Studying a definition for a boosted W/Z/H jet tagger at the FCChh, employing modern Machine Learning algorithms and customised features (beyond the usual substructure variables)

Danyyl Brzhechko

1 Introduction

A jet is a spray of particles, usually produced by the hadronization of a quark or gluon in a particle physics or heavy ion experiment. Reconstructed particles are clustered into jets using one of the available jet clustering algorithms ($k_T$, anti-$k_T$ etc.), which adopt different metrics to decide if two given particles belong to the same jet or not. Jets can also originate from the decay of high-momenta heavy particles, such as boosted vector boson. When these particles decay to quarks, the overlap of the hadronization products of each quark result into a single massive jet, different than the ordinary jets from quarks and gluons.

These special jets can be identified using substructure algorithms. In this study, we consider the performances of a commonly used substructure variable, N-subjettiness, with two variants of an alternative approach, based on the momentum flow around the jet axis. I focused on high-energy collision in a hypothetical future circular collider (FCC) colliding protons at a center-of-mass energy 100 TeV.

2 Setup

The PYTHIA8 event generator is used to create simulated samples of signal and background events. The Delphes package is used to simulate the detector reconstruction, taking as a reference the CMS Phase II detector. Jets are reconstructed in a custom C++ package, based on ROOT. This code is interfaced to FASTJET to cluster jets and compute substructure variables (allow you to use different filters).

2.1 Signal samples used

The production of heavy gravitons decaying to pairs of W bosons, $G^* \rightarrow WW$, is taken as a benchmark signal source of high-momentum Ws. Each W is forced to decay to light quark pairs (u, d, c, s, b). I generated 100000 events with 50 and 100 TeV energy of collisions with six different resonance mass values: 5, 10, 15, 20, 25, 30 TeV.

2.2 Background samples used

A sample of high-momentum ordinary jets is generated through the processes: $pp \rightarrow G^* \rightarrow qq, pp \rightarrow G^* \rightarrow gg$. For each we have 50000 events and the same energy and masses as for signal.

In addition, the performances of the tagging algorithm are then tested in a realistic physics search. In this case, a dijet background from multijet production at 100 TeV collisions is considered.

2.3 Processing chain

2.3.1 First step - generate HepMC files

Signal and background samples were generated through Pythia saving the output sample in the hepmc format.

2.3.2 Second step - generate ROOT files from HepMC

All HepMC files are given as input to Delphes, which is configured to run its particle-flow reconstruction and return a list of reconstructed particles in a ROOT TTree.
2.3.3 Third step - reconstruction with FASTJET
A customized C++ code, interfaced to FASTJET, is used to reconstruct jets and compute substructure variables. The following quantities are considered:

- mass of the jet (non-filtered and filtered)
- transverse momentum (for non-filtered and filtered jet)
- transverse momentum flow with respect to the beam and jet axes (for non-filtered and filtered jet)
- subjettiness.

Figure 1 shows a screenshot of a ROOT TBrowser, in which the leafs of a Tree containing the substructure information for th highest-pT jets in the generated events are shown.

Fig. 1: Leafs of a Tree containing the substructure information for th highest-pT jets in the generated events

3 Jet reconstruction

3.1 Jet clustering
The anti-k_T algorithm clusters jets according to the following algorithm

\[ d_{ij} = \min(p_{ti}^2, p_{tj}^2) \frac{\Delta R_{ij}^2}{R^2}, \]  

(1)

where \( p_{ti} \) is the transverse momentum of the particle \( i \) with respect to the beam direction, \( \Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2 \) with \( y_i = \frac{1}{2} \ln \frac{E_i + p_{zi}}{E_i - p_{zi}} \) and \( \phi_i \) respectively \( i \)'s rapidity and azimuth. The anti-k_T algorithm also involves a distance measure between every particle \( i \) and the beam \( d_{iB} = p_{ti}^{-2} \). \( R \) usually called the jet radius, is a parameter of the algorithm that determines its angular reach.
In the anti-kT algorithm, the \( d_{ij} \) and \( d_{iB} \) distances are the same as above. When a \( d_{iB} \) is smallest, then \( i \) is removed from the list of particles/pseudojets and added to the list of final inclusive jets (this is instead of being incorporated into a beam jet). There is no \( d_{cut} \) threshold and the clustering continues until no particles/pseudojets remain. Of the final jets, generally only those above some transverse momentum are actually used. Because the distance measures are the same in the inclusive and exclusive algorithms, the clustering sequence is common to both formulations (at least up to \( d_{cut} \)), a property that will be reflected in FASTJETs common interface to both formulations.

Sequential recombination jet algorithms are the main kind of algorithm in use at CERN Large Hadron Collider (LHC)[1].

