INFN Statistic School 2015

Multivariate Classification

General remarks on the computer exercise using TMVA

In ROOT6 (and perhaps future ROOT5 versions) all TMVA GUIs, plotting macros are part of the library, but right now we still have actually include the macros in the 'root macro path' and exectue a 'setup.sh' that sets the environment

both files are included in the 'INFN2015/Exercises/' directory in which you are supposed to do the exercises:

```
   wget http://hvoss.home.cern.ch/hvoss/INFN2015Exercises.tgz
tar -zxvf INFN2015Exercises.tgz
cd INFN2015/Exercises/
source setup.sh (or .csh) </full-path-to-your-ROOT-installation/bin
```

1 Understand the advantages of multivariate methods (e.g. Likelihood) over simple 'univariate' (rectangular) Cuts

This exercise is divided into two parts. In the first part only dealing with uncorrelated variables. In the second part the variables are correlated.

1.1 Uncorrelated Input Variables

Manually optimizing cut values is tedious and hence in TMVA there are also automatic ways to do this. One of the algorithms used to maximise background rejection for all given signal selection efficiencies is a Genetics Algorithm (the details of which are not discussed here). You can use one of the prepared macros to run this algorithm.

1. Execute
   `$ root TMVAClassification_uncorrelatedGauss.C\(\"Cuts\"\)`
   and look at the result (the GUI will pop up)

2. Take look at the "input variables", "correlations", and "2D scatter plots
   (sorry, in this version, sometimes you have to 'first look at the 'input variables linear correlation' before you can look at the 'scatter plots')

3. Take a look at the "Classifier Cut efficiencies" (respective GUI-button) What does it show? Assuming equal numbers of 1000 signal and 1000 background events in your data sample (just as it is in the training data), what is the cut value with best $S/\sqrt{S+B}$ (statistical significance)?

1.1.1 Comparison of "Cuts" with projective Likelihood

Ccompare the result of the "cut-based" analysis with the "projective Likelihood". They can be run in one job if you execute the macro like this:
`$ root TMVAClassification_uncorrelatedGauss.C\(\"Cuts,Likelihood\"\)`

1. Have a look at the *Classifier Output Distribution*. What does that plot represent ?
2. Pick the *Receiver-Operation-Characteristics* (ROC) curve form the GUI. All classifiers that are used in "one training run" are displayed in order to be able to compare them. (here: Cuts and Likelihood).
   (a) What would be the *ideal* ROC curve?
   (b) What is the best possible ROC curve achievable for this current example data? I.e. do you know what the best possible separation boundary is for (un)correlated multidimensional Gaussians, and why ?
   (c) Derive the functional form of the optimal decision boundary, if the probability density (PDF) for signal and background are both "multivariate Gaussians" with same correlation matrix (i.e. the PDFs differ only by their mean values).
      i. Write down the general optimal y(x) given by Neyman-Pearsons lemma

$$y(x) = PDF(x|S)/PDF(x|B) = \frac{exp(-(x-x_s)^T V(x-x_s)}{exp(-(x-x_B)^T V(x-x_B)}$$

$$= exp(-(x-x_s)^T V(x-x_s) + (x-x_B)^T V(x-x_B))$$

      ii. which obvious "monotonous" transformation you could think of to make it simpler? taking the logarithm we get a new y(x):

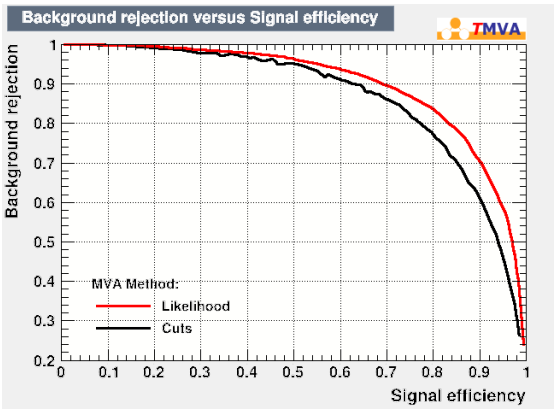
$$y(x) = -(x-x_s)^T V(x-x_s) + (x-x_B)^T V(x-x_B)$$

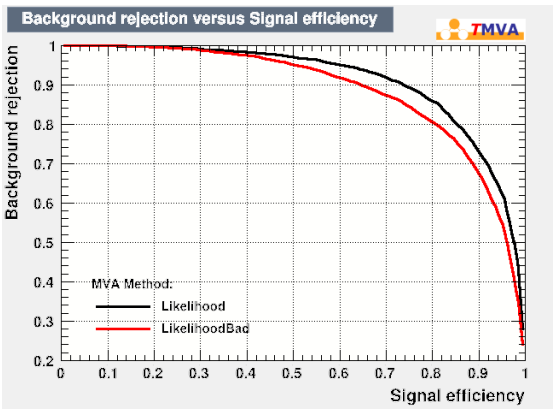
      due to the condition that $V_S = V_B = V$, the quadratic terms xVx cancel, the terms quadratic in x_s and x_B are constant (independent of x) and hence we end up with a linear function in x

