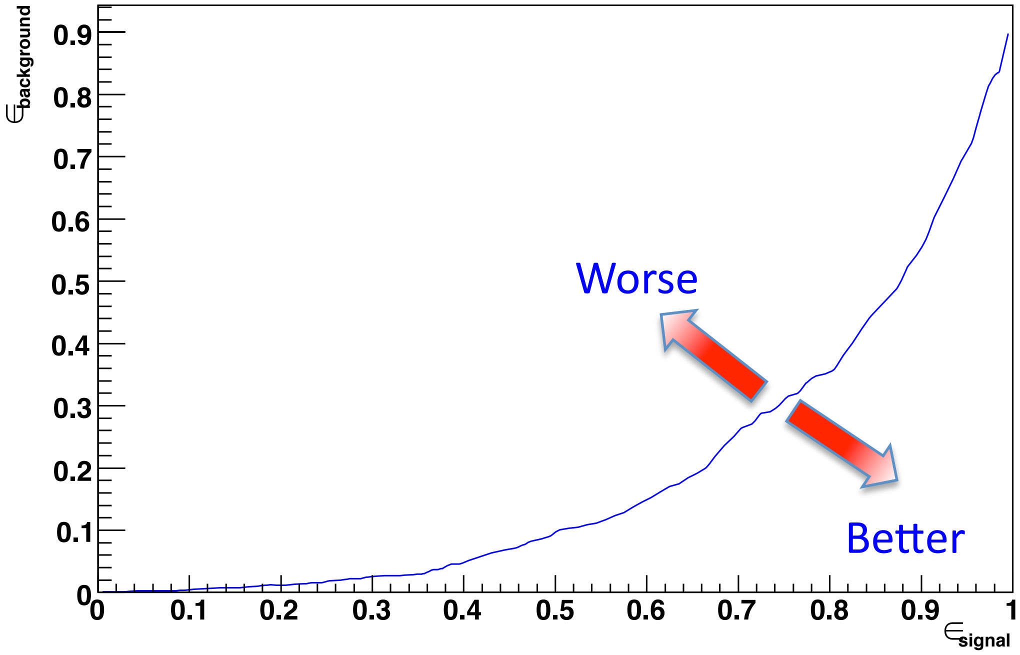
August 2009 Adrian Bevan 8

How can I compare algorithms?

- There are several ways of looking at algorithms and comparing their performance.
- Usually compare

$$\epsilon_{\text{signal}} \text{ vs. } \epsilon_{\text{background}}$$

for several MVAs



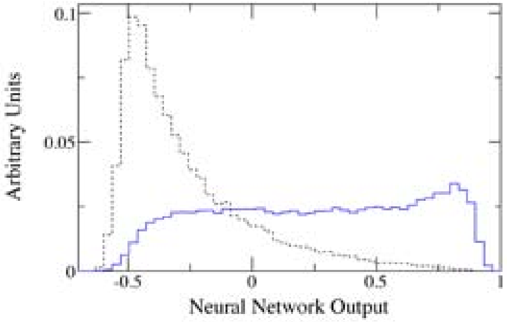
How can I compare algorithms?

- There are several ways of looking at algorithms and comparing their performance.
- Usually compare

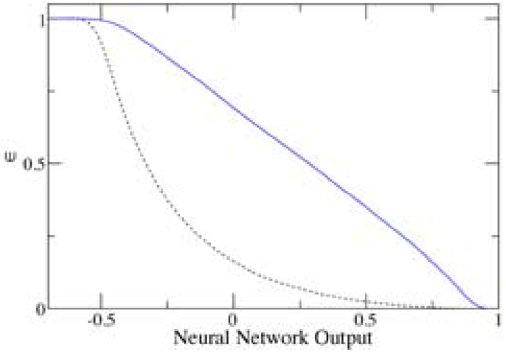
$$\epsilon_{\text{signal}} \text{ vs. } \epsilon_{\text{background}}$$

- Can also view the output variable and look at ε as a function of the algorithm's output.

From B. Aubert et al., PRD 76, 052007 (2007)



Shape of signal and background outputs of an MLP



Efficiency as a function of the NN

Can use this information to determine an optimal cut on the NN output if so desired.

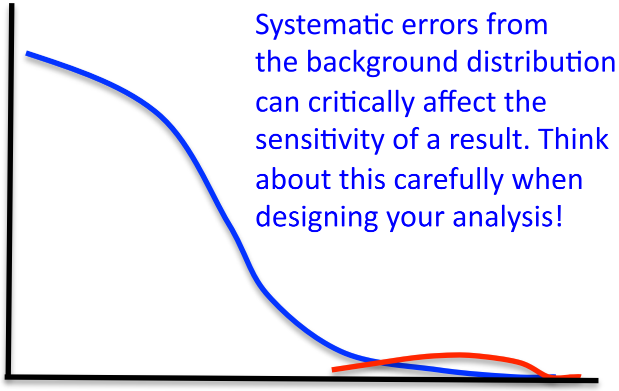
Note: note done for the particular example shown.

How can I compare algorithms?

- It can be useful to compare the performance of an algorithm on data.

- And it is instructive to look in more detail at how we want to use the output distribution in an analysis.

- We should always remember that to fully evaluate one algorithm over another we also have to consider systematic uncertainties.
  - If we have a tail of a distribution that dominates our data sample – do we understand how we can compute a sensible systematic uncertainty from this?
  - Do we understand the bulk of the distribution, and all other factors to extrapolate into the extremes?
  - What happens if we have not done this well?
    - Is our analysis still valid?
    - Do we have to go back to the drawing board?
    - Could we have done something different to avoid problems?



