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

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.*
  - *There was a TMVA day after the PHSTAT conference at CERN last week. See the agenda for details:*

    http://indico.cern.ch/conferenceDisplay.py?confId=112879

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
- Having do mpute a classifier lysis.

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!

  - 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.

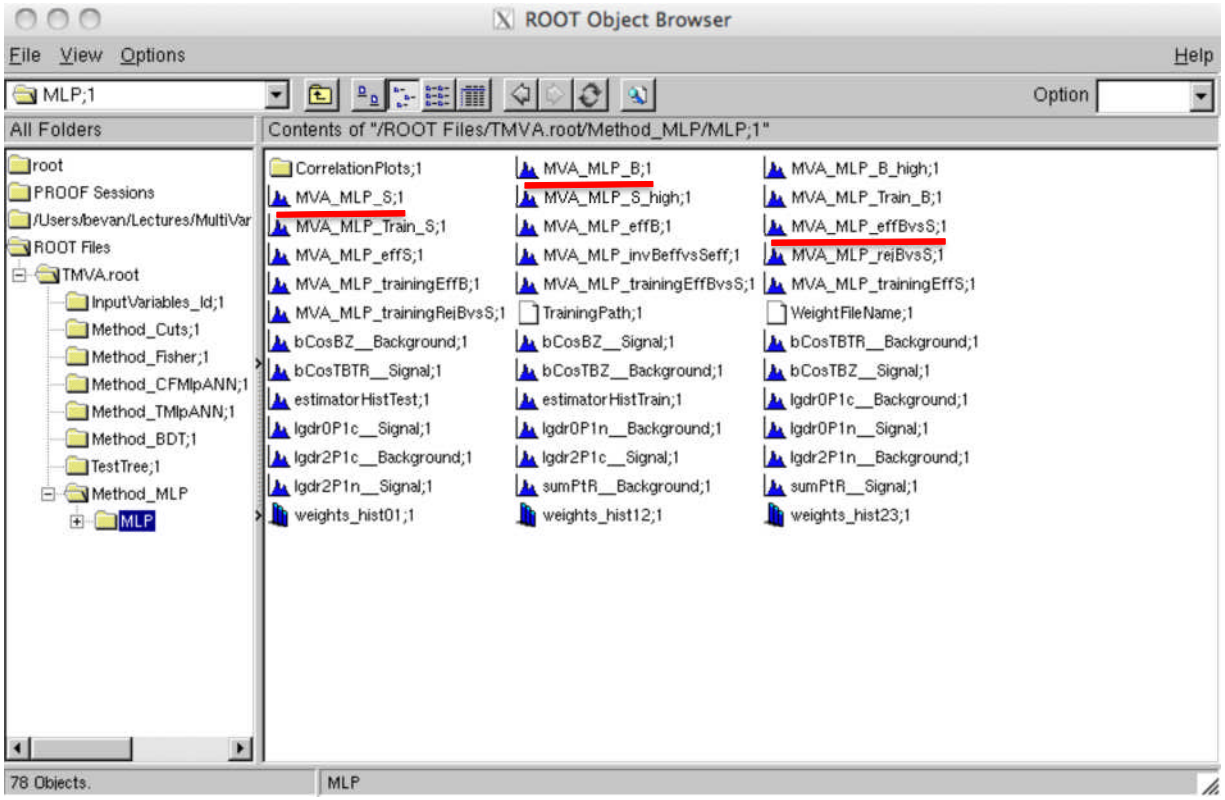
```
factory.TestAllMethods();
```

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.



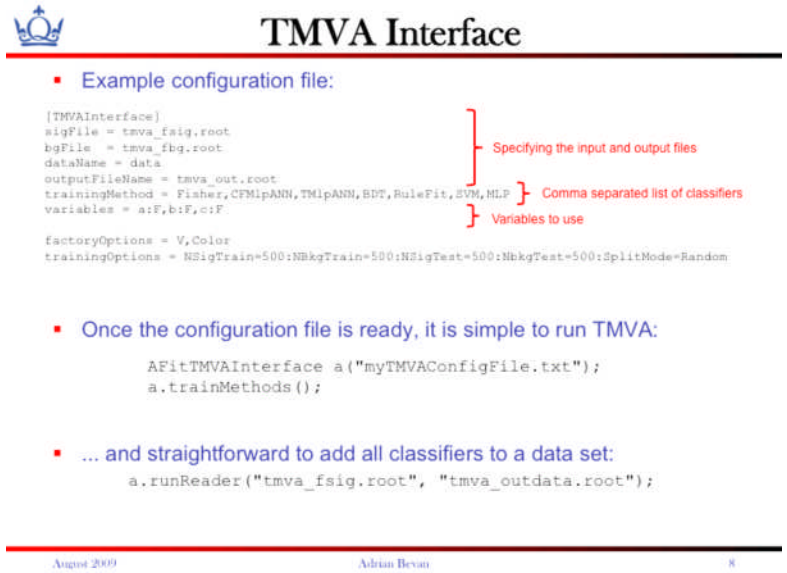
For each method booked, you should find a directory that contains histograms that quantify the performance of the classifier.

