Experimenting with rule induction algorithms in HEP data analysis

N. Stepanov

Helsinki Institute of Physics, Helsinki, Finland *

Abstract. The simple prototype rule induction system for the HEP analysis applications is presented. The feasibility of the approach is demonstrated on the benchmark classification task taken from the LHC physics.

INTRODUCTION AND MOTIVATION

It is a common practise now to use the machine learning methods in high energy physics (HEP) multi-variable data analysis. In particular, neural network systems (NN) are proven to be useful in numerous classification, prediction and regression applications. NN approach is certainly very attractive but has its doubters. Most of them are some how connected with the fact that NN is a black box system. The black box develops its own concept representation that to be used for concept recognition purposes, however, this internal description cannot be easily interpreted by the user and provides neither insight nor explanation of the recognition process. There are knowledge-oriented methods capable of developing descriptions understandable to the human being directly manipulating symbolic structures, e.g. rule induction algorithms (RIA). The purpose of this paper is to present a simple framework for a rule induction chain and to demonstrate the usefulness of the approach on the benchmark classification task taken from the LHC physics.

The rules we are going to use incorporate an "attributional" logic which is roughly equivalent to the zero-order logic (1): **rule := if<condition> then predict <class>** , where **<condition>** is a conjunct of attribute tests. In this scheme the concept to learn (e.g. "Higgs event") is characterised by values of some predefined set of attributes, such as "event E_t^{miss}" or "number of the isolated leptons in event" and each concept example is a vector of attribute values of the finite fixed length. The goal of our rule induction chain is to represent the target concept by the **decision list**, i.e. ordered rule set requiring the rule evaluation in a sequential order. Three attribute types are introduced: **symbolic** attribute (takes values from the finite set) with the only allowed test on the attribute value: $attr\_val = Val$; **integer** attribute (takes values from N) with possible tests: $attr\_val = Val; Val_1 < attr\_val \leq Val_2; Val_1 < attr\_val$ and **real** attribute (takes values from R) with tests: $attr\_val = Val; Val_1 < attr\_val < Val_2; Val_1 < attr\_val$.

RULE INDUCTION CHAIN

The rule induction system we are using contains three basic modules.

• Rule engine. It is the main module responsible for the raw rule set generation given the data. It could be any RIA discovered so far. At the moment we implemented and tested just two RIA's representing two main algorithm families: 'divide-and-conquer' and 'separate-and-conquer' (2). Our first RIA is inspired by the Quinlan's C4.5rule (3). Strictly speaking, it's not the RIA but the top down decision tree induction algorithm (TDIDT). However, any tree can be easily converted to the equivalent rule set. The second one is really RIA using the top down rule induction technique invented primarily by Michalski for his AQ family (4). We tested both RIA's on the same task with similar results - both demonstrate about the same performance and have the same problems, in particular, both need intensive pruning.

• Pruning module. Any rule set generated has to be pruned to handle noisy data and to control the overfitting. There are two pitfalls of the general post-pruning algorithms. The first one is that they have bad time complexity, e.g. Reduced Error Pruning (REP) has a complexity $O(n^4)$ (5) while the rule generation itself is only $O(n^2 \times \log n)$. Second, the results are not guaranteed because the greedy hill-climbing strategy is used and some potentially useful rules can be lost. In our case these pitfalls become of particular importance since we are experimenting with the interactive system. Much more simple two-step post-pruning scheme is adopted here. First,

* on leave from ITEP, Moscow, Russia

each rule is simplified by the dropping and modifying the "overfitted" tests from the rule body. The entire rule can be also dropped if: it contains just the "bad" tests or it becomes equivalent to some previously simplified rule (or its trivial specification). Second, simplified rule set is tested using another "pruning" data sample. Some new rules have now bad purity and thus are dropped. The resulting rule set is ordered in descending order according to the rule coverage. This simple scheme is sufficient for our purposes. The price is the loss of generality and new subjective bias introduced. This is because the "overfitness" criteria is task specific.

• Interactive fine-tuning module. This module helps user to update the generated decision list iteratively, monitoring the performance in each step. It is allowed both to drop rule or to modify rule. The modification includes the possibilities to drop or modify the existing test and to add the new one. The target concept can be now more complicated than the target used at the rule induction stage (see example below).

CASE STUDY

The test target was to optimise the b-tagging performance for the particular channel: $pp \to t\bar{t}(H \to b\bar{b}) \to l, 4b, jj, E_t^{miss}$. As it was shown recently (6) this channel allows about the only opportunity to observe the SM Higgs via $b\bar{b}$ decay up to $M_H \sim 130$ GeV and to discover the lightest Higgs of the MSSM almost everywhere in $M_A, \tan\beta$ parameter space. However, it requires the ultimate detector performance, in particular, to suppress the backgrounds, four b-jets have to be identified per event. The main background is the $pp \to t\bar{t} + X$ itself.

