

Parallelization of ROOT Machine Learning Methods

Pourya Vakilipourtakalou

Supervisors: Prof. Lorenzo Moneta (CERN)

Prof. Sergei Gleyzer (University of Florida)

Summer 2016

Introduction

Today computation is an inseparable part of scientific research. Specially in Particle Physics when there is a classification problem like discrimination of Signals from Backgrounds originating from the collisions of particles. On the other hand, Monte Carlo simulations can be used in order to generate a known data set of Signals and Backgrounds based on theoretical physics. The aim of Machine Learning is to train some algorithms on known data set and then apply these trained algorithms to the unknown data sets. However, the most common framework for data analysis in Particle Physics is ROOT. In order to use Machine Learning methods, a Toolkit for Multivariate Data Analysis (TMVA) has been added to ROOT.

The major consideration in this report is the parallelization of some TMVA methods, specially Cross-Validation and BDT.

Cross-Validation

Cross-Validation is a method in Machine Learning that evaluates the performance of a method. In this method the complete known dataset is split into number of folds (which is an arbitrary number set by the user). In the first iteration one of those parts is chosen as a test set and the other $k-1$ parts as training set. The method is trained on the training set and tested on the test set and as long as the dataset is known in the testing procedure the accuracy of the method can be assessed. In the next iteration the second part plays the role of a test set and the method is trained on the other $k-1$ parts and so on (Figure 1).

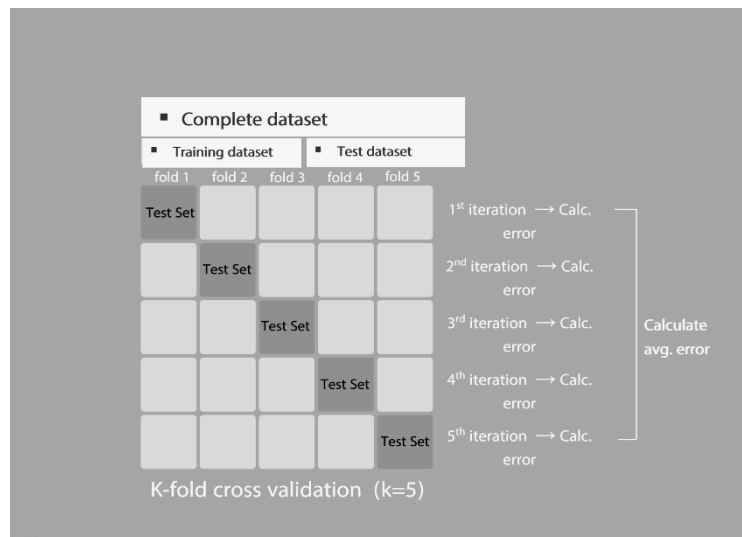


Figure 1. Cross-Validation

ROC Curves

ROC stands for Receiver Operating Characteristics. It is a graphical plot that gives a visualization of how the performance of a classifier is. Signal Efficiency is plotted vs Background Rejection in this curve. In the figure 2 the best and the worst classifiers are visualized. As you can see the bigger the area under the curve (AUC) is, the better the classifier has behaved.

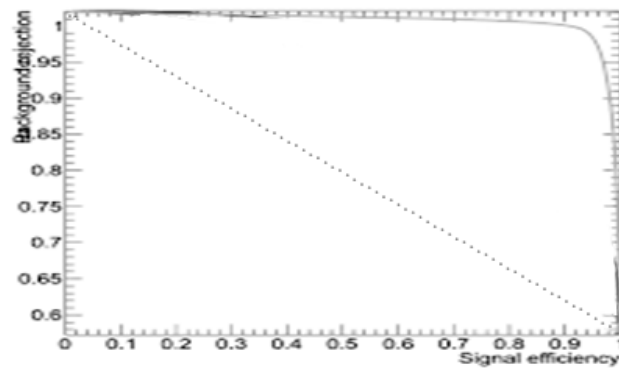


Figure 2. The dotted line shows the worst classifier and the curve shows the best classifier

To make Cross-Validation method in TMVA a little more user friendly, it is better to plot the ROC curve for each iteration in the Cross-Validation method.

This is done by a new added method named PlotROC() (Figure 3).

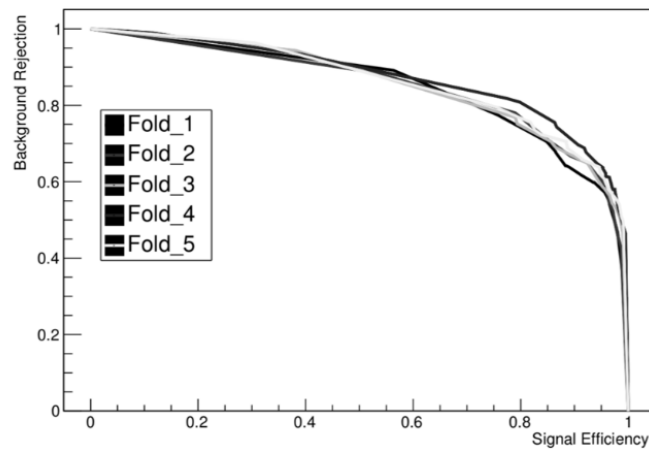


Figure 3. The ROC curve for each iteration

Parallelization Classes in ROOT

Parallelization literally means to run some calculations simultaneously. When the same process is executed lots of times, in order to get the speed up it is better to register some threads (multithreading) or processors (multiprocessing) to do the same process in parallel. Parallelization can be done in two ways **MultiThreading** and **MultiProcessing**.

