Finding Higgs $\to b\bar{b}$ in Boosted Final States

Terascale Monte Carlo School 2014, DESY Hamburg

2014-03-13

1 Introduction

In this tutorial we will discuss boosted final states, using the example of a Standard Model Higgs boson decaying to two b-jets. You may work in groups, where each member generates predictions from a different event generator. Ideally, you will have a selection of people who already know "their" generator from using it in the first tutorial. For instructions on how to exchange results, please review yesterday's tutorial worksheet.

2 General considerations

In this tutorial we are going to look at a boosted Higgs boson analysis. We will follow the analysis presented in [1], which aims at isolating the Higgs boson signal in the reaction $pp \to VH$, with the Higgs decaying to a pair of b-jets. The fact that the Higgs is boosted is essential in reducing backgrounds. The analyses are adapted from those used in [2].

Both background and signal samples are difficult to generate within the time frame of the tutorial, and they have therefore been prepared for you. Your task now is to isolate the signal from the background, first using cuts, and then some (primitive) boosted techniques. We will use Rivet as analysis tool. Start with defining cuts on the non-merged samples. Use the same cuts on the merged background samples, and then also on the merged signal samples. Then, move to the more sophisticated boosted techniques to isolate the signal.

3 Analysing the event samples

First you need to build the Rivet analysis plugins. Change to the directory `~/tutorials/boost/analysis/` and run

```
make
```

This will build a Rivet plugin library from the source files in this directory. Your task is to fill this skeleton with life by modifying the source code at a later time.

3.1 Running rivet on the event samples

The backgrounds we are going to consider are $pp \to V + \text{jets}$ and $pp \to t\bar{t} + \text{jets}$, where V can stand for either $W^{\pm}$ or Z. The signal sample contains $pp \to b\bar{b}V + \text{jets}$ events. Various types of (background and signal) samples are available:

A Non-merged samples, in which all jet activity is produced through parton showering:
`pp2v_nonmerged.hepmc.bz2`, a sample with $pp \to V + \text{jets}$ events,
`pp2vv_nonmerged.hepmc.bz2`, a sample with $pp \to VV + \text{jets}$ events,
`pp2tt_nonmerged.hepmc.bz2`, a sample with $pp \to t\bar{t} + \text{jets}$ events, with fully leptonic decays
`pp2hv_nonmerged.hepmc.bz2`, a sample with $pp \to HV + \text{jets}$ events (signal sample);

B Matched samples, in which the Born states are described at NLO accuracy:
`pp2v_matched.hepmc.bz2`, a sample with $pp \to V +$ jets events,
`pp2vv_matched.hepmc.bz2`, a sample with $pp \to VV +$ jets events;

C Merged samples, in which multiple jets are described with tree-level accuracy
`pp2tt_merged_central.hepmc.bz2`, a sample with $pp \to t\bar{t}+$jets events, with fully leptonic decays, and $\mu_f = \mu_r = \mu_{hard}$
`pp2tt_merged_high.hepmc.bz2`, a sample with $pp \to t\bar{t} +$ jets events, with fully leptonic decays, and $\mu_f = \mu_r = 2\mu_{hard}$
`pp2tt_merged_low.hepmc.bz2`, a sample with $pp \to t\bar{t}+$jets events, with fully leptonic decays, and $\mu_f = \mu_r = \frac{1}{2}\mu_{hard}$
`pp2v_merged_short.hepmc.bz2` and `pp2v_merged.hepmc.bz2`, samples with $pp \to V +$jets events, with 10K (100K) events,
`pp2vv_merged_short.hepmc.bz2` and `pp2vv_merged.hepmc.bz2`, samples with $pp \to VV +$ jets events, with 10K (100K) events,
`pp2hv_merged_short.hepmc.bz2` and `pp2hv_merged.hepmc.bz2`, samples with $pp \to HV +$ jets events, with 10K (100K) events (signal sample).

All samples have been produced at the hadron level (i.e. including MPI and hadronisation). The samples are available on an external file server. To get access, execute the `~/tutorials/mountSamples.sh` script with

```
sudo ~/tutorials/mountSamples.sh
```

Now the samples are mounted in `~/tutorials/samples/`. Change to this directory. In order not to expand the large file, we use a fifo pipe, which is created as

```
mkfifo events
```

Now you can unzip the event file and, at the same time, run rivet to analyse the events, e.g.

```
bzcat pp2v_nonmerged.hepmc.bz2 > events & \
rivet -a HIGGS_ANALYSIS -H pp2v_nonmerged.aida events
```

You can shorten it by analysing fewer events (consider the help options of Rivet for a way to do that), but your final plots may have insufficient statistics in this case.