Simulation details

The $t\bar{t}H$ ($M_H = 120$ GeV) and $t\bar{t} + X$ events were generated by PYTHIA (7). Only events with at least 6 jets with $E_t > 15$ GeV and isolated lepton with $P_t > 15$ GeV in η range $|\eta| < 2.4$ were selected. To learn the classifier, three jet classes were introduced: b-jets, c-jets and qg-jets. The jet classification was done using the parton level information. To avoid ambiguities, each heavy quark was allowed to classify the only closest jet.

To obtain the realistic b-tagging performance estimates, the detector response (in our case CMS detector) has to be simulated properly. The compromise approach was used. We applied the tuned fast MC for the Calorimeter system (8) to reconstruct jets but the detailed GEANT simulation for the Tracker and dedicated CMS pattern recognition packages for the track and vertex reconstruction (9, 10, 11).

Jet example structure

To b-tag jet one usually searches for high quality tracks with large transverse impact parameter (TIP) and (or) secondary vertices inside the jet cone. The proper quantities are track signed TIP significance defined as $SIG = sign(\cos(\vec{TIP} * \vec{P_t^{jet}})) \times TIP/err(TIP)$ and similarly defined secondary vertex significance SIV. Only "good" tracks satisfying the minimal quality criteria: $SIG > 2; P_t > 0.7$ GeV; $N_{hit} > 5; N_{pixelhit} > 1; \chi^2/ndf < 5$ and $TIP < 0.2$ cm were used for the analysis. In total, each example (jet) was characterised by 9 (correlated) attributes: 1) **integer jet** E_t (1: $E_t < 30$ GeV; 2: 30 - 60; 3: 60 - 120; 4: 120 - 200 and 5: $E_t > 200$ GeV); 2) **integer jet** η (1: $|\eta| < 0.7$ (barrel); 2: 0.7 - 1.2; 3: 1.2-1.6; 4: 1.6-2.0; 5: 2.0-2.4 ($\sim$ maximal CMS Tracker acceptance); 3) **integer NTR** (number of "good" tracks in a cone 0.4); 4) **real TSS** (total significance of associated tracks ΣSIG_i); 5) **real TPT** (ΣP_t^i); 6) **real MST** (effective mass of associated tracks); 7) **integer NSV** (number of secondary vertices associated with a jet); 8) **real TSV** (ΣSIV_i); 9) **real MSV** (effective mass of all tracks trapped in associated vertices)

Each jet example was also labeled with jet class ID.

Results

To estimate the system performance we introduce six parameters. First three monitor the overall b-tagging performance: **BE** (probability to b-tag b-jet); **QGI** (probability to b-tag qg-jet) and **CI** (the same for c-jets). At the rule generation and pruning stage the system was learned just to recognise b-jets. Three more parameters: **ES4** (fraction of "good" signal events 4-b-tagged); **EGL** (fraction of signal events 4 b-tagged) and **TTI** (fraction of $t\bar{t} + X$ events 4 b-tagged) were used at the interactive stage to tune the b-tagging rules for the particular $t\bar{t}H$ channel. Two different parameters for the signal efficiency monitoring were required since just ~ 0.53 selected $t\bar{t}H$ events with at least 6 jets actually contain at least 4 jets matched with b-quarks.

We made 3-fold cross-validation tests randomly dividing all simulated data on three subsets: 1000 ($t\bar{t}H$) + 1000 ($t\bar{t} + X$) events in the rule generation set, 2000 + 3000 in pruning and tuning one and 2000 + 3000 in the testing one. The best decision list generated was tested using the whole simulated event set. As a "straw man" we used a simple b-tagging rule learned by hand: a jet is b-tagged provided there are at least two good tracks with $SIG > 2$ and the reconstructed vertex in the jet cone (i.e. $NTR > 1 \vee NSV > 0$). However, it looks not so easy to beat the straw man performance: **BE = 0.569; QGI = 0.0194; CI = 0.1436; ES4 = 0.1446; EGL = 0.089; TTI = 0.00392**. Decision list performance (averaged over 3 tests) is indeed better: **BE = 0.623; QGI = 0.0198; CI = 0.163; ES4 = 0.2174; EGL = 0.134; TTI = 0.0032**.