3.2 Pruning

Pruning consists in reclustering a jets constituents with some given sequential recombination algorithm, but vetoing soft and large-angle recombinations between pseudojets \( i \) and \( j \), specifically when the two following conditions are met:

- the geometric distance between \( i \) and \( j \) is larger than a parameter \( R_{cut} \), with \( R_{cut} = R_{cut\_factor} \times 2m/p_t \), where \( m \) and \( p_t \) are the mass and transverse momentum of the original jet being pruned, \( R_{cut\_factor} \) is the external parameter of the pruner method;
- one of \( p_{ti}^2, p_{tj}^2 \) is \(< z_{cut} \times p_{ti} + j \), where \( z_{cut} \) is external parameter of pruner method.

When the veto condition occurs, the softer constituents are discarded, while the harder one continues to participate in the clustering.

Pruning bears similarity to filtering (see next section) in that it reduces the contamination of soft noise in a jet while aiming to retain hard perturbative radiation within the jet. However, because by default the parameters for the noise removal depend on the original mass of the jet, the type of radiation that is discarded significantly on the initial jet structure. As a result pruning, in its default form, is better thought of as a noise-removing boosted-object tagger (to be used in conjunction with a pruned-jet mass cut) rather than a generic noise-removal procedure.

The `fastjet::Pruner` class, derived from `Transformer`, can be used as follows, using a `JetAlgorithm` and two double parameters (`zcut` and `Rcut_factor`):

```cpp
#include "fastjet/tools/Pruner.hh"
// ...
Pruner pruner(jet_algorithm, zcut, Rcut_factor);
// ...
PseudoJet pruned_jet = pruner(jet);
```

The pruned jet will have a valid associated cluster sequence, so that one can, for instance, ask for its constituents with pruned jet.cluster() [1].

3.3 Filtering

Generally speaking, filtering clusters a jets constituents with a smaller-than original jet radius \( R_{filt} \). It then keeps just the \( n_{filt} \) hardest of the resulting subjets, rejecting the others. Trimming is similar, but selects the subjets to be kept based on a \( p_t \) cut. The use of filtering and trimming has been advocated in number of contexts, beyond just the realm of boosted object reconstruction[1].

3.4 Jet mass

Given a jet algorithm (with or without applying filtering/pruning), the jet mass is computed summing the four momenta of the jet constituents and taking the invariant mass of that. The jet mass is one of the most discriminating variables between signal and background.
4 Substructure variables

4.1 Subjettiness

The so-called N-subjettiness for a jet can be computed as

\[ \tau_N = \frac{1}{d_0} \sum_k p_{T,k} \min(\Delta R_{1,k}, \Delta R_{2,k}, ..., \Delta R_{N,k}). \] (2)

Here, \( k \) runs over the constituent particles in a given jet, \( p_{T,k} \) are their transverse momenta, and \( \Delta R_{J,k} = \sqrt{(\Delta \eta)^2 + (\Delta \phi)^2} \) is the distance in the rapidity-azimuth plane between a candidate subjet \( J \) and a constituent particle \( k \). The normalization factor \( d_0 \) is taken as

\[ d_0 = \sum_k p_{T,k} R_0, \] (3)

where \( R_0 \) is the characteristic jet radius used in the original jet clustering algorithm. On the Fig. 2 one can see an example of the distribution of \( \tau_2 \) for \( W \) jet originating from \( G^* \rightarrow WW \) decays, with the \( G^* \) mass set to 15 TeV.

Fig. 2: Distribution of 2-subjettiness for non-filtered data for \( W \) jet with the 15 TeV mass of the resonant

<table>
<thead>
<tr>
<th>htemp</th>
<th>Entries</th>
<th>Mean</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>tau2</td>
<td>37257</td>
<td>0.01092</td>
<td>0.006197</td>
</tr>
</tbody>
</table>

4.2 \( p_t \) flow with respect to the jet axis

We defined \( p_{t,flow} \) variables as

\[ p_{t,flow} = \frac{\sum_k \sqrt{p_{T,k,J}^2 - (\vec{p}_{k,J} \cdot \vec{n}_J)^2}}{p_{T,J}}, \] (4)

where \( \vec{p}_{k,J} \) is momentum of \( k \)'s particle from jet \( J \), \( \vec{n}_J = \frac{\vec{p}_J}{p_{T,J}} \) is a unit vector that points towards jet axis, \( p_{T,J} \) is transverse momentum of the jet.
4.3 $\mathbf{p}_T$ flow with respect to the beam

The $p_T$ flow variables can be computed also using the $p_T$ of the particles in the reference system of the detector, rather than with respect to the jet axis. In this case

$$p_{T,\text{flowbeam}} = \frac{\sum_k p_{T,k,J}}{p_{T,J}},$$

(5)
where $p_{T,k,J}$ is transverse momentum of the $k$'s particle in $J$'s jet.

