Machine Learned Tracklet Filters - Update

Rudolf Frühwirth, Jakob Lettenbichler, Thomas Madlener

HEPHY Institute of High Energy Physics

ÖAW Austrian Academy of Sciences

02.09.2015

What happened since Vienna?

- Development of tools for testing feasibility of incorporating advanced machine learning (ML) approaches into SectorMap approach of VXDTF
- Testing of different ML classifiers
  - Multilayer Perceptron
  - Boosted Decision Trees
- Started writing of Thesis

Three Hit Filters: Planned Approach

Replace/Support three hit filters with a ML classifier

- use three hit combinations passing the two filter stage of the VXDTF as inputs
  - $\rightarrow$ SNR $\approx$ 0.75 after two hit filters
- ML classifier labels input as signal or background

Possible Advantages:

+ Better separation of signal and background compared to current approach
  - $\rightarrow$ Reduced combinatorics in later stages
+ Generalization capabilities require reduced amount of
  - training samples
  - sectors / SectorMaps

ML Classifiers

Tested classifiers:

Multilayer Perceptron (MLP)

- one hidden layer with different numbers of neurons
- different activation functions of output neuron:
  - logsig: logistic function $f(z) = (1 - e^{-t})^{-1}$
  - linear: linear function $f(z) = z$
- done with MATLAB

Boosted Decision Trees (BDT)

- different Boosting algorithms:
  - *AdaBoost* (MATLAB)
  - *Stochastic Gradient Boosting* (FastBDT, BASF2)
- different tree depths / numbers of decision splits
- different numbers of boosting steps

Generation of Training and Test Samples

Plan:

- use SegmentNetwork to get samples/inputs
- feed tracklets from SegmentNetwork to classifier $\rightarrow$ classifier is a 'pluggable' substitute to current filters
- in a first step use SVD only

But:

- not yet ready in framework
- 'misuse' current VXDTF to generate samples

classifier - machine learned instance (BDT, MLP) with output that makes classification into background/noise and signal possible

Generating Samples with current VXDTF

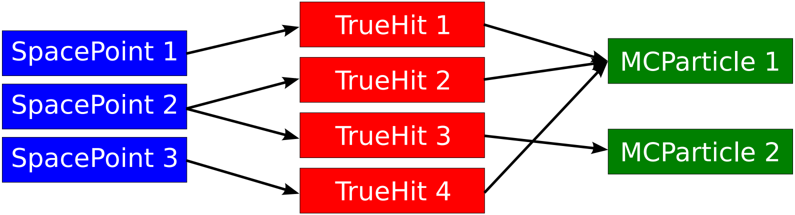
1. simulate generic events with background
2. VXDTF to get track candidates:
    - enable only two hit filters
        - *distance3D*
        - *distanceXY*
    - disable filtering/cleaning of overlapping track candidates (i.e. disable Hopfield network or greedy algorithm)
    - tune CutOff Values by 6 % (`tuneCutOffs: 0.06`)
3. convert to SPTCs for further processing
    - disable usage of single Cluster SPs (need global position)
4. create three hit samples from SPTCs
    - split SPTC into tracklets containing three SpacePoints each

Properties of three hit samples (tracklets)

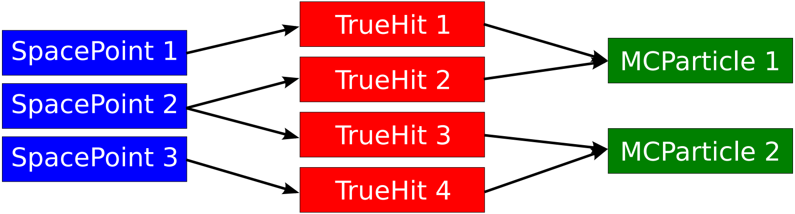
- hits are on consecutive layers (i.e. no overlapping parts at the moment)
- if all SpacePoints have relation to the *same* MCParticle $\rightarrow$ signal sample, else background/noise sample
- relations to other MCParticles are not considered
- SpacePoints with no relation to any MCParticle $\rightarrow$ background/noise sample

Input of classifiers:
global coordinates of SpacePoints $\rightarrow \mathbf{x} \in \mathbb{R}^9$

signal sample:



background/noise sample:

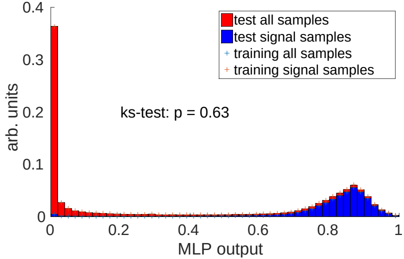


Data Sets and Preprocessing

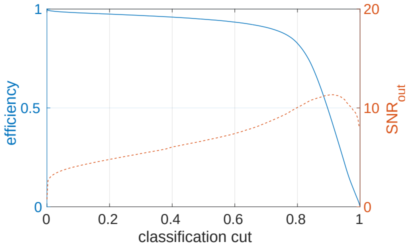
The whole data set is split up in a training set and a testing set (not used in training at all)

