Table of Contents

TMVA - Toolkit for Multivariate Data Analysis

Tutorial (updated 17 June 2007, currently under construction)

Preparation

The following tutorial has been tested for TMVA-v3.8.4 and various ROOT 5 releases. Since remote ROOT graphics is slow it is preferred that the users run the tutorial locally, i.e., have a ROOT >=5 release and TMVA-v3.8.4 installed locally. **Take care that the `ROOTSYS` environment variable is properly set**.

Local installation

It is assumed that ROOT >=5 has been installed (binaries are sufficient). If not, please consult the ROOT download page.

1. Download TMVA-v3.8.4 from sourceforge
2. Unpack the tar.gz file:

   ```
   tar xfz TMVA-v3.8.3.tgz
   ```
3. Check the contents of the package:

   ```
   cd TMVA
   ls -l
   ```

   The top directory contains `README` and `LICENSE` text files as well as a setup script for bash (zsh) and (t)csh. Content of the subdirectories:

   - **`src`**: header and source files of all TMVA classes
   - **`include`**: symbolic link to directory containing include files (link created by setup script)
   - **`lib`**: contains shared library `libTMVA.1.so` after source code compilation
   - **`macros`**: example scripts for a training and application analysis, and all plotting scripts for the TMVA evaluation
   - **`examples`**: example executables for a training and application analysis
   - **`python`**: example python script for a training analysis
   - **`reader`**: *obsolete*
4. Setup the environment:

   ```
   source setup.[c]sh
   ```
5. Create the TMVA shared library: `libTMVA.1.so` (this will take a few minutes)

   ```
   cd src
   make
   ```

   Note that if you use ROOT >= 5.11/03, older versions of TMVA will also be part of the ROOT distribution. There should not be any conflict with the newly created package as long as `libTMVA.1.so` is properly loaded (or linked).

Using CERN afs

Running the example analysis as a ROOT macro (using CINT)

TMVA comes with a toy data sample consisting of four Gaussian distributed, linearly correlated input variables. It is sufficient to illustrate the salient features of a TMVA analysis, but it should be kept in mind that this data is by no means representative for a realistic HEP analysis.

- To quickly run the example macro for the training of a Fisher discriminant, type the following:

  ```
  cd macros
  root -l TMVAnalysis.C\(\"Fisher\"\)
  ```

  The macro should run through very quickly, and pop up a GUI with buttons for validation and evaluation plots. The buttons show the distributions of the input variables (also for preprocessed data, if requested), correlation profiles and summaries, the classifier response distributions (and probability and *Rarity* distributions, if requested), efficiencies of cuts on the classifier response values, the background rejection versus the signal efficiency, as well as classifier-specific validation plots. **Try them all!**
- We can now attempt to train more than one classifier, e.g., Fisher and the projective likelihood:

  ```
  root -l TMVAnalysis.C\(\"Fisher,Likelihood\"\)
  ```

  Again the GUI will pop up, and it is interesting to check the background rejection versus the signal efficiency plot. Fisher performs better, which is due to the ignorance of correlations in the projective likelihood classifier. Plotting the likelihood response distributions indicates the problem: background slightly rises in the signal region, and vice versa.
- The likelihood performance on this toy data can be improved significantly by removing the correlations beforehand via linear transformation using a TMVA data preprocessing step:

  ```
  root -l TMVAnalysis.C\(\"Fisher,Likelihood,LikelihoodD\"\)
  ```

  Plotting the background rejection versus the signal efficiency graph shows that the decorrelated likelihood classifier is now as performing as Fisher.
- If time is sufficient, we could now run **all** preset classifiers, which may take O(10 mins):

  ```
  root -l TMVAnalysis.C
  ```

  After terminating, try how the the background rejection versus the signal efficiency plot looks like. Also, it is interesting to look into the response distributions of the classifiers.

Looking at the training analysis code

Let's look at the analysis script:

```
emacs TMVAnalysis.C
```

Applying the trained classifiers to data

Once the training phase (including training, testing, validation and evaluation) has been finalised, selected classifiers can be used to classify data samples with unknown composition of signal and background events:

```
root -l TMVApplication.C\(\"Fisher,Likelihood,LikelihoodD\"\)
```

The macro runs over signal events only. Plot the produced histograms:

```
.ls
```

```
MVA_Fisher.Draw()
MVA_Likelihood.Draw()
MVA_LikelihoodD.Draw()
```

Looking at the application code

Let's look at the application script:

```
emacs TMVApplication.C
```

Links:

- TMVA home
- TMVA download
- TMVA Users Guide
- TMVA sourceforge project page

-- AndreasHoecker - 17 Jun 2007

This topic: Main > TMVA
Topic revision: r3 - 2010-11-12 - EckhardVonToerne