The best decision list generated contains 12 rules: $NTR > 1 \vee TPT > 4 \vee TSV > 3$; $NTR > 2 \vee TPT > 8 \vee TSS > 8$; $NTR > 1 \vee TSV > 2 \vee MSS > 1$; $NTR > 1 \vee TPT > 15 \vee TSS > 20$; $NTR > 1 \vee TPT > 9 \vee TSS > 9 \vee E_t \leq 3 \vee \mid \eta \mid \leq 2$; $NTR > 1 \vee TPT > 5 \vee TSS > 7 \vee NSV > 0 \vee E_t \leq 2$; $NTR > 0 \vee TPT > 1 \vee TSV > 6 \vee NSV > 1 \vee E_t \leq 2$; $NTR > 1 \vee TPT > 5 \vee TSS > 7 \vee E_t = 1$; $NTR > 1 \vee TPT > 4 \vee MSS > 0.5 \vee E_t = 1 \vee \mid \eta \mid \leq 3$; $NTR = 1 \vee TSV > 6 \vee 0.8 < MSV < 3 \vee TPT > 3 \vee E_t \leq 3 \vee \mid \eta \mid \leq 2$; $NTR = 1 \vee TSV > 8 \vee TPT > 5 \vee TSS > 6 \vee E_t \leq 3 \vee \mid \eta \mid \leq 2$; $NTR = 1 \vee TSV > 5 \vee TPT > 2 \vee TSS > 6 \vee E_t \leq 4 \vee \mid \eta \mid \leq 2 \vee 1 < MSV < 5$

The structure of this decision list looks quite logical and understandable - first four general rules in the list cover most of examples, the rest suits to catch the low E_t and (or) small η exceptions. The truncated list containing just first 4 rules demonstrates excellent purity parameters and outperform the straw man in the efficiency: **BE = 0.584; QGI = 0.0160; CI = 0.141; ES4 = 0.1588; EGL = 0.0975; TTI = 0.00143**. The full list allows one to gain in efficiency by some impurity expenses: **BE = 0.628; QGI = 0.0204; CI = 0.167; ES4 = 0.22; EGL = 0.136; TTI = 0.00356**.

Unfortunately, the present $t\bar{t}$ statistics accumulated does not allow to estimate accurately the signal significance improvement, but preliminary rough estimates indicate that we are close to the level of 5σ signal significance for $L_{int} = 30fb^{-1}$ which is much better than all previous attempts. In near future we are planning to increase the background statistics in ten times. This will allow to monitor directly the $t\bar{t}H$ signal significance at the interactive stage.

SUMMARY

We presented the prototype framework for the rule induction system and demonstrated the feasibility of the approach on the challenging classification problem, namely, b-jet identification for the particular $pp \to t\bar{t}H$ physics channel. The rules set generated is simple but it easily outperforms the standard hand-written b-tagging algorithms developed for CMS Tracker so far (12). At the moment the system relies too heavily on the human expert help at the pruning and interactive stage of the algorithm thus creating additional subjective bias. But our goal was to develop the user assistant interactive system rather than the autonomous discovery system. However, we are planning to experiment with algorithms incorporating the background knowledge in more general ways (see e.g (13)).

The physics is simple and intrinsically logical, it is the natural room for the symbolic learning algorithms.

REFERENCES

1. Michalski, R. S., *Discovering Classification Rules Using Variable-Valued Logic System CL1*. Proceedings of the 3rd International Conference on Artificial Intelligence, IJCAI, 1973, pp.162-172
2. Kubat, M, Bratko, I, Michalski R, *A Review of Machine Learning Methods.* in Machine Learning and Data Mining: Methods and Applications, Edited by R.S.Michalski, I.Bratko and M.Kubat, John Willey & Sons Ltd, 1996
3. Quinlan, J. R., *C4.5: Programs for Machine Learning.* Morgan Kaufman, San Mateo, CA, 1993
4. Michalski, R. S., *On the quasi-minimal solution of the covering problem.* In Proceedings of the 5th International Symposium on Information Processing (FCIP-69), Vol.A3, pp. 125-128, 1969.
5. Cohen, W. W., *Efficient pruning methods for separate-and-conquer rule learning systems.* In Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 988-994, 1993
6. *ATLAS Detector and Physics Performance Technical Design Report.* CERN-LHCC-99-15, ATLAS TDR 15, 1999
7. Sjöstrand, T. http://www.thep.lu.se/ torbjorn/Pythia.html
8. Abdullin, S., Khanov, A., Stepanov, N. *CMSJET* CMS TN/94-180, (last update 1999)
9. *CMSIM. CMS Simulation and Reconstruction Package.* http://cmsdoc.cern.ch/cmsim/cmsim.html
10. Caner, A. et al. *Track and Vertex Finding Performance with the CMS Inner Tracker.* Nucl. Inst. Meth. A435, pp. 118-143, 1999
11. Stepanov, N., Khanov, A. *Vertex finding with deformable templates at LHC.* Nucl. Inst. Meth. A389, pp. 177-182, 1997
12. *The Tracker Project Technical Design Report.* CERN-LHCC-98-6, CMS TDR 5, 1998
13. Clark, P. and Matwin, S. *Learning Domain Theories using Abstract Background Knowledge.* Tech.Report TR-92-95, Dept CS, Univ. Ottawa, Canada, 1992