- For comparable results the same training and testing sets are used for all classifiers.
- Still some randomness in training (network initialization, random splits in stochastic gradient boosting)
- Input data is decorrelated before splitting (negligible difference)
- Comparison of output distributions of both sets used to check if a classifier is overtrained

Determining Classification Cut



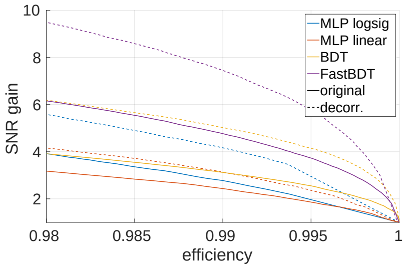
Exemplary output distribution of a MLP.



efficiency and SNR in output (SNR_{out}) depending on the applied classification cut

SNR_{out} - ratio of true positives to false positives

Comparison of different classifiers



SNR gain vs. efficiency for different tested classifiers

$$\textit{SNR gain} = \mathrm{SNR}_{\mathrm{out}} / \mathrm{SNR}_{\mathrm{in}}$$

Classifiers:

- **MLP logsig** - 50 hidden neurons, logsig output
- **MLP linear** - 50 hidden neurons, linear output
- **BDT** - 50 decision splits, 2000 boosting steps, *AdaBoost*
- **FastBDT** - tree depth 6, 2000 boosting steps

Comparison of different classifiers

→ Decorrelating improves performance of all tested classifiers by a factor of approx. $1.3 - 1.6$

→ BDTs (including FastBDTs) generally perform better than MLPs (at least with 50 hidden neurons)

→ evaluation time rules out BDTs trained with *AdaBoost* (table below)

[μs/sample]	training	evaluation
MLP w/ $H = 50$	$\sim 2400 - 3400$	$\sim 2.1 - 2.4$
BDT w/ $D = 50, N = 2000$	$\sim 10^4$	$\sim 10^3$
FastBDT w/ $N = 2000$	$\sim 250 - 270$	$\sim 10 - 10^2$

NOTE: MLP tested with MATLAB → evaluation times probably do not translate to BASF2

Performance Analysis

Perform a more detailed analysis of the classifiers to spot possible weak (or sweet) spots

- θ- and ϕ-dependent performance
- p- and p_T-dependent performance
- charge and PDG code dependent performance

Performance Analysis

Prerequisites for following analysis:

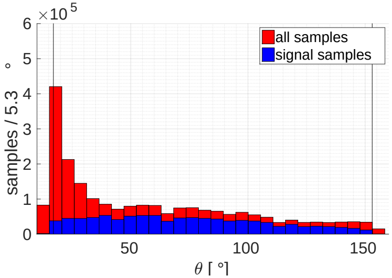
- Only one (global) classification cut determined from overall performance, such that overall efficiency is ≥ 0.99
- No MC information available for background samples
  $\rightarrow$ only efficiency can be analyzed

Main Result:

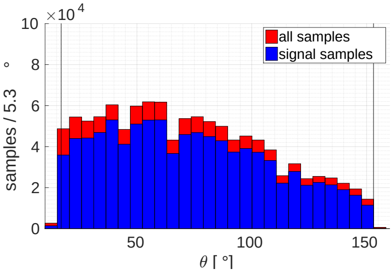
$\rightarrow$ all tested classifiers show qualitatively same characteristics

$\rightarrow$ following plots obtained with best performing FastBDT (tree depth = 6, 2000 boosting steps)

θ-dependent performance

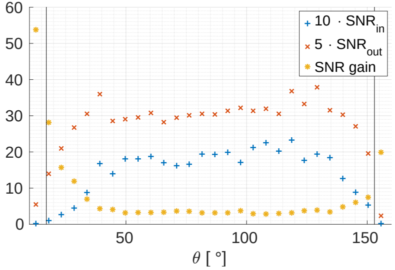


input in bins of θ

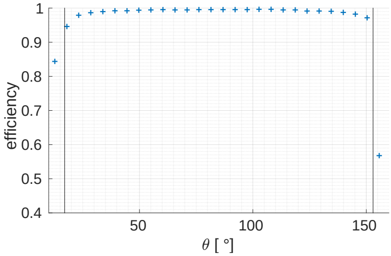


output in bins of θ

θ-dependent performance



SNR in input and output and ratio in bins of θ

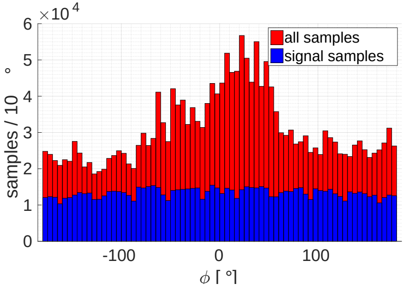


efficiency in bins of θ

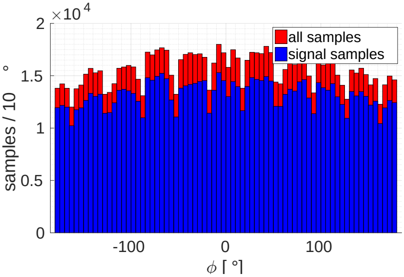
θ-dependent performance

- → Efficiency stable over wide range, dropping below 0.99 at the edges only
- → Efficiency below 0.9 only at θ outside of official detector boundaries
- → SNR gain stable at approx. $3 - 4$ for wide range reaching up to almost 30 for forward direction with high background
- → choosing cuts such that each bin has 0.99 efficiency yields similar results with reduced SNR gain at the edges