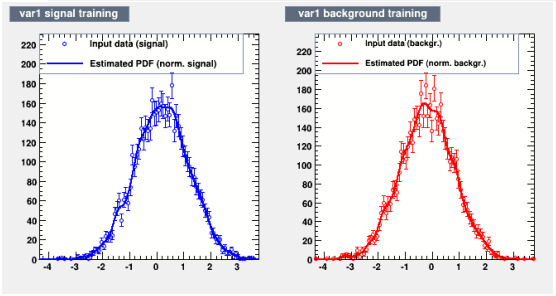
$$y(x) = const - x^T V(x_s - x_B) - (x_S - x_B)^T Vx$$

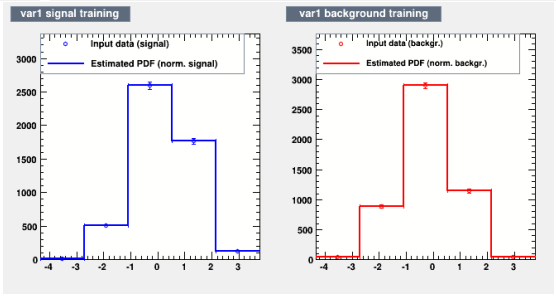
      Therefore the optimal decision boundary is linear (a linear hyperplane)
   (d) Which of the two methods (Cuts and Likelihood) shows better performance?
   (e) Can you explain in words why one of the methods is better?
3. Select the display of the Likelihood reference distributions from the GUI.
   (a) Can you explain in words what these reference distributions are? What are they supposed to represent/estimate?

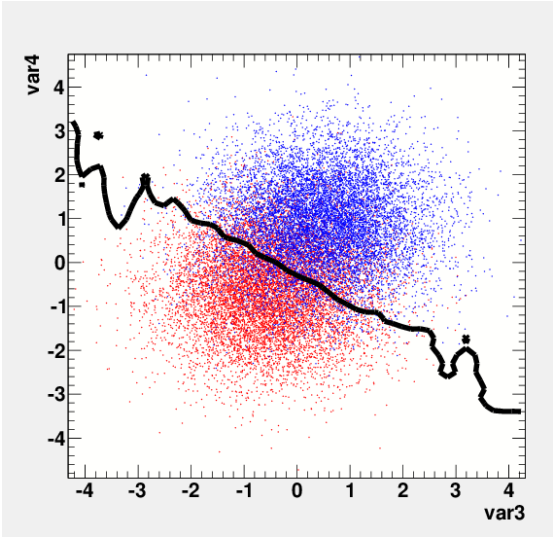
(b) What happens if you set in the maro for the Likelihood classifier the option: `NAvEvtPerBin=1000` and look at the reference distributions again? Which one do you think is 'better' ? note: all options available for the various classifiers are listed on
`http://tmva.sourceforge.net/optionRef.html`











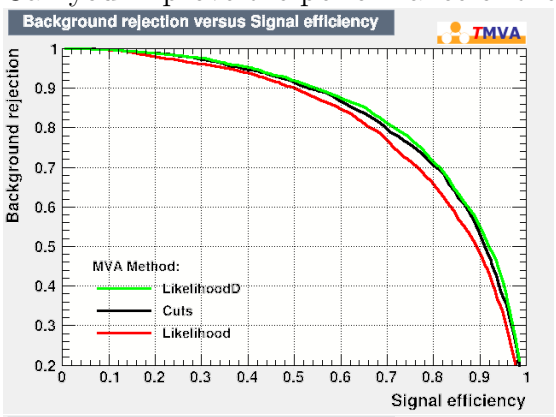
the best decision boundary between two Gaussian distributed variables is linear. As one can see, where there is enough training data, the Likelihood is getting very close to the optimal linear boundary

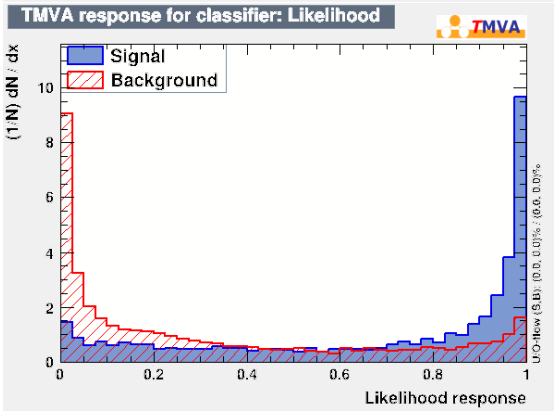
1.2 Correlated input variables

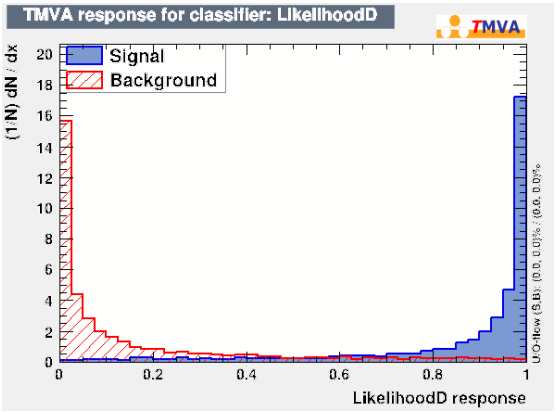
Up to now the input variables were uncorrelated. There is a second data file and macros for with a 4 correlated Gaussian distributed variables.

