TMVA charm tagger training and validation

This document explains the workflow of training and validating the TMVA based charm tagger for CMS. It starts assuming that specialized samples (ntuples) containing jet properties have been extracted from AOD samples. Examples (for ttbar) of such files can be found in for example (on the local servers at the VUB!! not accessible to everyone.):
'/user/smoortga/Ctag/TMVActag_v1/Non_Flat_TrainingTrees/CMSSW_7_5_1/TTJets/*.root'

The procedure of training and validating the charm tagger requires some preparation and involves the following steps:

1. Flattening the samples = remove all vector-like branches in the trees and assign a separate branch to every vector-element, such that all branches represent exactly one value of a certain parameter

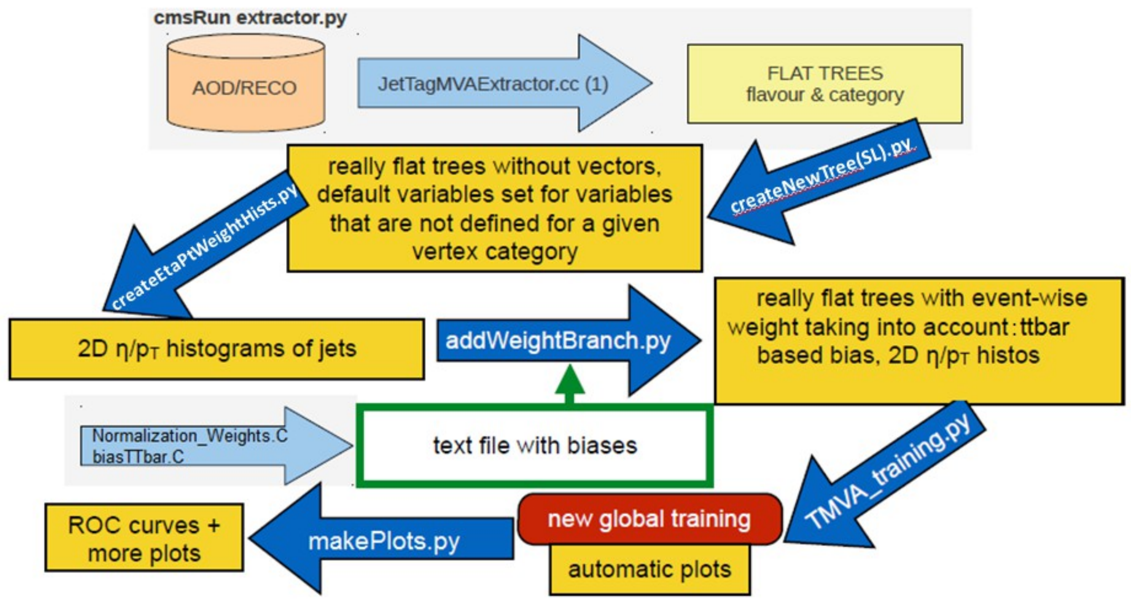
2. Adding weights to the event:
   - apply biases from the QCD training sample distributions to the Ttbar validation sample distributions
   - Make the distributions in pT and Eta flat to make the training as independent as possible of the kinematics of the training sample

3. Perform the training of the MVA + Validate the performance of the MVA

4. Produce the necessary output plots

This workflow is illustrated in the figure below

Each of the steps with the relevant scripts will be explained in more detail below. The scripts can all be found on my GitHub page: https://github.com/smoortga/CTagTraining/


cmsRun extractor.py
AOD/RECO
JetTagMVAExtractor.cc (1)
FLAT TREES flavour & category
createNewTree(SL).py
really flat trees without vectors, default variables set for variables that are not defined for a given vertex category
createEtaPtWeightHists.py
2D η/pT histograms of jets
addWeightBranch.py
really flat trees with event-wise weight taking into account: ttbar based bias, 2D η/pT histos
Normalization_Weights.C
biasTTbar.C
text file with biases
TMVA_training.py
new global training
automatic plots
makePlots.py
ROC curves + more plots

Step 1: Create ("Really") Flat Trees

Files: createNewTreeSL.py or createNewTree.py (if *no soft lepton information* is in the trees)

Purpose: remove all vector-like branches in the trees and assign a separate branch to every vector-element, such that all branches represent exactly one value of a certain parameter.

Extra Info:
Typically a vector branch named "variable" is split in three (this number is configurable in the file) separate branches called "variable_0", "variable_1" and "variable_2", containing each one of the *three most displaced tracks* in the vector.
If a variable is not set in a certain jet (for example no SV information is available for jets in the NoVertex Category, or if a vector has less than 3 elements/tracks), a default value (also configurable in this script) is assigned to it such that for each jet, each branch of the three contains exactly one value.

