Table of Contents

Jet Rejector Tool **1**
- Introduction 1
- How to run the tool 1
- Variables 2
- Talks 3

Jet Rejector Tool under MVA Framework **4**
- Introduction 4
- How to get the code 4

Jet Rejector Tool

Introduction

The JetRejector Tool produces a Combined Likelihood Ratio value for each jet in order to allow the user to choose the jets with the highest CLR value. This tool produces, also, S/(S+B), fit and correlation of the variables and the Combined Likelihood Ratio value for Signal and Background, purity and efficiency plots. The structure of the tool is modulare and flexible: the user can choose the definition of the correct/unwanted jets in the cfg file and the observables to be used to make the CLR.

How to run the tool

```
scramv1 project CMSSW CMSSW_1_6_0
cd CMSSW_1_6_0/src
cmscvsroot CMSSW
eval `scramv1 runtime -csh`

cvs co -r v00-01 PhysicsTools/JetRejectorTool
cvs co -r TQAF_160_070908 AnalysisDataFormats/TopObjects
cvs co -r TQAF_16X_2007 TopQuarkAnalysis/TopTools
cvs co -r V00-05-00 PhysicsTools/JetMCAlgos
scramv1 b
```

The user can choose the signal definition using the appropriate configuration files which are characterized by the suffix *_genjet.cfg, *_parton.cfg or *_partonjet.cfg according to the "MC truth" definition used.

```
1) go in PhysicsTools/JetRejectorTool/test
    open JetRejObsProducer_*.cfg
    set your input file, your cuts and your input branches
    > cmsRun JetRejObsProducer_*.cfg
```

To produce the fit function: (If you don't want to produce the fit functions you can skip the point 2 and go to the point 3)

```
2) go in PhysicsTools/JetRejectorTool/bin
    > eval `scramv1 runtime -csh`
    > scramv1 b
    > JetRejCombLR_SoverSplusBLoop_*
```

and you can have a look at the observable distributions and the S/(S+B) fit in

```
> cd ../data
```

In JetRejCombLRAllObs.root and JetRejCombLRAllObs_*.ps you can find the plots of all the Observables for the Wanted and Unwanted Jets, the S/(S+B) plots and the fit, the correlations between all the Observables.

```
3) go in PhysicsTools/JetRejectorTool/test
    open JetRejComLR_*.cfg
    choose the observables
    > cmsRun JetRejComLR_*.cfg
```

In JetRejComLR.root is stored the total distribution of the CombinedLikelihoodRatio (doubles_jetcomLR__MYLR), while in

```
    >  cd ../data
```

JetRejLRJetCombSelObsAndPurity.ps and JetRejLRJetCombSelObsAndPurity.root the distributions of the selected observables, the CombLikelihoodRatio for Signal and Background and the Purity vs Efficiency plot are stored.

In your code:

```
  Handle<vector<reco::CaloJet> > jets;
  iEvent.getByLabel ("seljets",jets);

  Handle<vector<double> > LRValue;
  iEvent.getByLabel ("jetcomLR",LRValue);

   for(unsigned int ji = 0; ji <  LRValue->size(); ji++ ){
(*jets)[ji] .......    // use the pre-selected CaloJets in a LRValue loop
 }
```

Variables

Observables in CMSSW_1_6_0:

The following preselection cuts are applied as default:

* Number of Constituents > 1; * Et jet > 2 GeV (to be tuned); * Number of tracks associated to the Jet > 0.

The user can change the first and the second in the

```
 >  PhysicsTools/JetRejector/test/JetRejObsProcuder.cfg

    double jetEnergycut  = 2   // Et jet > 2GeV
    double jetNConstMin = 2   // Number of constituents > 1
```

Jet Variables

```
 obs1 : ETA Jet
 obs2 : (EMCalEnergyFraction - HadCalEnergyFraction) / (EMCalEnergyFraction + HadCalEnergyFractio
 obs3: Pt Jet
 obs4: n90

 Tower Variables
  obs5: MaxEnergyEMTower
  obs6: MaxEnergyHadTower
  obs7: Tower Number
  obs8: Highest Et Tower / Et Jet
  obs9: Sum(E Twr * DeltaR(Twr-Jet)) / Sum(E Twr)
  obs10: Pt Jet / Towers Area
  obs11: Highest Et Tower / Towers Area
```

```
Track Variables
 obs12: Highest TrackPV Pt / Pt Jet
 obs13: Beta^2 = (Sum(Pt TrackPV) / Sum(Pt Track))^2
 obs14: Alpha = Sum(Pt TrackPV) / Pt Jet
 obs15: N Track PV
```

