Machine Learning Developments in ROOT

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 072046 doi:10.1088/1742-6596/898/7/072046

Machine Learning Developments in ROOT

A Bagoly[1],A Bevan[2], A Carnes[3], S V Gleyzer[3], L Moneta[4],A Moudgil[5],S Pfreundschuh[6], T Stevenson[2], S Wunsch[7] and O Zapata[8]

[1] Eötvös Lornd University
[2] Queen Mary University of London
[3] University of Florida
[4] CERN
[5] IIT Hyderabad
[6] Chalmers Institute of Technology
[7] Karlsruhe Institute of Technology
[8] University of Antioquia and Metropolitan Institute of Technology

E-mail: Sergei.Gleyzer@cern.ch, Lorenzo.Moneta@cern.ch, Omar.Zapata@cern.ch

Abstract. ROOT is a software framework for large-scale data analysis that provides basic and advanced statistical methods used by high-energy physics experiments. It includes machine learning tools from the ROOT-integrated Toolkit for Multivariate Analysis (TMVA). We present several recent developments in TMVA, including a new modular design, new algorithms for pre-processing, cross-validation, hyperparameter-tuning, deep-learning and interfaces to other machine-learning software packages. TMVA is additionally integrated with Jupyter, making it accessible with a browser.

1. Introduction

ROOT is an object-oriented framework that provides statistical methods, visualization and storage libraries for data analysis of high-energy physics (HEP) experiments, such as the Large Hadron Collider (LHC) in Geneva, Switzerland [1]. Although originally designed for HEP applications, ROOT is also widely used in other scientific fields outside of particle physics.

Today, machine learning is at the core of many particle physics analyses, such as searches for new physics and precision Standard Model studies, including identifying rare decays of the newly discovered Higgs boson. The ROOT framework provides machine learning tools with the Toolkit for Multivariate Analysis (TMVA) [2],that contains many machine learning algorithms. Here are some of the popular ones:

- Fisher and Linear Discriminants (LD)
- Boosted Decision Trees (BDT)
- Decision Rule Ensembles
- k-Nearest Neighbor Classifier (KNN)
- Artificial Neural Networks (ANN)
- Deep Learning Neural Networks (DNN)
- Support Vector Machines (SVM)

TMVA provides a way to compare the performance of these algorithms on the same dataset, useful in choosing the optimal algorithm for a particular analysis task. TMVA provides implementations of these methods for both classification and regression. Recently, TMVA has undergone a significant upgrade targeting greater flexibility, modular design, new features and interfaces.

2. New Algorithms and Features

Several high-level algorithms and libraries have been added: deep neural networks, k-fold cross-validation and hyper-parameter tuning. Additionally, TMVA is integrated with Jupyter [3], allowing for interactive execution in a browser. In what follows, the new functionality and features of TMVA are described.

2.1. Pre-processing

Pre-processing algorithms in TMVA transform variable input to a representation that can be effectively exploited by machine learning tasks (Figure 1). Existing pre-processing classes have been extended with VarTransformHandler class that implements algorithms such as Hessian Local Linear Embedding (HLLE) [4].

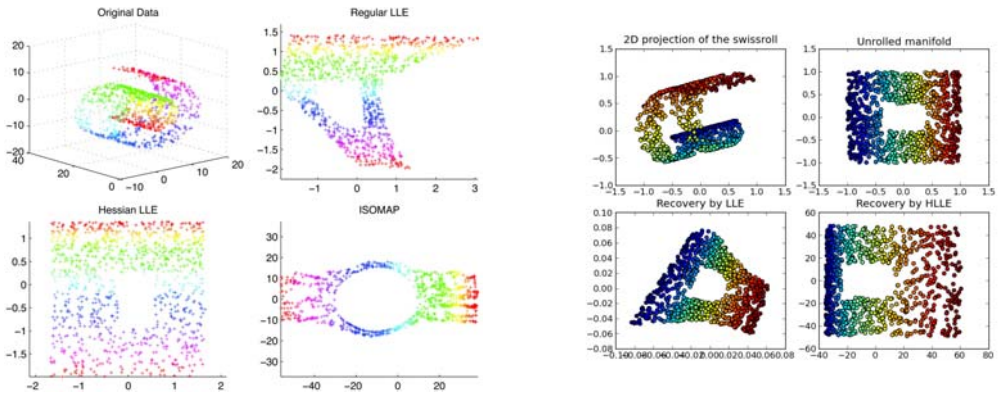

Figure 1. sample input data before (left) after HLLE transformation (right)

2.2. Deep Learning

A deep neural network is an artificial neural network with several hidden layers and a large number of neurons in each layer. Recent developments in machine learning have shown that these networks are capable of learning complex, non-linear relationships when trained on a sufficiently large dataset.