Guidelines:
Define all the variables in the non-flat trees with their type (preferably floats), default value and number of "flat" branches in case of a vector variable. This can be done starting from line
https://github.com/smoortga/CTagTraining/blob/master/scripts/createNewTreeSL.py#L205

Define all samples for either
QCD : https://github.com/smoortga/CTagTraining/blob/master/scripts/createNewTreeSL.py#L133
or
Ttbar: https://github.com/smoortga/CTagTraining/blob/master/scripts/createNewTreeSL.py#L168
And comment out the one (Ttbar or QCD) you don't need. You have to do this once for QCD and once for Ttbar.
Remark: This script assumes that all QCD samples follow the naming convention :
skimmed_20k_eachptetabin_CombinedSV[VERTEXCATEGORY][SOFTLEPTONCATEGORY]_[FLAVOUR].root
and the Ttbar samples follow the naming convention:
skimmed_CombinedSV[VERTEXCATEGORY][SOFTLEPTONCATEGORY]_[FLAVOUR].root

This script runs in parallel jobs each handling a maximum of 25000 jets. This leads to separate output files with event-numbers as a suffix. The script is designed to Merge all the final files using a "hadd" command
For QCD use :
https://github.com/smoortga/CTagTraining/blob/master/scripts/createNewTreeSL.py#L307
For Ttbar use :
https://github.com/smoortga/CTagTraining/blob/master/scripts/createNewTreeSL.py#L309
This is designed to end up with a total of 9 output files (3 vertex categories x 3 flavours, but inclusively for all Soft Lepton categories):

NoVertex_B	NoVertex_C	NoVertex_DUSG
PseudoVertex_B	PseudoVertex_C	PseudoVertex_DUSG
RecoVertex_B	RecoVertex_C	RecoVertex_DUSG

The name of the output directory is defined in (preferably separate QCD and TTbar):
https://github.com/smoortga/CTagTraining/blob/master/scripts/createNewTreeSL.py#L126

Step 2: Calculate Normalization weights and Ttbar biases

Files: Normalization_Weights.C and biasTTbar.C

Purpose: Calculate the normalization weights and Ttbar bias weights as explained in my thesis section 3.4.4. (paragraph ttbar bias weights):
http://mon.iihe.ac.be/~smoortga/Master_Thesis/MSc_Thesis_Seth_Moortgat.pdf

Extra Info:
This script creates a text file for each SV_category and flavour in which the weights are given per pT-eta bin. In the terminal it also spits out all of these weights one under the other. Simply copy paste this combined output and put it in a .txt file named (follow this convention for further use in scripts):
QCD_norm.txt For the Normalization weights or
ttbarBias.txt For the Ttbar bias weights
After having created these two files you can actually remove all the separate automatically generated .txt files (This should of course be resolved in the future and should immediately create one file)

Valere had a file that calculated these weights also for each SoftLepton category separately (to end up with 27 SL-SV-Flavour categories instead of 9 SV-Flavour categories). In this case you don't want to merge the files the way I do in step one and leave the Soft Lepton categories non-merged.

Guidelines:
Define the place of the samples to calculate the weights from (see output of Step 1):
https://github.com/smoortga/CTagTraining/blob/master/scripts/biasTTbar.C#L27
https://github.com/smoortga/CTagTraining/blob/master/scripts/Normalization_Weights.C#L25

Step 3: Add Normalization weights and Ttbar bias branches to the flat trees

Files: https://github.com/smoortga/CTagTraining/blob/master/scripts/addWeightBranch.py

Purpose: Add branches to the flat trees containing the weights from the *QCD_norm.txt* and *ttbarBias.txt* files.

Extra Info:
This script reads in the *QCD_norm.txt* and *ttbarBias.txt* files and for every jet will add a value to a branch containing the appropriate weight (based on the SV category, flavour, pT and Eta of the jet).

The weights for the Normalization weights are put in a branch called *"weight_norm"*and the ttbar bias weights are put in a branch called *"weight_category"*.

There is also a flavour weight (*"weight_flavour"*)based on the relative occurrence of each jet flavour in ttbar events
B:C:DUSG = 2:1:3
see : https://github.com/smoortga/CTagTraining/blob/master/scripts/addWeightBranch.py#L161
This makes each flavour roughly equally likely to occur.

Any other weights in this file (such as Pt-Eta weights) are not yet filled (see next steps).

Guidelines:
Define the name of the directory containing the flat trees (input for this script) in
https://github.com/smoortga/CTagTraining/blob/master/scripts/addWeightBranch.py#L153

Define the name of the directory (temporary directory) in which you will store the output trees (with the weight branches) in:
https://github.com/smoortga/CTagTraining/blob/master/scripts/addWeightBranch.py#L151

Define the names of the Normalisation and Ttbar biases .txt files in
https://github.com/smoortga/CTagTraining/blob/master/scripts/addWeightBranch.py#L154
These should be in the same directory as this script and if you followed the naming convention in the previous step this should already be ok.

In the end you will have a copy of the flat trees containing the weight branches in the specified (temporary) directory.

Step 4: Create histograms containing the weights for all jets in bins of pT and Eta.

Files: https://github.com/smoortga/CTagTraining/blob/master/scripts/createEtaPtWeightHists.py

Purpose: Create histograms containing the weights to make the distributions of jet-pT and jet-Eta flat.

Extra Info:
This script creates 2D histograms in pT and Eta bins which contain the weights to make these distributions flat (adding these weights to the flat trees only happens in the next step).

