ETH Advanced Statistical Computing Workshop 2014

Multivariate Classification Algorithms

Note:
As a starting point for the 'exercise' there are some ROOT macros available that need to be 'extended'

There is very little 'ROOT' functionality used - basically just the histograms for visualisation, hope that is kind of 'self explaining' for those that do not know ROOT. Have a look at the (solutions) macros for some "guidance"

```
./exercises/macros
```

and "solutions" (if you need) in

```
./exercises/solutionsMacros
```

1 Neural Network: MLP

Build your own "neural network". Try to "see" a little bit "how" this piece-wise construction of a decision boundary works.

1. Use two input variables x1,x2 (or x[0], x[1]) in the range of [-10,10][-10,10] and plot the "sigmoid" function of the input x+y to this function. (see miniMLP.C)
   In order to run the macro, use the command:
   `root`-l miniMLP.C+ *
   What you have now is a very simple neural network with 2 input variables, no hidden layer and one output node.

2. Looking at this function, what are the parameters (weights) ?

3. What does the 'decision boundary' look like?

4. Change the 'weights' and see what happens to the decision boundary.

5. Write a second function with different *input weights*, and plot it?

6. Now build a slightly more interesting network out of these two functions by "adding them together" (taking a linear combination between the output of the two functions) (miniMLP2.C)
   For this, one typically also adds a "bias" node.. which is as if there were an additional variable (observable) that has a constant value of '1'. This allows to shift the 'active area' of the sigmoid in a simple way, which otherwise would have to be done by changing all weights simultanously.

7. What does the decision boundary look like now? What has changed ?

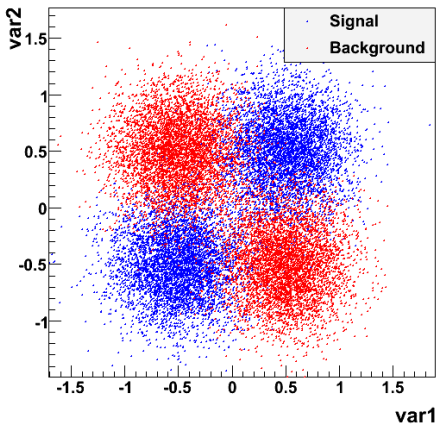
8. What "architecture" (i.e. #nodes,#layers) does this correspond to?

9. Look at the resulting function in 2D. Can you see what the "decision boundaries" of this functions would look like?

10. "Play" a bit with the 'weight parameters' to make a good 'classifier' for the example events given in 'miniMLP4.C'

2 Boosted Decision Trees

Consider a simple 2 x 2 "chess-board" like signal and background event distribution:



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. What was the 'figure of merit' that the 'decsion tree learning algorithm' uses to determine the split?

3. Which first split would this algorithm come up with?

4. Why does it work nonetheless ?