A new deep neural network (DNN) was implemented in TMVA, extending the existing artificial neural network Multi-Layer Perceptron (MLP) library. The new DNN implementation provides a deep learning library for efficient training on modern multi-core and GPU architectures.

As common for deep neural networks, DNN implementation uses the *stochastic batch gradient descent* method to train the network. In each training step the weights $W_{i,j}^k$ and bias terms θ_i^k of a given layer k are updated using

$$W_{i,j}^k \to W_{i,j}^k - \alpha \frac{\partial J(\mathbf{x}_b, \mathbf{y}_b)}{\partial W_{i,j}^k} \tag{1}$$

$$\theta_i^k \to \theta_i^k - \alpha \frac{\partial J(\mathbf{x}_b, \mathbf{y}_b)}{\partial \theta_i^k} \tag{2}$$

$J(\mathbf{x}_b, \mathbf{y}_b)$ is the value of the loss function corresponding to the randomly chosen input batch $\mathbf{x}_b$ and expected output $\mathbf{y}_b$. If regularization is applied, the loss may also contain contributions from the weight terms $W_{i,j}^k$ of each layer. This implementation also supports training with momentum. In this case training updates take the form:

$$\Delta W_{i,j}^k \to p\Delta W_{i,j}^k + (1.0 - p)\frac{\partial J(\mathbf{x}_b, \mathbf{y}_b)}{\partial W_{i,j}^k} \tag{3}$$

$$W_{i,j}^k \to W_{i,j}^k - \alpha \Delta W_{i,j}^k \tag{4}$$

$$\Delta \theta_i^k \to p\Delta \theta_i^k + (1.0 - p)\frac{\partial J(\mathbf{x}_b, \mathbf{y}_b)}{\partial \theta_i^k} \tag{5}$$

$$\theta_i^k \to \theta_i^k - \alpha \Delta \theta_i^k \tag{6}$$

For $p = 0$ the standard stochastic batch gradient descent method is obtained. The training batch $(\mathbf{x}_b, \mathbf{y}_b)$ is chosen randomly and without replacement from the set of training samples.

The new implementation provides several backends. The standard backend is based on multi-core CPU architecture and will work on any platform where ROOT is installed. It uses multi-threading to perform the training in parallel and requires a multi-threaded BLAS implementation and the Intel TBB library [5]. The new GPU backends can be used to train on CUDA and OpenCL-capable GPU architectures.

Numerical throughput and classification performance of the new deep learning library was evaluated on the Higgs dataset [6]. This dataset contains both low-level (kinematical) and high-level (domain knowledge inspired) features. As figure 2 shows, the new library shows higher classification performance compared to boosted decision trees, trained on the same dataset. Additionally, the new library is able to extract useful features directly from low-level features, as illustrated by Figure 2. Furthermore, the TMVA-Cuda backend exhibits superior numerical throughput performance compared to TMVA-OpenCL, TMVA-CPU and Theano [7] deep learning implementations (Figure 3).

2.3. Regression

Previously, a single hard-coded loss function was used for boosted decision trees. A new loss function class, with a number of new loss function options, was added. These options include least-squares and absolute deviation, in addition to the default Huber loss function [8]. Figure 4 left shows an example of using different loss functions on the default input dataset in TMVA. Figure 4 right shows an example application of the deep learning (DNN) library on the same dataset. One can observe the benefit of adding depth to the neural network for this regression task.

2.4. Cross Validation

K-fold cross-validation is a machine-learning model evaluation technique, relevant to model generalization to unseen data. During k-fold cross-validation, the dataset is partitioned into k folds

