Nuclear Instruments and Methods in Physics Research A 672 (2012) 6–12

Contents lists available at SciVerse ScienceDirect

Nuclear Instruments and Methods in Physics Research A

journal homepage: www.elsevier.com/locate/nima

Comparison of the CPU and memory performance of StatPatternRecognitions (SPR) and Toolkit for MultiVariate Analysis (TMVA)

G. Palombo

California Institute of Technology, United States

ARTICLE INFO

Article history:
Received 19 June 2011
Received in revised form
5 December 2011
Accepted 5 December 2011
Available online 21 December 2011

Keywords:
StatPatternRecognition
Toolkit for MultiVariate Analysis
Machine learning
Data analysis
Computational physics

ABSTRACT

High Energy Physics data sets are often characterized by a huge number of events. Therefore, it is extremely important to use statistical packages able to efficiently analyze these unprecedented amounts of data. We compare the performance of the statistical packages StatPatternRecognition (SPR) and Toolkit for MultiVariate Analysis (TMVA). We focus on how CPU time and memory usage of the learning process scale versus data set size. As classifiers, we consider Random Forests, Boosted Decision Trees and Neural Networks only, each with specific settings. For our tests, we employ a data set widely used in the machine learning community, "Threenorm" data set, as well as data tailored for testing various edge cases. For each data set, we constantly increase its size and check CPU time and memory needed to build the classifiers implemented in SPR and TMVA. We show that SPR is often significantly faster and consumes significantly less memory. For example, the SPR implementation of Random Forest is by an order of magnitude faster and consumes an order of magnitude less memory than TMVA on Threenorm data.

1. Introduction

In modern High Energy Physics (HEP) analyses, the use of machine learning techniques has become increasingly common. Machine learning classifiers are used to separate signal events from unwanted background [1]. Several HEP experiments are characterized by an extremely large number of events. Therefore, it is crucial to use statistical packages able to efficiently analyze data sets described by million events or even larger. Statistical packages widely used among statisticians, such as R [2] or Weka [3], often implement many statistical techniques that would be extremely useful in HEP analyses. However, they are not always easily extendable for HEP needs and the community has traditionally shown a strong preference for home grown statistical tools [4].

The two statistical software most used in the HEP community are StatPatternRecognition (SPR) [5] and Toolkit for MultiVariate Analysis (TMVA) [6]. Both packages have been developed within the HEP community and are targeted to HEP statistical analyses. In this work, we compare how CPU time and memory usage scale with data set size in SPR and TMVA.

A detailed description of the machine learning techniques described in this work can be found in Ref. [7].

2. SPR and TMVA

SPR is an open source standalone C++ package that can be run within ROOT [8] or from the command line. It implements linear and quadratic discriminant analysis [9], logistic regression [7], binary decision splits, bump hunter [10], two flavors of decision trees [11], a feedforward backpropagation neural net with a logistic activation function [12], several flavors of boosting [13] including the arc-x4 algorithm [14], bagging [15] and random forest [16]. The package also includes two multiclass methods that allow to use any binary classifier for a multiclass classification problem [17]. Moreover, it implements algorithms to boost or bag any sequence of classifiers as well as combine classifiers trained on subsets of input variables.

TMVA is an open source project integrated with ROOT. It implements the following multivariate techniques: rectangular cuts, projective and multidimensional likelihood estimators [18], *k*-Nearest Neighbor [19], Fisher and H-matrix discriminants, linear and function discriminant analysis [20], multilayer perceptron neural networks, support vector machine [21], boosted decision trees, random forest, and rulefit [22]. It also allows to boost any classifier.

Both packages implement similar techniques to pre-process the data, such as normalization, principal component analysis, correlations, and cuts. Variable importance in TMVA is estimated for neural networks by calculating the weight of neural network links and for decision trees by calculating the improvement in the

E-mail addresses: g.palombo@campus.unimib.it, giulio.palombo@gmail.com, palombo@caltech.edu, giulio@justanswer.com

0168-9002/$ - see front matter © 2012 Published by Elsevier B.V.
doi:10.1016/j.nima.2011.12.044

classifier performance given by the splits on each variable. In addition to these techniques, SPR also presents other variable importance algorithms that work with any classifier: random permutation of the class label, "Add N Remove R", and interactions.

Finally, SPR also includes cross-validation techniques, allows to choose among 10 Figures of Merit to optimize the classifier and test its performance, and implements Friedman's machine learning-based Goodness of Fit test [23].

For a more comprehensive description of SPR and TMVA, please see readme and user's guides included in package distributions [5,6].

Although a significant number of classifiers are included in both packages, their implementation can differ, leading to different results [4]. Different implementation implies that, for instance, the best parameters for SPR Boosted Decision Trees (BDT) are not necessarily the best parameters for TMVA BDT.

3. Classifiers and data sets

For our tests, we use release 08.02.00 of SPR and 4.0.3 of TMVA. CPU time reported in this work refers to elapsed real time. The CPU time variation, for a given experiment, is at most at the order of seconds.