phi-dependent performance

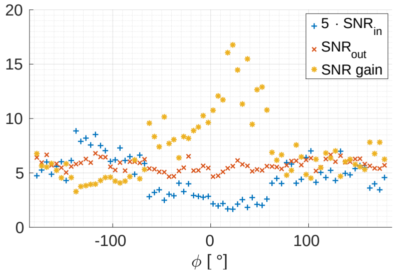


input in bins of ϕ

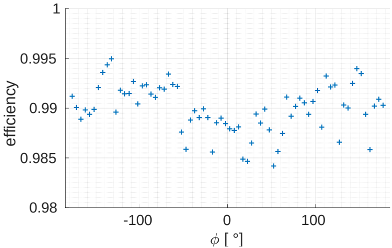


output in bins of ϕ

ϕ-dependent performance



SNR in input and output and ratio in bins of ϕ

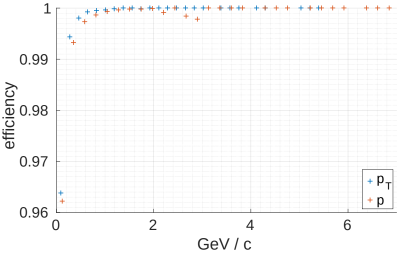


efficiency in bins of ϕ

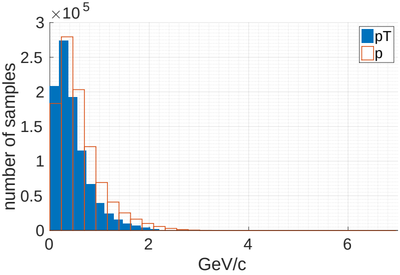
ϕ-dependent performance

- → Efficiency stable ≥ 0.98 over whole range
- → SNR_{out} stable over whole range
- → SNR gain with broad peak around $\phi \approx 40$ due to high background in input there
  - → unclear if this is due to SectorMap or stems from simulation
  - → naively expected an almost flat distribution as input
- → dips in output at overlapping parts of layer 4
  - → hits in overlapping parts excluded
  - → Why from layer 4?
- → choosing cuts such that efficiency is 0.99 in all bins has no significant effects

p- and p_T-dependent performance



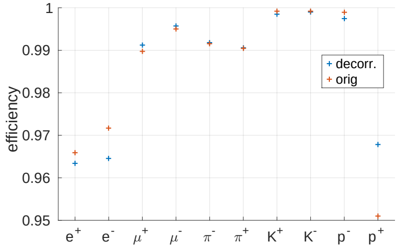
efficiency in bins of p and p_T



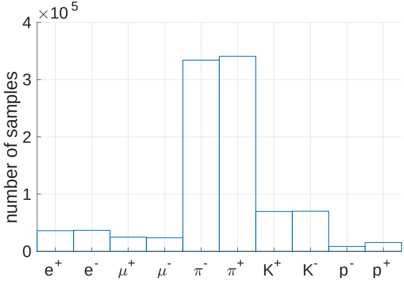
occupancy in bins of p and p_T

- $\rightarrow$ efficiency ≥ 0.95 for all values of p and p_T
- $\rightarrow$ only first bin ($p_T = 0.1\,\mathrm{GeV/c}$, $p = 0.12\,\mathrm{GeV/c}$) below 0.99 efficiency

charge and PDG code dependent performance



efficiency depending on particle



occupancy for different particles

- → Efficiency higher for neg. charged particles although number of pos. and neg. charged particles almost balanced
- → Effect is bigger for decorrelated data
- → lowest efficiency for e^{-}/e^{+}

Summary Outlook

Summary

- → BDTs (incl. FastBDTs) with better classification performance compared to MLPs
- → **Decorrelation** of inputs improves performance of all tested classifiers significantly
- → Performance looks promising however no real prediction possible on the impact on the VXDTF

Open Question

- → Why is input not flat in ϕ?
- → Why is efficiency better for neg. charged particles?
- → How does this effect the VXDTF?

Summary Outlook

Next Steps

- → Check input distribution in ϕ with particle gun instead of generic events and without background to discern 'external' sources
- → Check performance in cases where hits are not on consecutive layers
- → Once SectorMap is ready check how ML filter can be implemented and test effects
- → Continue writing theses
- → ... Your Suggestions / Requests

Thank You

Questions or Remarks?