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");
```

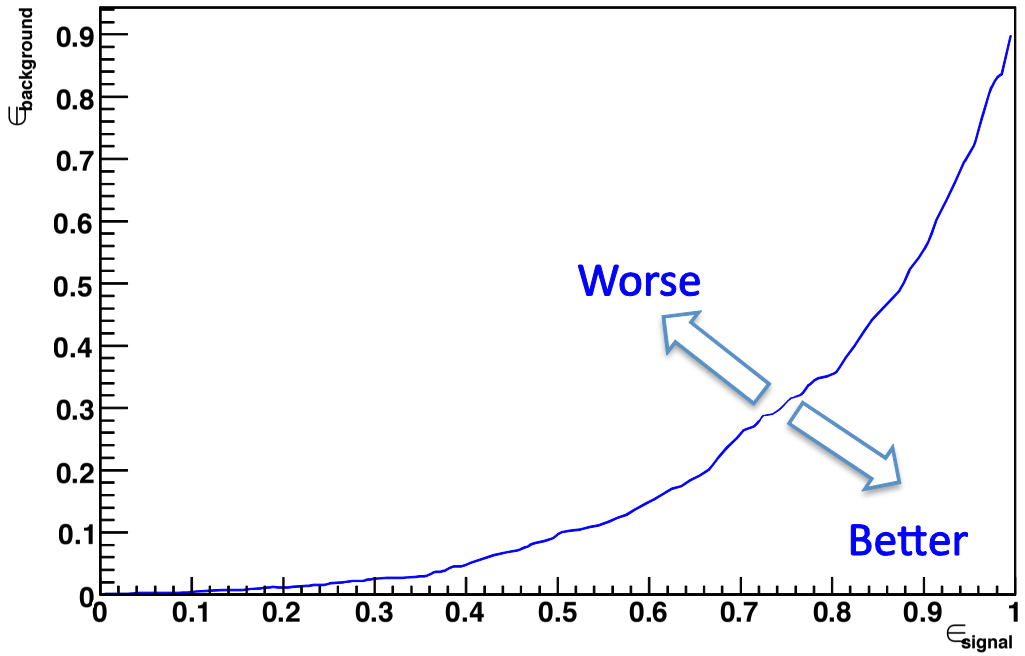
August 2009 Adrian Bevan 8

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



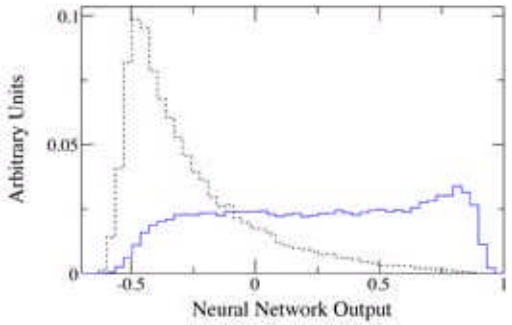
How can I compare algorithms?

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

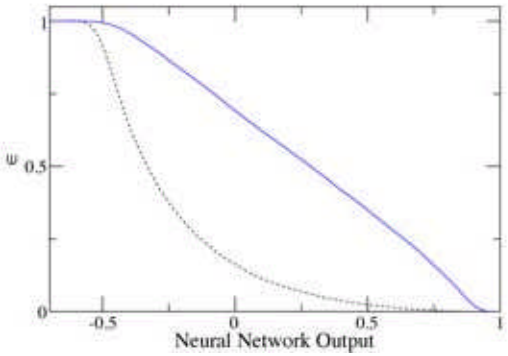
$$\epsilon_{\text{signal}} \text{ VS. } \epsilon_{\text{background}}$$

- 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



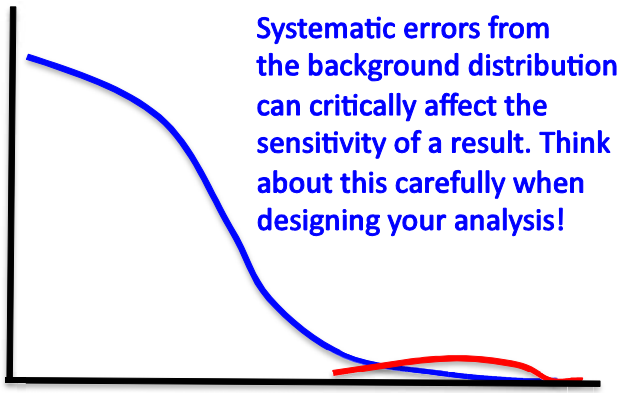
Efficiency as a function of the NN

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:

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		★★	★★	★	★	★★	★★	○	○	○	○

★★ = good
★ = fair
O = bad