Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree

2013 JINST 8 P02013

(http://iopscience.iop.org/1748-0221/8/02/P02013)

PUBLISHED BY IOP PUBLISHING FOR SISSA MEDIALAB

RECEIVED: *November 21, 2012*
REVISED: *December 18, 2012*
ACCEPTED: *January 22, 2013*
PUBLISHED: *February 20, 2013*

2013 JINST 8 P02013

Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree

V.V. Gligorov[a] **and M. Williams**[b,1,2]

[a] *Organisation Européenne pour la Recherche Nucléaire (CERN), Geneva, Switzerland*
[b] *Imperial College London, London SW7 2AZ, U.K.*

E-mail: mwill@mit.edu

ABSTRACT: High-level triggering is a vital component of many modern particle physics experiments. This paper describes a modification to the standard boosted decision tree (BDT) classifier, the so-called *bonsai* BDT, that has the following important properties: it is more efficient than traditional cut-based approaches; it is robust against detector instabilities, and it is very fast. Thus, it is fit-for-purpose for the online running conditions faced by any large-scale data acquisition system.

KEYWORDS: Trigger concepts and systems (hardware and software); Trigger algorithms; Online farms and online filtering; Performance of High Energy Physics Detectors

ARXIV EPRINT: 1210.6861

[1] Corresponding author.
[2] Current address: Massachusetts Institute of Technology, Cambridge, MA, U.S.A.

doi:10.1088/1748-0221/8/02/P02013

Contents

2013 JINST 8 P02013

1 Introduction

The increase in data-production rates in particle physics experiments has led to an increase in the importance of high level trigger (HLT) algorithms performing partial or even full event reconstruction. This increase in data production has been caused by the high collision rates and/or luminosities provided by modern accelerators and the large event sizes provided by modern detectors. The resources required to store and process these large volumes of data do not exist; thus, trigger systems are employed to reduce the data to manageable levels *online*, i.e., during the running of the experiment. This reduction is achieved by preferentially selecting certain events thought to be of particular interest for further analysis, which we shall refer to as "signal" from here on.

First level (referred to as L0 here) triggers, i.e., triggers implemented purely as part of a detector's hardware, have been used since the earliest days of particle physics experiments. In their simplest form, as used in early bubble chambers, L0 triggers make their decision based on the accelerator clock and do not discriminate between different event types. In higher data rate environments they also use information from the detector itself in order to preferentially select events containing signal. An early example of this is the Cronin and Fitch CP-violation experiment which triggered on the coincidence of signals in the scintillator and Cerenkov subdetectors [1]. In even higher data rate environments such as the LHC, L0 triggers are used to rapidly reduce the event rate to the point at which the full detector can be read out. For reasons of speed L0 triggers will typically reconstruct only low-level quantities, for example calorimeter energy deposits or tracks in low-occupancy detectors such as muon chambers. Furthermore, L0 triggers often split the detector into a few regions, each of which is processed independently by a separate L0 processor; this saves time in sending information from one end of the detector to the other, but means that trigger objects cannot straddle these regions. While these simplifications make L0 triggers fast, they also mean

that L0 triggers are only efficient if (a) the signal has a very clear discriminating trait from the background and (b) that trait is accessible by reading out a small number of electronic signals from the detector (without any significant algorithmic processing). Unfortunately, conditions (a) and (b) are often times not satisfied, and in such situations HLT algorithms must be used.

The distinction between L0 and HLT algorithms is that the former makes decisions based on event information from a single subdetector or simple coincidences between subdetectors, while the latter performs an online event reconstruction and makes a decision based on the full event information. Due to timing constraints, the online reconstruction often differs from the offline one, e.g., using a simplified detector geometry or a coarser granularity. An early example of an HLT based trigger was that of the ACCMOR collaboration [2] which used Cerenkov information to separate kaons from pions and momentum information to select events containing ϕ mesons based on the invariant mass of the K^+K^- pairs. This trigger is also notable as it was implemented using commercially-available network-linked PCs.