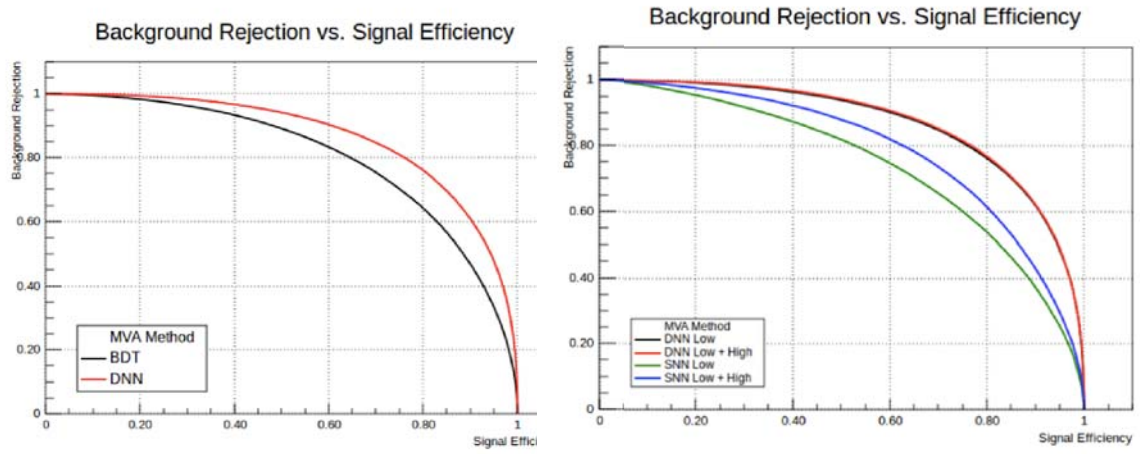

Figure 2. Left: comparison of deep neural network (DNN) and boosted decision tree classifier performance on the Higgs dataset. Right: comparison of shallow and deep neural network classifier performance applied to different subsets of variables (low-level and full set)

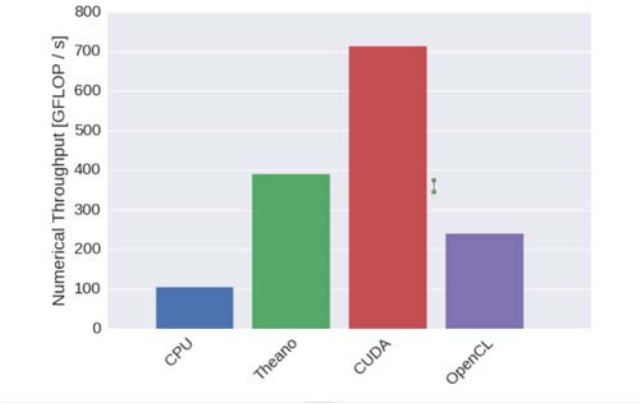

Figure 3. Numerical throughput comparison of TMVA-CPU, TMVA-OpenCL, TMVA-CUDA and Theano deep learning libraries tested on the same Nvidia Tesla K20 GPU.

or partitions. During one cross-validation round, k-1 folds are used for model training, and the remaining one for model testing. Several rounds of cross-validation are performed with different partitions and model performance results are averaged. One advantage of cross-validation over a simple split into training and testing set, is the use of the full dataset to validate the model. Performing cross-validation is known to reduce over-fitting of the data, leading to a more accurate estimate of the performance of the machine-learning model on unseen data [9].

Cross-validation in TMVA is done with a standalone CrossValidation class. Figure 5 illustrates five receiver-operating characteristic (ROC) curves for each cross-validation fold for a basic TMVA example. This example has four random variables with gaussian distributions plus several derived variables after applying basic mathematical operations to these variables.

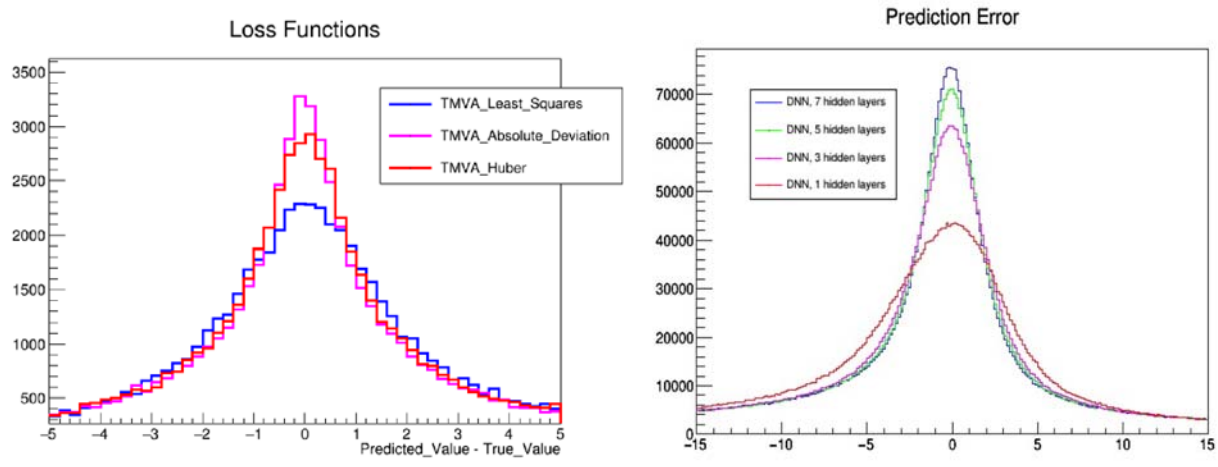

Figure 4. left: Different choices of regression loss functions: least-squares, absolute-deviation and Huber on a sample dataset right: Effect of varying the depth of deep learning neural network