1. Redo the exercise 1.1.1 with the correlated data, using the example macro: TMVAClassification_correlatedGauss.C Has something changed in the performance difference between the cuts and the Likelihood method? Why is that ?

2. Can you improve the performance of the Likelihood ? How?







2 Fisher Discriminant

2.1 Compare Fisher and Likelihood

First use the same uncorrelated and correlated data sets as before, but now we use all for variables.

1. Run the Likelihood and Fisher simultaneously on the uncorrelated data:
   `$ root TMVAClassification_uncorrelatedGauss.C\(\"Likelihood,Fisher\"\)`

2. How do the performances compare to each other? Can you explain why?

3. Now run the same methods on the correlated data:
   `$ root TMVAClassification_correlatedGauss.C\(\"Likelihood,LikelihoodD,Fisher\"\)`

4. Compare the performances between Likelihood and Fisher discriminant and explain the 'behaviour'

2.1.1 Classifying with slightly more complexity

Now use a new data set:
`data_gaussWithSameMean.root`

1. Run the Likelihood and Fisher simultaneously:
   `$ root TMVAClassification_gaussWithSameMean.C\(\"Likelihood,Fisher\"\)`

2. Which of the two classifiers performs better?

3. The Fisher performs bad in this case. Find out why by

   (a) looking at the input distributions

   (b) understanding what the Fisher discriminant is actually trying to optimise.

   (c) Explain why this strategy does not work on this data.

4. A (manual) transformation of the input can solve this problem:

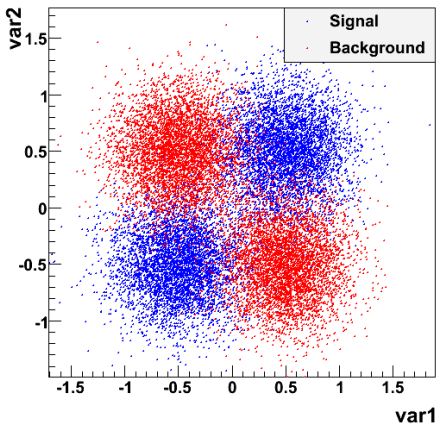
   (a) Try to think of a variable transformation that fixes the problem. (Note: Simple variable transformations can be immediately given in the `AddVariable` command (e.g. `factory->AddVariable("log(var0)")`). Apply a suitable transformation and then run again.

   (b) Why has the picture changed (i.e. perhaps Fisher now performs better than the Likelihood)?

   (c) Now .. can you also "recover" the Likelihood performance?

3 Boosted Decision Trees

To introduce the Boosted Decision Trees another problem is taken. The chessboard data is a bit more difficult than the simple Guassians. We consider first a simple 2 x 2 board:



, the data file is `data/data_schachbrett2x2_2D.root`

1. Look at this Signal and Background distribution and write down by hand the decision tree that would optimally discriminate between signal and background.
2. Run the macro `TMVAClassification_Schachbrett2x2_2D.C\(\"BDT\"\)`
   (a) Have a look from the TMVAGui::Decision Trees at how the Decision Tree looks like.
   (b) How does this tree compare with the one you've written down ?
   (c) Why doesn't the "stupid" Decision Tree in TMVA find this perfect apparently so simple solution?
   (d) Understand how this decision tree is build. (see lecture)
   (e) Add more nodes to decision tree (larger MaxDepth) and see how it develops
   (f) Now *instead of increasing MaxDept* add more trees to the forest and see how many you need to get very good separation (in the macro).
3. Take a look at the decision boundaries resulting from the cut on the MVA values:
   - Open a root session
   - `root [3] .L PlotDecisionBoundary_Schachbrett.C` +
   - `root [4] PlotDecisionBoundary_Schachbrett()`
4. Now run also the MLP and see how that one performs.
5. Can you improve the performance of the MLP? (see MLP Training options)
   (a) Look at the architecture that is there as 'default' today in this macro

(b) Increase the number of nodes. How many do you need ? Play a little bit until you are happy with the MLP and BDT performance.

(c) Now: Black boxes, even powerful ones, should never discourage you to first think about your data. What does the intelligent physicist do with the variables?

(d) How does the truly "optimal" classifier compare in performance with your previously trained BDT and Neural Network?