To-date, HLT algorithms have been cut-based and designed to accommodate the inevitable detector instabilities that occur during online running. Examples of such instabilities include[1] for example: data taking following a major intervention to a piece of the detector hardware and prior to realignment, deployment of incorrect alignment or detector geometry databases, failure of a subdetector region or an entire subdetector, unexpected change in detector occupancy due to unforeseen machine conditions. In all of these cases it is vital that the trigger system is able to keep taking data with minimally altered signal efficiency and background rejection. In this context, cut-based means that while the HLT has access to the full detector information, correlations between the different discriminating variables are largely ignored and an event is selected based on a series of individual criteria combined through a chain of logical ANDs. So the HLT might look for a signal with high transverse momentum AND high displacement from a primary interaction AND high invariant mass. There are, however, experiments currently running, and many more planned for the near future, where cut-based HLT algorithms simply are not powerful enough to discriminate between signal and background. Multivariate classifiers, e.g., neural networks and boosted decision trees (BDTs), are known to be more powerful than cut-based selections. There are, however, some very important concerns regarding using a BDT in an HLT algorithm (these are discussed in detail in section 2).

In this paper we introduce a modification to the standard BDT algorithm that, by construction, addresses these concerns making this algorithm fit-for-purpose for use in a HLT. This HLT algorithm was designed for the LHCb experiment at CERN, whose trigger performance is described in detail in [3]. In 2011 the LHC proton-proton collision rate, at the input to the LHCb L0 trigger, was more than 10 MHz. The LHCb detector could be fully read out at 1 MHz, which was the input rate to the LHCb HLT, which then had around 30 ms in which to make a decision whether to keep or reject an event. With event read-out sizes of $O(100\,\mathrm{kB})$, the LHCb detector was exposed to $O(0.1\,\mathrm{TB})$ of data per second after the L0 trigger, which meant that the data had to be reduced online before being put into permanent storage or used for analysis. None of the signals of interest could be separated from the massive LHC backgrounds using only information available at L0, and the vast majority could not be separated from the backgrounds using single-track algorithms or even by cut-based multi-track ones. Thus, a multivariate classifier had to be developed and deployed.

[1] Every single example given actually occurred during datataking by the LHCb detector in the period 2010–2012.

2013 JINST 8 P02013

The BDT HLT algorithm presented in this paper has been running since the start of LHCb data taking in 2011. Its performance has been excellent. We will refer to the LHCb HLT algorithm as an example; however, the algorithm itself is not LHCb specific. This paper has been written to illustrate how this algorithm works so that it may be used by other experiments. Section 2 describes the basic idea behind the algorithm. Section 3 provides a toy-model example to illustrate how to use the algorithm and also benchmarks its performance against a cut-based approach. In section 4 we discuss the BBDT algorithm performance in the LHCb context, before summarizing in section 5.

2 The bonsai algorithm

Decision trees (DTs) are multivariate classifiers that are built by looping over the variates and repeatedly performing one-dimensional splits of the data. The criteria that determines where to split the data involves maximizing a chosen figure of merit (FOM). For example, a commonly used FOM in particle physics is the so-called *signal significance* $S/\sqrt{S+B}$, where S and B are the number of signal and background events, respectively.

To limit the effects of *over-training* (fine tuning of the DT structure that often occurs during training due to the finite size of the training data sample), DTs are often *boosted*, a technique first used in particle physics by the MiniBoone collaboration for particle identification [4]. One example of a boosting algorithm is *bagging* [5]. Bagging involves making a large number of bootstrap copy training samples by sampling with replacement from the original. One then trains an independent BDT on each bootstrap copy sample; the BDT response is then the fraction of these DTs in which an event is in a signal *leaf* (as opposed to a background one). This procedure greatly enhances the power of the DT. We note here that any FOM and any type of boosting can be used in conjunction with the algorithm described below; these choices were provided to help illustrate how BDT algorithms work. For a good review of BDTs, see ref. [6].