2.5. Hyperparameter Tuning

Support for hyperparameter tuning has been added in TMVA for the following algorithms: boosted decision trees and support vector machines. It allows an automatic search through hyperparameter space to find the optimal classifier, and relies on k-fold cross-validation to accurately evaluate classifier performance.

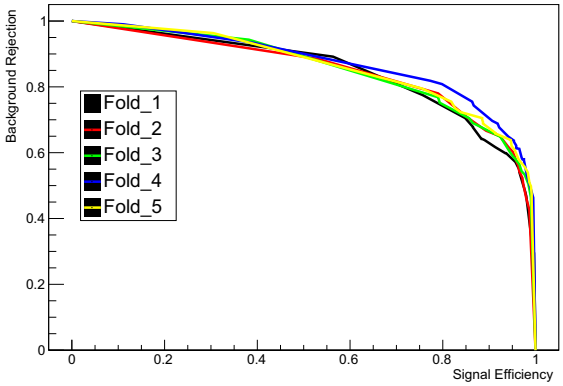

Figure 5. Cross validation ROC curves.

2.6. TMVA and Jupyter Notebooks

Another new feature in TMVA is the integration of TMVA and Jupyter notebooks. Jupyter is a web application that combines live code, rich text, links and formula in a user-friendly format. All the previous functionality of the TMVA Graphical User Interface (GUI) is available in Jupyter notebooks, requiring only access to a browser. Additional interactive features have been added: neural network and boosted decision tree visualizations 6, ability to interactively

monitor training progress and pause training 7, and visual neural network builder. Furthermore, the output of TMVA was improved with html formatting.

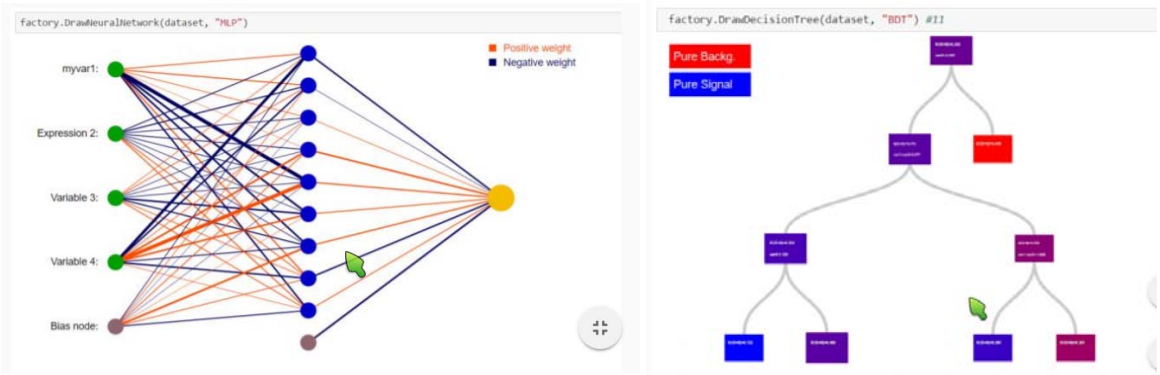

Figure 6. New classifier visualizations for neural networks (left) and boosted decision trees (right)

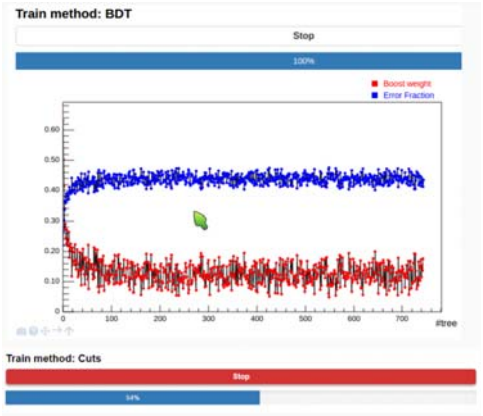

Figure 7. Interactive training functionality allowing the monitoring of errors during training and ability to stop training on demand

3. TMVA interfaces to external machine learning tools

Other useful new functionality added to TMVA are the interfaces to external machine-learning tools in R and Python languages.