The histograms are named after the files that are used to create them with an extension "_EtaPtWeightHisto"

Guidelines:
Define the name of the input trees (which are the output trees of the next step containing the Normalization weights and the ttbar bias weights) in:
https://github.com/smoortga/CTagTraining/blob/master/scripts/createEtaPtWeightHists.py#L81

define the name of the output directory for the histograms in :
https://github.com/smoortga/CTagTraining/blob/master/scripts/createEtaPtWeightHists.py#L79
(line 80 can just point to the same directory. It will store a histogram combining all the SV catogries for a given flavour.)

Step 5: Add pT-Eta weight branches to the flat trees

Files: https://github.com/smoortga/CTagTraining/blob/master/scripts/addEtaPtWeightBranch.py

Purpose: For each jet look in the histograms created in the previous step and add a weight (depending on the jet pT and Eta of course) to a new branch in the flat trees.

Extra Info:
This script Is very similar to the one from step 3. Since now you have all the weights available (Normalization, Ttbar bias, Flavour, Pt-flat and Eta-flat), this script takes as input again the flat trees WITHOUT ANY WEIGHTS IN THEM and adds all the weight branches to these flat trees. This indeed again adds the weights from step 3, but it was easier to not change the script and add all the weights from scratch.

The output flat trees contain all the necessary weight branches and combines them (just multiply all the weights in
https://github.com/smoortga/CTagTraining/blob/master/scripts/addEtaPtWeightBranch.py#L127) and adds them to a general weight branch called *"weight"*. This is the final jet-pet-jet weight needed in the training.

Guidelines:
Define the name of the directory containing the flat trees WITHOUT ANY WEIGHT BRANCHES in https://github.com/smoortga/CTagTraining/blob/master/scripts/addEtaPtWeightBranch.py#L153

Define the name of the directory (temporary directory) in which you will store the output trees (with the weight branches) in:
https://github.com/smoortga/CTagTraining/blob/master/scripts/addEtaPtWeightBranch.py#L151
These trees will serve as input to the training

Define the names of the Normalisation and Ttbar biases .txt files in
https://github.com/smoortga/CTagTraining/blob/master/scripts/addEtaPtWeightBranch.py#L154
https://github.com/smoortga/CTagTraining/blob/master/scripts/addEtaPtWeightBranch.py#L155
These should be in the same directory as this script and if you followed the naming convention in the previous step this should already be ok.

Define the name of the directory that contains the pT-Eta 2D histograms in:
https://github.com/smoortga/CTagTraining/blob/master/scripts/addEtaPtWeightBranch.py#L152

In the end you will have your final flat trees that serve as input to the training.

Step 6: Training + validating the charm tagger

Files: https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py

Purpose: Train the TMVA charm tagger on the QCD samples and Validate on the Ttbar samples.

Extra Info:
The output of this script will create root files containing all the discriminator distributions, variable distributions etc…

These files are called *“trainPlusBDTG_CombinedSV[SVCategory]_[Flavour].root”*

Guidelines:
Define the variables you want to use in your MVA in
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L9
Floats and Ints seperately (but for now we work with all Floats)

Define your QCD input trees for the training (output from the previous step) in:
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L83

Define Signal (flavour == 4 for charm tagger) in
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L99
and background (either 5 for B or !=4 && !=5 for light) in
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L100

The weight branch is (already correctly) set in
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L120

The MVA method is bookmarked in
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L131

Define the directory name of your Ttbar input trees for the validation in:
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L295
and the names of those files underneath that.
Note: for CvsDUSG you might want to validate only on C and DUSG samples since the B samples don't interest you… But it you want to make 2-dimensional performance curves you also need discriminator distributions for B jets so then you have to validate (and train) on these as well.

Define the options of your MVA method in
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L330

The training is executed on line
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L341
and the validation on line
https://github.com/smoortga/CTagTraining/blob/master/scripts/tmva_training.py#L343

The training .xml files are stored in a subdirectory of your working directory named *“./weights/”* and are named something like *“TMVAClassification_BDTG.weights.xml”*

Step 7 (optional): Draw some performance curves and discriminator distributions

Files: https://github.com/smoortga/CTagTraining/blob/master/scripts/makePlots.py

Purpose: Create root files containing Graphs and histograms with discriminator distributions and performance curves.

Extra Info:
These output root files are saved in a subfolder of the current working directory called *“./histos/”*. The

Combined (merged) file in which you are probably interested is called *“AllHistograms.root”*.

Guidelines:
No configuration is in principle needed.

Step 8 (optional): Draw some Overlays of performance curves or discriminator distributions.

Files: https://github.com/smoortga/CTagTraining/blob/master/scripts/DrawOverlays.C
https://github.com/smoortga/CTagTraining/blob/master/scripts/DrawOverlays.h

Purpose: Some personal script for drawing overlays of performance curves or discriminator distributions.

Extra Info:
This script just reads some *“AllHistograms.root”* files and draws some overlays. You can write your own if you want but you can use my script if you like it :-)

Guidelines:
See file itself. You need to define some paths to the AllHistograms.root files and some legends if you like. It is rather straight forward.