Observables in CMSSW_1_3_1:

```
 obs1: ETA Jet
 obs2: EMCalEnergyFraction/ (EMCalEnergyFraction + HadCalEnergyFraction)
 obs3: Pt Jet (log)
 obs4: n90

Tower Variables
obs5: MaxEnergyEMTower
obs6: MaxEnergyHadTower
obs7: Tower Number
obs8: Highest Et Tower / Et Jet
obs9: Sum(E Twr * DeltaR(Twr-Jet)) / Sum(E Twr)

Track Variables
     only if there are tracks associated with the Jet
 obs10: Highest TrackPV Pt / Pt Jet
 obs11: Beta = Sum(Pt TrackPV) / Sum(Pt Track)
 obs12: Alpha = Sum(Pt TrackPV) / Pt Jet
 obs13: N Track PV
(if there are no tracks asoociated with the jet the value of the observables 10, 11, 12, 13 is
```

Talks

12 Feb 2008, Performance of the JetRejector Tool(JetMET POG)|

4 Oct 2007,http://indico.cern.ch/getFile.py/access?contribId=8&resId=1&materialId=slides&confId=22133
12 July 2007, Update on JetRejector Tool (JetMET POG)

Responsible: Ilaria Villella, Andrea Giammanco

-- MonicaVazquezAcosta - 13 Jul 2007

Jet Rejector Tool under MVA Framework

Introduction

A new implementation of Jet Rejector Tool is done using the SWGuideMVAFramework. By replacing the JetRejCombLR_SoverSplusBLoop and JetRejComLR with the new implementation, one can not only fully utilize the discriminants provided by the MVA framework but still keep the flexibility of the original JetRejector tool. An effort of validating this new implementation is also undertaken.

How to get the code

This implementation still keeps the original JetRejObsProducer. The code can be obtained from the CMS CVS.

```
scramv1 project CMSSW CMSSW_1_6_9
cd CMSSW_1_6_9/src
cmscvsroot CMSSW
eval `scramv1 runtime -csh`

cvs co -r V00-01-00 -d PhysicsTools/JetRejectorTool UserCode/YuanChao/PhysicsTools/JetRejector
cvs co -r btag_CMSSW_1_6_X_backport_branch CondFormats/PhysicsToolsObjects
cvs co -r btag_CMSSW_1_6_X_backport_branch PhysicsTools/MVAComputer
cvs co -r btag_CMSSW_1_6_X_backport_branch PhysicsTools/MVATrainer
cvs co -r TQAF_160_070908 AnalysisDataFormats/TopObjects
cvs co -r TQAF_16X_2007 TopQuarkAnalysis/TopTools
cvs co -r V00-05-00 PhysicsTools/JetMCAlgos
scramv1 b
```

1) Calculate the input variables: The user choose the signal definition using the appropriate configuration files and generate the variables mentioned in the 1) step above.

2) The Training phase: One need to write an XML file to select the input variables and discriminant algorithm. An example (JetRejMVATrainer.xml) is available under JetRejectorTool/bin which using the same input and algorithm as the original one.

```
> # modify JetRejMVATrainer_genjet.cfg to meet your need and point to the file generated in 1
> eval `scramv1 runtime -csh`
> cmsRun JetRejMVATrainer_genjet.cfg
```

A definition file containing the input PDFs will be generated as JetRejMVACom_genjet.mva. This file is needed for discriminant calculation in your analysis.

Several .root files will be generated by the MVA framework. One can get the input PDFs and discriminant performance plots using the ROOT program ViewMonitoring.C. A copy of this program is also available in the CVS.

```
> root -l ViewMonitoring.C
```

3) Get the output of the MVA discriminant.

```
# go to test/
cmsRun JetRejMVAEval.cfg
```

This will store the MVA discriminant into a new root file. One can also refer to the code src/JetRejMVAEval.cc to directly calculate in your analysis code. Note the the trained .mva file has to be properly set in the .cfg file. Details can be found the tutorial of MVAComputer.

-- YuanChao - 27 May 2008

-- YuanChao - 12-Mar-2010

This topic: Main > YuanChaoJetRejTool
Topic revision: r1 - 2010-03-12 - YuanChao