3.1. RMVA Interface

RMVA is a set of TMVA plugins that allows the use of machine-learning methods available in R directly from TMVA. The goal of RMVA is to allow direct comparison of R-based algorithms with existing tools in TMVA for a given problem. Currently, the following machine-learning packages in R are supported:

- Decision trees and rule-based models (C50) [10].
- Stuttgart Neural Networks in R (SNNS)[11].
- Support Vector Machines in R (e1071)[12].

- eXtreme Gradient Boost (xgboost) An optimized general purpose gradient boosting library[13].

3.2. Python with TMVA (PyMVA)

PyMVA is a set of TMVA plugins based on Python API that allows direct use of machine-learning methods written in Python from within TMVA. The following Python based methods from the Scikit-learn software package [14] are currently available in TMVA:

- Random Forest (PyRandomForest)
- Gradient Boosted Regression Trees (PyGTB)
- Adaptive Boosting (PyAdaBoost)

3.3. PyKeras

An interface to Keras [15] has been added to TMVA. Keras is a high-level deep learning library, written in python, that works with Theano [7] and Tensorflow [16] deep learning frameworks. The new interface permits running Keras from within TMVA.

3.4. Conclusions

Machine learning tools in ROOT have undergone a significant upgrade. In particular, TMVA has a new design targeting greater flexibility and modularity, new features such as deep learning neural networks, cross-validation, hyper parameter tuning and interfaces to R and python-based machine-learning tools. In addition, TMVA is available in Jupyter notebooks, making it accessible in a web browser.

Acknowledgments

The work of O. A. Zapata was partially supported by Sostenibilidad-UdeA, UdeA/CODI grant IN361CE and COLCIENCIAS grant 111565842691.

References

[1] Antcheva I, Ballintijn M, Bellenot B, Biskup M, Brun R, Buncic N, Canal P, Casadei D, Couet O, Fine V, Franco L, Ganis G, Gheata A, Maline D G, Goto M, Iwaszkiewicz J, Kreshuk A, Segura D M, Maunder R, Moneta L, Naumann A, Offermann E, Onuchin V, Panacek S, Rademakers F, Russo P and Tadel M 2009 *Computer Physics Communications* **180** 2499 – 2512 ISSN 0010-4655 40 {YEARS} {OF} CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures URL `http://www.sciencedirect.com/science/article/pii/S0010465509002550`

[2] Hoecker A, Speckmayer P, Stelzer J, Therhaag J, von Toerne E and Voss H 2007 (*Preprint* `physics/0703039`)

[3] Pérez F and Granger B E 2007 *Computing in Science and Engineering* **9** 21–29 ISSN 1521-9615 URL `http://ipython.org`

[4] Donoho D L and Grimes C 2003 *Proceedings of the National Academy of Sciences* **100** 5591–5596

[5] Pheatt C 2008 *J. Comput. Sci. Coll.* **23** 298–298 ISSN 1937-4771 URL `http://dl.acm.org/citation.cfm?id=1352079.1352134`

[6] Lichman M 2013 UCI machine learning repository URL `http://archive.ics.uci.edu/ml`

[7] et al R A R (Theano Development Team) 2016 *arXiv e-prints* **abs/1605.02688** URL `http://arxiv.org/abs/1605.02688`

[8] Huber P J 1964 *Ann. Math. Statist.* **35** 73–101 URL `http://dx.doi.org/10.1214/aoms/1177703732`

[9] Arlot S, Celisse A *et al.* 2010 *Statistics surveys* **4** 40–79

[10] Kuhn M, Weston S, Coulter N and code for C50 by R Quinlan M C C 2015 *C50: C5.0 Decision Trees and Rule-Based Models* r package version 0.1.0-24 URL `https://CRAN.R-project.org/package=C50`

[11] Bergmeir C and Benítez J M 2012 *Journal of Statistical Software* **46** 1–26 URL `http://www.jstatsoft.org/v46/i07/`

[12] Meyer D, Dimitriadou E, Hornik K, Weingessel A and Leisch F 2015 *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien* r package version 1.6-7 URL `https://CRAN.R-project.org/package=e1071`
[13] Chen T and He T 2015 *R package version 0.4-2*
[14] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V *et al.* 2011 *Journal of Machine Learning Research* **12** 2825–2830
[15] Chollet F 2015 keras `https://github.com/fchollet/keras`
[16] et al M A 2015 TensorFlow: Large-scale machine learning on heterogeneous systems software available from tensorflow.org URL `http://tensorflow.org/`