To estimate memory usage, we consider Resident Set Size (RSS) [24]. The amount of consumed memory on a given computer by a given executable with given input parameters is a deterministic number. That is, RSS of a given experiment does not vary, unless the memory usage comes close to the total RAM available at the node. Memory usage traces are collected by running the top utility at intervals proportional to the expected length of the process. Maximum value for RSS is reported here. Both packages quickly reach a value close to the maximum and then their memory usage becomes almost constant until the end of the process.

The executables are run on a dedicated machine with no other major task running simultaneously. The machine runs CentOS Linux 5.4 and has 8 Intel(R) Xeon(R) @ 2.33 GHz processors with 8 GB of RAM.

As classifiers, the attention is restricted to RF, BDT, and Neural Networks (NN).

With regards to the most important parameters, for RF we choose 50 trees, at least 5 events per leaf, and each split chosen among $D/2$ variables randomly selected at each node, where D is the data set dimensionality. For BDT, we choose 100 cycles and 10% of the events as minimum number of events per leaf. For NN, layer structure is $D:D:D/2:1$. As suggested by TMVA user's guide [6], TMVA neural network model is the MultiLayer Perceptron (MLP). All other parameters are set so that the packages would perform the same task for each classifier.

The goal of this analysis is to check how CPU time and memory usage scale to data set size. The chosen parameter values are not necessarily the best ones for each data set in terms of classifier predictive power. Finding the best classifier configuration for each data set would make hard to exactly keep trace of CPU time and memory usage with respect to the data set size. However, the chosen parameters are likely to be reasonably close to the best possible configuration for every classifier [25,26].

TMVA's variable transformation options are disabled in order to achieve a major similarity between the tasks executed by the packages. TMVA's variable transformation options, which are on by default in the code example, include decorrelation, principal component decomposition, and gaussianization. They consume a very large amount of memory. For instance, those variable transformation options take up to 5 or 6 times the maximum amount of memory needed to build BDT. The bigger is the data set, the larger is the ratio between the memory needed for variable transformation and for training the BDT classifiers. SPR does not need such an adjustment since uses different executables to perform different tasks.

As a benchmark data set, we choose the "Threenorm" data set introduced by Breiman in Ref. [26]. Given $a=2/(20)^{1/2}$, one class is drawn from a unit multivariate normal with mean $(a,-a,a,-a,\ldots,a)$. The other class is drawn with equal probability from a unit multivariate normal with mean $(a,a,\ldots,a)$ and from a unit multivariate normal with mean $(-a,-a,\ldots,-a)$. This data set is considered a difficult one for classification problems [26] and has been widely used for comparison of machine learning algorithms [27].

As an additional test, we train our classifiers on edge cases. That is, edge cases are data sets described by a single variable which follows an unusual distribution. Example of edge cases are a random noise variable, a variable for which all events except for one have the same value, and a variable for which 50% of events are useless and 50% have discriminant power. With respect to edge cases, we only test the behavior of RF and BDT. Results refer to the training part only. We start with small data sets and gradually increase dimensionality and number of events. We start with 2×10^4 events and 20 variables. For BDT, we gradually increase dimensionality up to 100 variables and size up to 4×10^6 events. For RF, being slower given the optimization parameters, we increase up to 100 variables and 10^6 events; and for NN, the slowest among the three, up to 70 variables and 10^6 events. For edge cases, we only increase the number of events as described above.

4. Test on "Threenorm" data set

We test learning time and memory usage on Threenorm data set for BDT, RF, and NN. Fig. 1 shows a comparison of CPU time needed by SPR and TMVA to build BDT with 20 variables (left) and 100 variables (right). CPU time of tree-based classifiers is expected to grow linearly versus $N\log N$, where N is the number of events. The linearity should hold as long as the data set size does not approach the amount of available RAM. SPR BDT are constantly faster than TMVA BDT. For 20 variables, TMVA CPU time is about two times larger. For 100 variables, TMVA needs up to three times more time to build the BDT. We also check configurations with 40 and 70 variables. Results are as expected. That is, they are in between the 20 and 100 variables outputs.

Fig. 2 shows SPR and TMVA memory usage for 20 variables and 100 variables. TMVA consumes more memory in both configurations.

The trees built by SPR and TMVA have a very similar structure, with their depth being equal to either 2 or 3. This depends on the optimization parameters chosen, i.e. 10% of events as minimum number of events per leaf. Therefore, this analysis gives a useful indication of time and memory difference to build classifiers based on small trees.

We repeat the same test using RF classifiers. CPU time results are shown in Fig. 3. SPR is again faster than TMVA. The difference between SPR and TMVA is larger than for the BDT test. RF trees are larger than BDT trees. Therefore, bigger trees lead to a larger difference in terms of performance between SPR and TMVA. Furthermore, the ratio between TMVA and SPR CPU time increases as the data set size gets larger. SPR is about 5 times faster when the number of events is less than 400 K, and about 10 times faster when the data set is described by 1 million events. That is, SPR CPU time increases linearly with respect to $N\log N$ and TMVA CPU time increases more than linearly.