Multivariate classifiers work by defining n-dimensional regions of the multivariate space as signal or *keep regions* (i.e., regions to be selected) by learning from the training data samples provided to them. There are three major concerns which need to be addressed prior to using such a classifier in an HLT algorithm:

- If the keep regions are small relative to the resolution or stability of the detector, the signal could oscillate in and out of the keep regions. This would result in, at best, a less efficient trigger and, at worst, a trigger whose efficiency is very difficult to understand.
- In many cases the signal samples by necessity must come from simulations because the signals have, in fact, not yet been observed in data. In other cases the trigger is meant to be *inclusive*, i.e., the trigger is meant to select classes of signal types rather than one specific signal channel. In both cases, the signal PDFs might not be completely accurate or even available during the training process.
- Any HLT algorithm must run in the online environment; thus, it must be extremely fast.

The simplest way to address these concerns is to discretize all of the variables used in the BDT. This limits where the splits of the data can be made and, in effect, permits the analyst to control and shape its growth; thus, we are calling this a bonsai BDT (BBDT).

2013 JINST 8 P02013

This technique works by enforcing that the smallest keep interval that can be created when training the BBDT is

$$\Delta x_{\text{min}} > \delta_x \; \forall \; x \text{ on all leaves}, \tag{2.1}$$

where,

$$\delta_x = \text{MIN}\{|x_i - x_j| : x_i, x_j \in x_{\text{discrete}}\}. \tag{2.2}$$

The constraints that govern the choice of $\{x_{\text{discrete}}\}$ and ensure that the concerns of using a BDT in an HLT algorithm are addressed are as follows:

- δ_x should be greater than the resolution on x in the detector and should be large with respect to the expected online variations in x.

- The discretization should reflect what properties of the training signal PDFs that are to be incorporated in the BBDT. For example, if the BBDT is meant to be inclusive, then the discretization should help ensure that the BBDT learns the common traits shared by all signals of a given type rather than learning a large sum of traits specific to the signal samples used in the training.

- Discretization means that the data can be thought of as being binned, even though many of the possible bins may not form leaves in the BBDT; thus, there are a finite number, $n^{\text{max}}_{\text{keep}}$, of possible keep regions that can be defined. If the $n^{\text{max}}_{\text{keep}}$ BBDT response values can be stored in memory,[2] then the extremely large number of if/else statements that make up a BDT can be converted into a one-dimensional array of response values. One-dimensional array look-up speeds are extremely fast; they take, in essence, zero time.[3]

Thus, by construction the BBDT is fit for purpose for use in an HLT algorithm and addresses all of the concerns listed above.

3 Toy model example

Having described the method, we will now demonstrate its superiority over a cut-based selection algorithm using a simplified situation in which only two discriminants are available: the transverse momentum[4] (p_T) of the candidates, and their displacement from the primary interaction. Although simplified, the signals and backgrounds used reflect, as much as possible, the actual situation faced by the LHCb experiment in triggering on generic decays of B mesons.

[2] If there is not enough memory available to store all of the response values, there are a number of simple alternatives that can be used. For example, if the cut value is known then just the list of indices for keep regions could be stored.

[3] Obtaining the information required to make the decision takes time; however, since any decision-making algorithm (cuts, BDT, neural network, etc.) will use it, this time is not considered here. The BBDT response is a 1-D array look-up, and its time of execution is too small to be measured using the profiling tools available at LHCb which are sensitive to times of about 1% of the total HLT execution time.

[4] Transverse here means perpendicular to the beam axis.

2013 JINST 8 P02013

3.1 Trigger discriminating variables

The simulated trigger models a situation in which a proton-proton collision has been reconstructed in the tracking system of a detector. Only charged tracks are considered to form part of the event. The p_T of each track is assumed to be reconstructed with no uncertainty.[5] The perpendicular distance of closest approach to the primary interaction for each track, henceforth impact parameter (IP), is calculated from the event geometry. The IP of each track is measured with a precision that varies with the track p_T; larger p_T tracks are more precisely reconstructed. The resolution is implemented as a Gaussian smearing of the actual track IP.

The simulated trigger makes its decisions based on forming a three track vertex. All possible combinations of tracks are assumed to be available to the trigger, and at least one combination must satisfy the trigger algorithm in order for the event to pass. The information available to the trigger is the sums of the p_T and IP of the tracks making up the vertex.