3.2 Cut-based analysis

The analysis code (`HIGGS_ANALYSIS.cc`) contains three histograms:

`_h_H_mass` gives the mass of the hardest jet (with jets defined in the anti-k$_\perp$ algorithm with $R = 0.5$, $p_{\perp j,min} = 30$ GeV);

`_h_H_mass_btag` gives the mass of the hardest b-tagged jet (assumming a b-tagging efficiency of 100% and a uniform mistag probability of 2%);

`_h_sigmacut` gives the cut flow, i.e. how the cross section changes upon selection cuts.

Why does it make sense to histogram the mass of the hardest jet? Can you think of other useful histograms?

As already mentioned above, you should start with defining cuts on the non-merged samples. The analysis code already contains one very simple selection:

Selection A: Veto on events with more than two charged leptons.

Think about the structure of the signal final state, and devise new selection criteria. Some possible choices could be

Selection B: An e^+e^- pair or $\mu^+\mu^-$ pair with an invariant mass close to the Z-boson mass, possibly with a high transverse momentum;

Selection C: A large missing transverse momentum;

Selection D: Missing transverse momentum and a lepton with transverse momentum consistent with a W boson of nominal W-boson mass, possibly with $p_T^W > 200$ GeV.

Note that some of these selections will also influence your signal efficiency.

Then, use the same cuts on the merged background samples. What do you see? Finally, use the merged signal samples. How does this influence the efficiency of your cuts? Do not spend too much time on optimising the cuts, and instead move on to the more sophisticated boosted techniques to isolate the signal.

3.3 Trimming, Pruning and Filtering

Jet substructure studies has in recent years developed into a vibrant and productive field. Here, we cannot possibly cover all developments. Instead, we give very basic suggestions, and hope that you'll fill the gaps creatively.

Since a standard Rivet installation contains Fastjet, you can use different powerful jet substructure tools right away. So far your analysis only contained selection cuts. Now try the more sophisticated methods. The natural place where to employ such methods in the `hasCandidateHiggs` function, where the filtered jet (`filteredJets`) is defined.

Let's begin with trimming. Trimming basically tries to remove soft radiation and underlying event contamination by discarding soft particles. Replace the line

```
filteredJet = tagged_jet;
```

with

```
// Trimming.
double r_filter    = 0.3;
fastjet::Selector sel_above_ptfrac   = fastjet::SelectorPtFractionMin(0.05);
fastjet::Filter trimmer(r_filter,sel_above_ptfrac);
filteredJet        = trimmer(tagged_jet);
```

This leads to sub-jets that carry less than 5% of the jet's transverse momentum are removed. After this change, you will have to recompile the analysis, see Sec. 3. Can you find differences between trimmed and non-trimmed results? What process is most affected?

Another common method is pruning. On top of rejecting soft jets, pruning also introduces a "pruning radius", which can lead to merging soft jets with harder jets. To use pruning, remove the trimming lines, and instead add

```
// Pruning.
double zcut        = 0.1;
double rcut_factor = 0.5;
fastjet::Pruner pruner(fastjet::cambridge_algorithm,zcut,rcut_factor);
filteredJet        =  pruner(tagged_jet);
if ( filteredJet == 0 || filteredJet.pieces().size() < 1 ) return false;
```

Then recompile. Given the choice, would you choose pruning or trimming, and why?

Finally, we will look at the mass drop method. This method is, in a fashion, independent of the previous ones, since it can be used in combination with these methods. However, it is often used in conjunction with "filtering". The first step in the mass drop method is to replace the jet tagger

```
const fastjet::PseudoJet& tagged_jet = fastjet::PseudoJet(ptJets[0]);
```

by the more sophisticated

```
// Better tagger: Mass drop
fastjet::MassDropTagger mdtagger(0.67,0.09);
const fastjet::PseudoJet& tagged_jet = mdtagger(ptJets[0]);
// Return if no jet has been tagged.
if ( tagged_jet == 0 ) return false;
```

This checks if there is a significant drop in mass of the jet when combining subjets. If so, a heavy intermediate particle is likely, and thus the jet is tagged. You can apply filtering on top of this improved tagger by removing the pruning (and trimming) lines, and instead adding

```
// Filtering.
const std::vector<fastjet::PseudoJet>& tagged_pieces = tagged_jet.pieces();
double r_filter    = min(0.3,0.5*tagged_pieces[0].delta_R(tagged_pieces[1]));
fastjet::Filter filter(r_filter,fastjet::SelectorNHardest(3));
filteredJet        = filter(tagged_jet);
```

Again, you need to recompile. What differences do you find between the primitive and the mass-drop tagger?

References

[1] J. M. Butterworth, A. R. Davison, M. Rubin and G. P. Salam, Phys. Rev. Lett. **100** (2008) 242001 [arXiv:0802.2470 [hep-ph]].

[2] P. Richardson and D. Winn, Eur. Phys. J. C **72** (2012) 2178 [arXiv:1207.0380 [hep-ph]].