Systematic errors from the background distribution can critically affect the sensitivity of a result. Think about this carefully when designing your analysis!

What am I going to do with my MVA?

- This question is related to the comparison issue.
  - How you use the MVA may relate to the systematic uncertainties you generate.

Cut on the variable to enhance signal

For example include the MVA in a cut based analysis to simplify optimization of cuts when dealing with a number of inputs.

Use as an input to a fit

To use as a background suppression variable in a fit that may or may not include other discriminating variables: single top search from the Tevatron, almost any BaBar analysis.

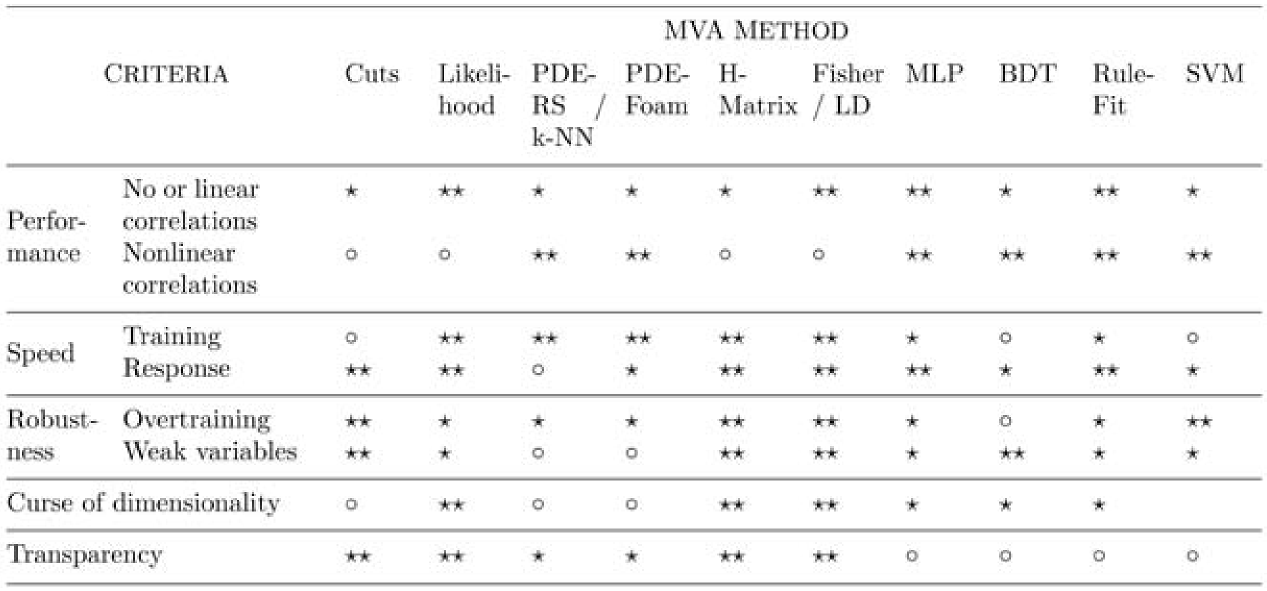
Use as an input to another MVA

For example the BaBar B-flavor tagging algorithm that has a number of sub-nets that are combined into a single output xNN that quantifies the B or B-bar nature of an event.

some other use ...

Which Algorithm to use?

- The TMVA user guide has the following summary table to help users think about this question:



		MVA METHOD									
CRITERIA		Cuts	Likelihood	PDE-RS / k-NN	PDE-Foam	H-Matrix	Fisher / LD	MLP	BDT	Rule-Fit	SVM
Performance	No or linear correlations	⋆	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
	Nonlinear correlations	○	○	⋆⋆	⋆⋆	○	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆
Speed	Training	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆	○	⋆	○
	Response	⋆⋆	⋆⋆	○	⋆	⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
Robustness	Overtraining	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	○	⋆	⋆⋆
	Weak variables	⋆⋆	⋆	○	○	⋆⋆	⋆⋆	⋆	⋆⋆	⋆	⋆
Curse of dimensionality		○	⋆⋆	○	○	⋆⋆	⋆⋆	⋆	⋆	⋆	
Transparency		⋆⋆	⋆⋆	⋆	⋆	⋆⋆	⋆⋆	○	○	○	○

★★	= good
★	= fair
O	= bad

Visualizing data in hyperspace

- Matt Bellis recently introduced BaBar to a NASA developed programme: Viewpoints
  - http://astrophysics.arc.nasa.gov/~pgazis/viewpoints.htm
  - http://www.slac.stanford.edu/~bellis/viewpoints_demo.html
  - This is a useful way of visualizing data in an n-dimensional hyperspace that has been used to visualize large numbers of dimensions (n up to 100) and large samples of data (RAM dependent >10^6 events).
  - Matt has given a comprehensive introduction, so I won't repeat that – please think about this tool when you are trying to understand how your n-dimensional MVA is behaving!
- If you're interested in this please see Matt's talk on Wednesday at 11am!

Summary #2

- We have looked at some modern MVA techniques to extend our understanding beyond cutting on data, Fisher discriminants, and MLPs.

- We've taken a quick look at TMVA as a useful toolkit.
  - In the next session we will use this in a tutorial.

- There are other tools that can be useful as well: See Matt Bellis' talk: Wednesday at 11am!

- The question of what is the right algorithm to use has been re-visited with consideration of algorithms beyond complexity and comprehension.
  - This is not a simple problem to cover, and is something that has to be considered carefully on a case-by-case basis.