3.2 Toy model data: signal

The signal events are B meson decays. Two different decays are modelled: a decay into four charged tracks (two kaons and two pions), and a decay into five charged tracks (two kaons and three pions). The momentum of the B mesons is sampled from PYTHIA [7], while their lifetime is taken to be 1.5 ps. The B mesons are decayed using the `TGenPhaseSpace` class of the ROOT [8] software framework and assumed to contain no intermediate resonances.

Two different signals are generated to test the inclusive nature of the algorithm; the algorithm will be trained on the four-body decay and tested on both the four- and five-body decays. In order to make the five-body signal a tougher challenge, its lowest transverse momentum track is discarded prior to forming the three-track vertex candidates for the trigger to consider. This reflects the reality that low transverse momentum tracks suffer greater multiple scattering and are harder to reconstruct, especially in the time-limited online environment. The result is that some candidates that could have otherwise fired the trigger, e.g., because of the exceptionally large IP of this low momentum track, are removed from consideration. The signal distributions are shown in figure 1. The five-body signal candidate vertices generally have smaller p_T and IP than the four-body ones.

3.3 Toy model data: backgrounds

Three types of backgrounds are simulated: pure combinatorial background, consisting of three random tracks originating directly from the primary interaction and, therefore, having a true IP of zero; so-called *ghost* background consisting of two tracks from the primary interaction plus one fake track or *ghost*; and prompt charm backgrounds, consisting of two tracks from the decay of a D meson combined with a fake track. These backgrounds reflect the reality of the LHC environment in which one is fighting against multiple fake signatures each of which has its own distribution in the discriminating variables.

The inclusion of fake tracks is important because any tracking algorithm produces some proportion of ghosts which are empirically observed to have an almost random measured IP. The measured IP of ghost tracks is modelled here with a very long-lived[6] exponential. These ghosts

[5] LHC tracking systems universally achieve sub percent precision for this quantity at the HLT reconstruction stage.

[6] The half-life is set to 250 mm to be compared to the distributions in figure 1.