MultiThreading: In multithreading some threads are registered and they do a process at the same time and at last all of them will report the result they achieved. In multithreading, threads share the memory so if an error happens during the execution of one thread it will directly affect the execution of other threads. So it makes it a little hard to work with threads.

MultiProcessing : In multiprocessing, processors are registered to do the same process but in this case each processor has its own memory. Actually each processor makes a copy of the whole memory for itself and if an error happens during the execution of one processor, it won't affect the execution of the other processors. As a result, it is really easier to work with the processors than threads. But the problem in some cases is that multiprocessing is slower than the multithreading because each processor should make a copy for itself.

As can be seen in Figure 1, in the serialized version of Cross-Validation the same process is done 5 times but for each iteration a thread (or processor) can be registered in order to make this method parallelized.

ROOT has some classes for multiprocessing and multithreading which are called TProcPool and ThreadPool, respectively. The procedure which both of them are executed is exactly the same. We create a pool of threads (or processors) and then We put the part of the code that is needed to be parallelized in a Lambda Function and by using Map() method in those classes we call the function as much as we want and automatically the execution would be done in parallel.

After the parallelization of Cross-Validation using TProcPool we got the speed up.

The execution is done on the Small Higgs Data Set with 1,000,000 events (Figure 4).

As you can see by increasing the number of processors there is a good amount of speed up.

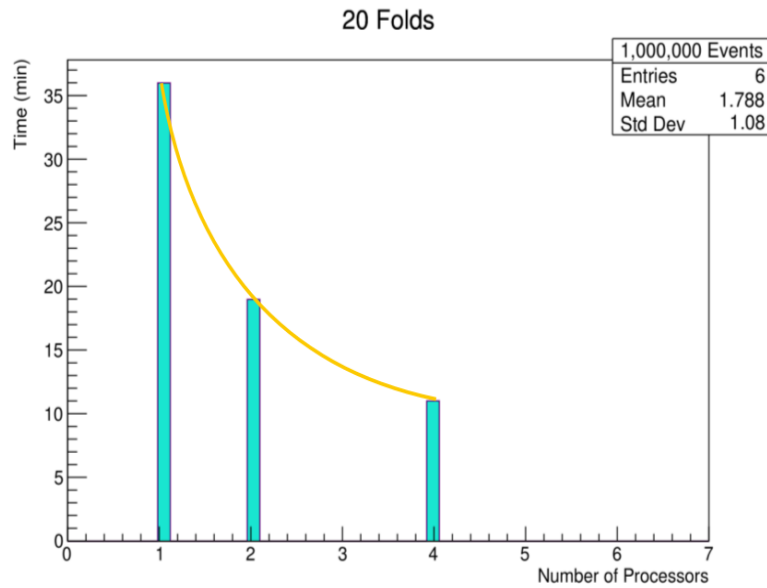


Figure 4. As the number of processors increases the time needed for Cross-Validation method decreases

Decision Trees

Another method for classification is Decision Tree, in which for each event a tree is built. A tree has some nodes and in each node there is a kind of question. Consequently, given an event, it is possible to follow the tree and classify that event as Signal or Background, Figure 5.

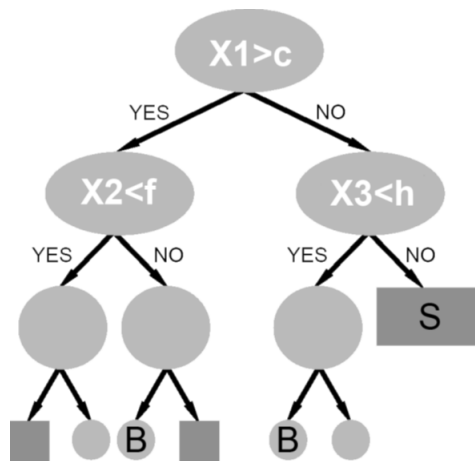


Figure 5. A decision tree. c, f, h are known numbers

There are 11,000,000 events in the Higgs data set, and some threads (or processors) can be registered in order to split the whole data set and build the tree for the events. This is the parallelized version of Decision Tree, Table 1.

1 thread	133 seconds
2 threads	115 seconds
4 threads	107 seconds

Table 1. Speed up in training the method as we increase the number of threads

Conclusion

Parallelization is one of the ways that we can use to speed up some processes. Some methods like Cross-Validation and BDT are parallelizable. So by parallelizing them we can see the speed up. On the other hand, in order to make TMVA more user friendly, it is possible to add some methods which give visualization. PlotROC() is a new method that illustrates ROC curves for each iteration of Cross-Validation.