Fig. 5: Schematic view of a decision tree. Starting from the root node, a sequence of binary splits using the discriminating variables $x_i$ is applied to the data. Each split uses the variable that at this node gives the best separation between signal and background when being cut on. The same variable may thus be used at several nodes, while others might not be used at all. The leaf nodes at the bottom end of the tree are labeled S for signal and B for background depending on the majority of events that end up in the respective nodes. For regression trees, the node splitting is performed on the variable that gives the maximum decrease in the average squared error when attributing a constant value of the target variable as output of the node, given by the average of the training events in the corresponding (leaf) node.

5 Tagging algorithm

5.1 BDT training

A decision tree is a binary tree structured classifier similar to the one sketched in Fig. 5. Repeated left/right (yes/no) decisions are taken on one single variable at a time until a stop criterion is fulfilled. The phase space is split this way into many regions that are eventually classified as signal or background, depending on the majority of training events that end up in the final leaf node. In case of regression trees, each output node represents a specific value of the target variable. The boosting of a decision tree extends this concept from one tree to several trees which form a forest. The trees are derived from the same training ensemble by reweighting events, and are finally combined into a single classifier which is given by a (weighted) average of the individual decision trees. Boosting stabilizes the response of the decision trees with respect to fluctuations in the training sample and is able to considerably enhance the performance w.r.t. a single tree[3].

Several BDTs were trained on different sets of variables:

1. Subjetinesses (without filters, with pruner or with trimmer):
5 Tagging algorithm

- Mass of the jet;
- 1-subjettiness: $\tau_1$;
- 2-subjettiness: $\tau_2$;
- 3-subjettiness: $\tau_3$;

2. transverse momentum flow with respect to the jet axis (without filters, with pruner or with trimmer):
- Mass of the jet;
- pt_flow[0];
- pt_flow[1];
- pt_flow[2];
- pt_flow[3];
- pt_flow[4];

![Fig. 6: Rejected background vs. mass of the resonant on 90% of the signal.](image)

3. transverse momentum flow with respect to the beam (without filters, with pruner or with trimmer):
- Mass of the jet;
- pt_flowbeam[0];
- pt_flowbeam[1];
- pt_flowbeam[2];
- pt_flowbeam[3];
- pt_flowbeam[4];

I used six masses of the resonant ($G^*$): 5, 10, 15, 20, 25, 30 TeV. So, there were generated different weight files and we can make some conclusions.
Fig. 7: Rejected background vs. mass of the resonant on 90% of the signal. Different colours relate to different algorithms.

![Background vs. Mass of the Resonant (90% of the signal)]

Fig. 8: Rejected background vs. mass of the resonant on 90% of the signal. Different colours relate to different algorithms.

![Background vs. Mass of the Resonant (90% of the background)]

5.2 Comparison of algorithms with different jets and variables

After training we have to consider which variable is the best to recognize jets from data. Fig. 6 shows rejected background vs. different masses of the resonant on 90% efficiency of the signal. The best choice is transverse
momentum with respect to the beam using trimmer.

In practice, one doesn’t know the mass of the particle one is searching for. It is then needed to define an algorithm that allows to discriminate Ws from plain jets without requiring a given mass (hence a jet $p_T$ distribution).

TMVA generates file with weights for every BDT training. We can use these weight files for different resonance
mass and apply each for different masses. For example, we have trained BDT for some mass value. We can use its weight file to run TMVA for every resonance mass and get BDT response histograms (we have six). After that, we sum these histograms, so we have BDT response that doesn’t depend on mass anymore, except the dependence of the algorithm (weight file). After that we have six summed BDT response histograms for each trained BDT (for 5, 10, 15, 20, 25, 30 TeV). We chose algorithm, that was trained for mass resonance of 20 TeV as the best independent of mass algorithm for mass range from 5 to 30 TeV. Fig. 7 shows one the rate of the rejected background vs. mass of the resonant. Also I have created another plots to choose the best algorithm (weight file). On Fig. 8, 9 and 10 you can see different plots.

### 5.3 Performances of the best algorithm vs mass

So, the best algorithm to choose is 20 TeV algorithm, because it places closest to the transverse momentum with respect to the beam (blue triangles) for higher masses of the resonant (bigger then 10-13 TeV). Typical background rate of 5-15% is obtained for a signal efficiency of 90%. Typical signal rate of 60-95% is obtained for a background efficiency of 90%. This study highlights the potential interest in exploring the jet momentum flow to tag jets from boosted heavy particles, for future high-energy colliders but also at the LHC.

### References