2013 JINST 8 P02013

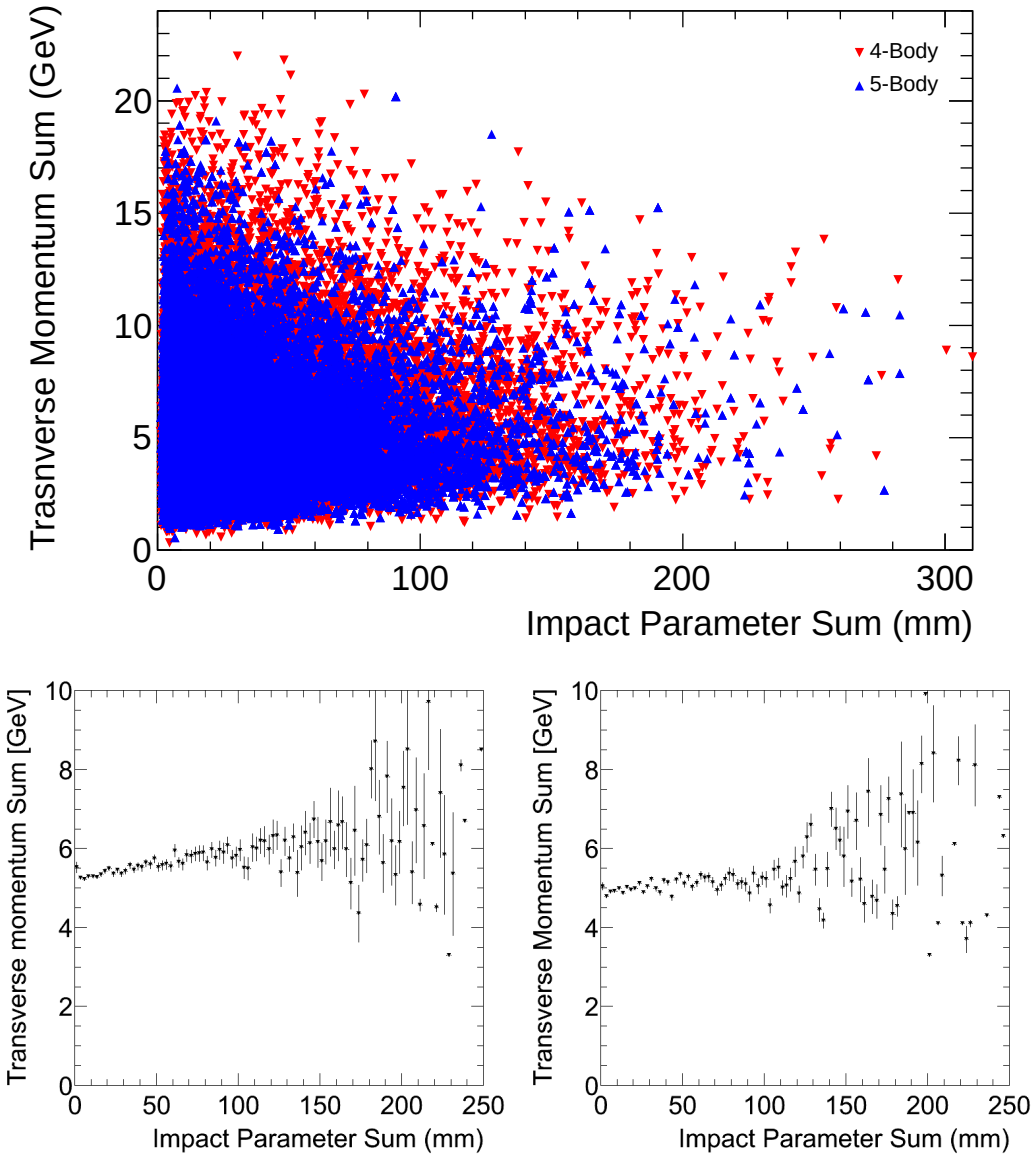

Figure 1. Top: the discriminating variable distributions for the signals. Bottom: the average transverse momentum sum as a function of the IP sum for the four- and five-body signals (left and right respectively).

have a similar p_T distribution to genuine tracks, and present a very challenging background, especially when combined with the decay products of genuinely long-lived D mesons. The latter are modeled in an analogous way to B mesons; the D^+ is chosen as it is the most challenging to reject, with a lifetime of ~ 1 ps.

Every background event is simulated with 30 tracks originating from the primary interaction, of which 10% are ghosts, as well as a single $D^+ \to K^-\pi^+\pi^+$ decay. The background distributions are shown in figure 2. Note that there is an enormous difference in the relative abundances of these three background types, with the pure combinatorics being by far the most abundant and the ghost plus charm background the least abundant (but most dangerous).

2013 JINST 8 P02013

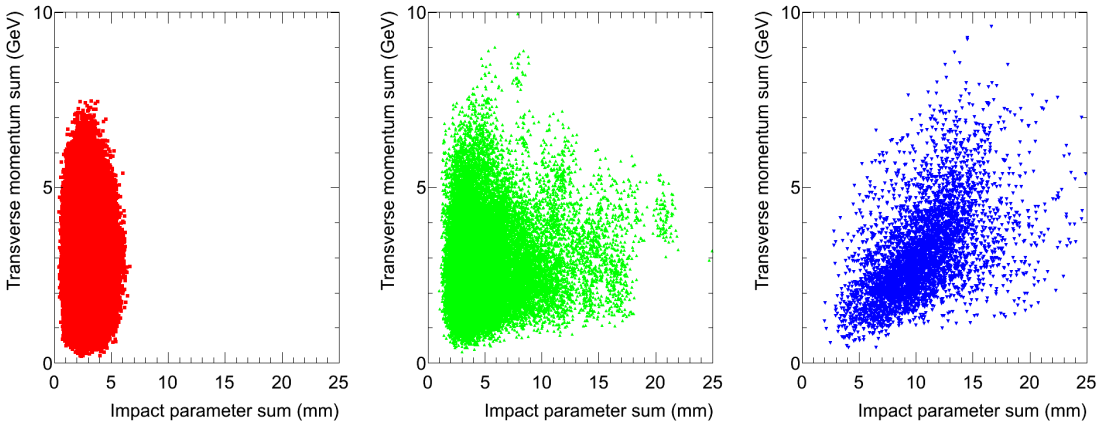

Figure 2. Discriminating variable distributions for the backgrounds. From left to right: pure combinatorial, ghost background, prompt charm.

