The future of TMVA (Draft version 16/09/2015)

Introduction:

- **TMVA is the ROOT-integrated package for Machine Learning (ML)**
- **Commonly used in many published HEP analyses**
- **In production software of the two major LHC experiments (ATLAS/CMS)**
- **Provides the first point of contact for people in HEP trying to use ML**
- **Has basic neural networks, boosted decision trees, etc**
- **Provides a common interface and associated support - very useful to HEP**
- **Written about 10 years ago, and ML has evolved significantly in that time**
  - **Originally written to introduce ML techniques to the HEP community**
  - **It has now fulfilled that purpose - time to move to the next stage**

TMVA use cases:

- **Standard analysis users (exotics, SUSY, etc)**
  - **Happy with standard BDT, etc**
  - **Would benefit from integration of modern algorithms and full TMVA support**
- **Precision analysis or performance users (for example: b-tagging)**
  - **Want the best performance possible**
  - **Willing to invest significant time into cutting edge algorithms**
  - **Already willing to work with ML algorithms which are not yet in TMVA**
  - **Would benefit from having a generic TMVA interface to make the process more straightforward that includes a consistent toolkit for evaluation/monitoring**
- **Advanced users and other interested parties**
  - **Are willing to probe the limits of what is possible in TMVA**
  - **Are potential future TMVA developers or at least contributors**
  - **Have the potential to put the LHC at the forefront of ML in the areas relevant to HEP (instead of just playing catch-up, they may try new things)**
- **ML experts who are working in HEP or interested in HEP problems (challenges, etc)**
  - **Are used to the flexibility of using languages/libraries as necessary**
  - **Work on pushing the boundaries of ML, want to use this in HEP**
  - **Are likely to write code outside of TMVA and want to use it with an interface**

Current status of ML:

- **Modern BDTs (such as XGBoost) considerably outperform TMVA BDTs both in speed and accuracy, as seen in the HiggsML challenge.**
- **Other recent ML developments promise significant gains when applied to the right set of problems (for example: deep learning). They are also developed and validated by a larger ML community**
- **Modern ML packages are optimized for computing performance, often making use of parallelization and/or GPUs, with flexible data access where only the relevant part of the dataset is held in memory (important for scalability)**

Core requirements:

- **The core TMVA package should provide a set of competitive and simple algorithms for standard HEP analysis usage**
    - **XGBoost is a promising C++ package for integration as the core BDT algorithm**
    - **Other core algorithms should also be updated**
- **TMVA interfaces for R and python (with support libraries) for high-performance use**
    - **Allows usage of modern ML packages for performance users**
- **Provide full and straightforward separation of training, testing and application**
    - **Packages which are not simple to integrate with TMVA can then be trained externally and the results can be applied through TMVA**

Modernising TMVA:

- **Flexibility**
    - **The code should be made more modular, such that adding interfaces is straightforward**
        - **Significant progress has already been made by the RTMVA group**
    - **The core should be more flexible, allowing for decoupling for datasets/methods/variables in contrast to the current approach**
        - **The new re-design by the RTMVA group addresses this issue**

- **Computational Performance**
    - **The core code should be redesigned for improved computational performance**
        - **The C++ standard and programming techniques have been substantially updated in the past decade**
        - **Change matrix algebra to something like eigen which links to kernel level packages (ATLAS, BLAS)**
    - **Dataset I/O should be revisited**
        - **Dataset sizes are increasing and it is not always feasible to hold everything in memory. Many methods exist for optimized handling of I/O where only relevant parts of the dataset are held in memory**

- **Latest ML improvements:**
    - **Natively implementing latest improvements in ML in TMVA is a long and complicated process (akin to re-inventing the wheel) with potential pitfalls/bugs**
    - **Easy interfaces to the most powerful methods is a more desirable road to take with possible exceptions for significant game-changers. This is the approach chosen for the new RMVA (including PyMVA) interfaces in TMVA**
    - **The TMVA interface should allow for the use of more advanced ML algorithms for performance studies, high precision measurements, etc**
        - **R and python interfaces, with additional support libraries (scikit-learn, pandas etc) will cover the majority of the ML community, and thus are good starting points. The RMVA group has already done a great job here and is investigating further python/library support**
        - **A fully flexible interface for arbitrary language wrappers would be very useful, and should be easier after the R and python interfaces**

- **Desired Features:**
    - **Cross-validation**
        - **Standard in ML**
        - **New redesign by the RTMVA team allows easy implementation due to feature/method/dataset decoupling**
    - **Additional information for Analyzer:**
        - **Variable importance, accurate feature ranking**
            - **FAST algorithm for feature importance currently being integrated by the RTMVA team with the new redesign.**
    - **Parallelization**
        - **Many places where it applies, the RTMVA team is currently working on a general prototype**
        - **Thread safety for multi-threading (important for production)**
    - **GPU support for the most computationally intensive algorithms**
    - **Alternative input file types to more easily work with ML community (example: HDF5)**
    - **A high-statistics sample for testing purposes: the current sample within TMVA is not adequate for studying modern algorithm performance**
    - **Ability to output a standalone c++ code/executable without additional dependencies**
    - **Expert users should be able to pause and resume training after tweaking hyperparameters as is done in the ML community**
    - **Make it easier for the ML community to contribute directly to TMVA such as through a GitHub repository which is open to pull requests**

How a TMVA redesign affects the three use cases:

- **All users: improved computational performance and dataset flexibility**
- **Simple users: provides access to modern ML algorithms for additional power**
- **Performance users: provides access to cutting-edge ML algorithms through interfaces**
- **Potential developers: improved modularity makes it easier to contribute, interfaces make it easier to try new things, lots of areas for interested parties to contribute**
- **ML experts working with HEP: facilitates interactions between HEP and ML**
    - **Easier to re-import ML results into HEP, increasing the benefit of ML challenges and reducing the overhead of exploiting new ML techniques**
    - **Easier for ML community to work with the software they are familiar with and which may be better optimized for a given problem (in a way we did not consider); if we place restrictions on how they approach problems, this may become a limitation**

References for work done by the RTMVA group (Lorenzo+Sergei+students):

- **TMVA restructuring for modularity: http://oproject.org/TMVA**
- **RMVA interface: http://oproject.org/RMVA**
- **PyMVA (scikit-learn) interface: http://oproject.org/PyMVA**