Table 1. Performance on the toy model data of a cut-based, BDT-based and BBDT-based HLT algorithm. $\varepsilon_{n-\text{body}}$ are the efficiencies on the n-body signals. The instability is the increase in the rate under the imperfect online conditions (see text for details).

type	$\varepsilon_{4-\text{body}}$ (%)	$\varepsilon_{5-\text{body}}$ (%)	instability (%)
cuts	63.2 ± 0.5	55.3 ± 0.5	9 ± 3
BDT	76.9 ± 0.4	68.1 ± 0.5	55 ± 4
BBDT	73.8 ± 0.4	68.9 ± 0.5	10 ± 3

3.4 Performance

For each of the HLT algorithms defined below, the optimal selections are determined using *training* samples of the 4-body signal and background. Their performances are then evaluated using *validation* samples of the 4-body signal, 5-body signal and background. The optimal selections are simply those that maximize the efficiency on the 4-body signal while achieving a factor of 100 reduction in the background. The stability of each HLT algorithm is tested using validation samples that include an additional 1σ smearing of the IP.

Three HLT algorithms are studied: cut-based, BDT-based and BBDT-based. The performance of each is given in table 1. As expected, the BDT and BBDT are much more efficient than the cuts. This simple example only has two variables, but the relative efficiency of the (B)BDT-based HLT is (19%)22% higher than the cut-based one. The BBDT is 3% less efficient than the BDT on the 4-body signal; however, it is 1% more efficient on the 5-body signal. For an inclusive HLT, the efficiency on signals not included in the training is just as important than on those in the training. The BBDT in this example has a very slight edge here over the BDT. In a real-world example, the more variables that are included in the BDT the better the performance of the BBDT (compared to the BDT) should be on signals not used in the training.

The most important column in table 1 is the one showing the instability of the rate under

2013 JINST 8 P02013

imperfect online running conditions. The BDT rate increases 6 times more than the cut-based rate does; however, the BBDT rate is very close to the cut-based one. To reiterate the points made above, this happens because the BDT is able to define leaves (regions of the multivariate space) that are to be kept that are small with respect to the variations found in the variables during online running. This means that a simple 1σ decrease in resolution is able to move many background events into a leaf that is to be kept resulting in a much higher rate. The BBDT avoids this by discretizing the variables in a way that is *safe* online; e.g., the smallest allowed leaf is 5σ wide in IP which limits how much a 1σ decrease in resolution can affect the rate.

For an experiment with this level of possible instability online, the BDT performance is unacceptable. If one knew how much instability to expect, the training data could just be smeared to account for it. Unfortunately, online instabilities are expected to occur but where and by how much is unknown; thus, it is very difficult to produce training samples that guard against all possible online instabilities. It is also very time consuming to attempt to do so and difficult to validate.

This section demonstrates that the BBDT is almost as stable as cuts while being almost as efficient as a BDT. This makes it ideal for an HLT algorithm. This BBDT had 10 allowed split points per variable. The smallest allowed keep region that could be defined was 5σ wide (in the lowest region of IP). In general, more allowed split points gives higher efficiency but also introduces more sensitivity to online instabilities. When defining the allowed split points, the analyst has total control over the size of the smallest possible keep regions that can be defined. Typically, it is possible to define the split points such that online instabilities will not be an issue without the need for complicated simulation studies.

4 Performance at LHCb

The performance of the BBDT algorithm as actually deployed in the LHCb experiment is described in detail in [3]. Without duplicating any results, we would emphasize the following points. The BBDT is implemented for the so-called *topological* trigger that aims to select any B meson decay which produces charged tracks (kaons, pions, protons, muons or electrons). It does so by reconstructing two-, three-, and four-track vertices displaced from the primary interaction and then calculating the BBDT response based on the properties of this vertex and the properties of the tracks from which the vertex has been reconstructed.

Since its deployment, comprising about 98% of the luminosity collected to-date, the BBDT topological trigger has served as the main trigger for B physics at LHCb. During this time it has undergone no fundamental changes or retuning. In particular, it has proven robust against the misalignments present at the start of each calendar year (introduced following hardware interventions during winter shutdowns of the accelerator), and against the change from 7 to 8 TeV centre of mass energy of the collisions.

The BBDT topological trigger has been measured in a data-driven manner to be highly efficient for a variety of B decay topologies. Its efficiency has been measured as a function of various B meson properties and has displayed no pathological structures indicative of overtraining or resolution effects. The BBDT algorithm is on average about 20% more efficient than its cut-based predecessor; this value is close to 50% for many important decay modes at LHCb. Since obtaining the response only involves a 1D array lookup, it is not possible to observe any timing differences

2013 JINST 8 P02013

between the BBDT and cut-based algorithms. Furthermore, as described in the LHCb upgrade letter of intent [9], the BBDT topological trigger has been verified in simulation to maintain its performance at 14 TeV centre of mass energy and a 25% greater per-bunch luminosity. We conclude that the BBDT achieves in practice its stated aims of being efficient on signal, powerful against background, robust against mis-reconstruction, and understandable in its behavior.

5 Summary

The BBDT algorithm presented in this paper has the following important properties: it is more efficient than traditional cut-based approaches; it is robust against detector instabilities, and it is very fast. Thus, it is fit-for-purpose for the online running conditions faced by any large-scale data acquisition system. The BBDT has been proven to work at the LHCb experiment at CERN. It should be considered for use in any current or future HLT algorithm.

Acknowledgments

We would like to thank our colleagues at the LHC*b* experiment, especially Roel Aaij, Johannes Albrecht, Vanya Belyaev, Hans Dijkstra, Ulrik Egede, Richard Jacobsson and Gerhard Raven. MW was supported in this work by STFC grant number ST/H000992/1. VVG was supported in this work by a Marie Curie Action: "Cofunding of the CERN Fellowship Programme (COFUND-CERN)" of the European Community's Seventh Framework Programme under contract number (PCOFUND-GA-2008-229600).

References

[1] J. Christenson, J. Cronin, V. Fitch and R. Turlay, *Evidence for the 2π decay of the K_2^0 meson*, *Phys. Rev. Lett.* **13** (1964) 138 [INSPIRE].

[2] ACCMOR collaboration, C. Daum et al., *Inclusive ϕ meson production in 93 GeV and 63 GeV hadron interaction*, *Nucl. Phys.* **B 186** (1981) 205 [INSPIRE].

[3] LHCB TRIGGER GROUP collaboration, R. Aaij et al., *The LHCb Trigger and its Performance*, submitted to *JINST* (2012), arXiv:1211.3055 [INSPIRE].

[4] B.P. Roe et al., *Boosted decision trees, an alternative to artificial neural networks*, *Nucl. Instrum. Meth.* **A 543** (2005) 577 [physics/0408124] [INSPIRE].

[5] L. Breiman, *Bagging Predictors*, *Machine Learning* **24** (1996) 123.

[6] L. Brieman et al., *Classification and regression trees*, Wadsworth International Group, Belmont U.S.A. (1984).

[7] T. Sjöstrand, S. Mrenna and P.Z. Skands, *PYTHIA 6.4 Physics and Manual*, *JHEP* **05** (2006) 026 [hep-ph/0603175] [INSPIRE].

[8] R. Brun and F. Rademakers, *ROOT: An object oriented data analysis framework*, *Nucl. Instrum. Meth.* **A 389** (1997) 81 [INSPIRE]. See also http://root.cern.ch/.

[9] LHCB collaboration, *Letter of Intent for the LHCb Upgrade*, CERN-LHCC-2011-001 (2011).

2013 JINST 8 P02013