Identification of Hadronically Decaying Tau Leptons at the ATLAS Detector Using Artificial Neural Networks

Nico Madysa

IKTP Dresden

November 16, 2015

Introduction

The Task

signal: jets from hadronic tau decays

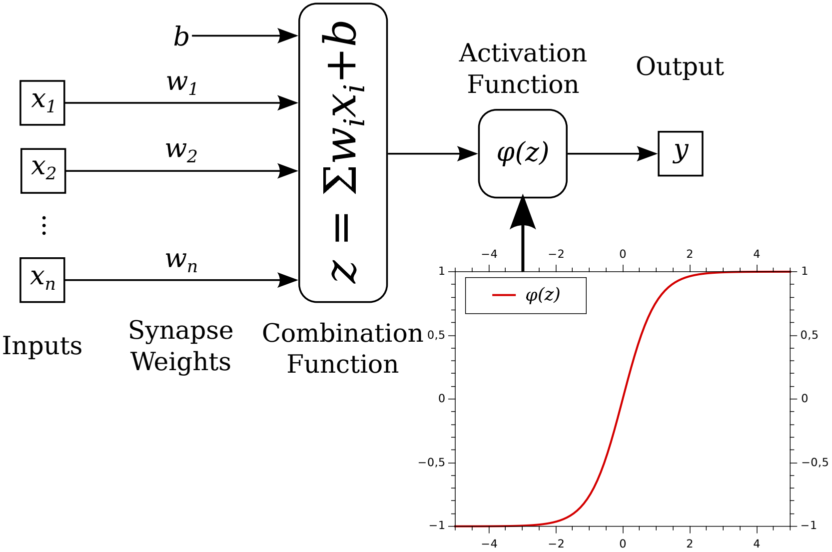
- more collimated than background

background: jets from pure QCD processes

- higher particle multiplicity than signal

Classification

- give each tau candidate a *tau score* $\in [0;1]$
- $0 \simeq$ background-like, $1 \simeq$ signal-like


b
w1
w2
wn
x1
x2
xn
z = ∑wixi+b
Activation
Function
Output
φ(z)
y
Inputs
Synapse
Weights
Combination
Function
φ(z)
−4
−2
0
2
4
1
0,5
0
−0,5
−1

ANNs/Artificial Neurons

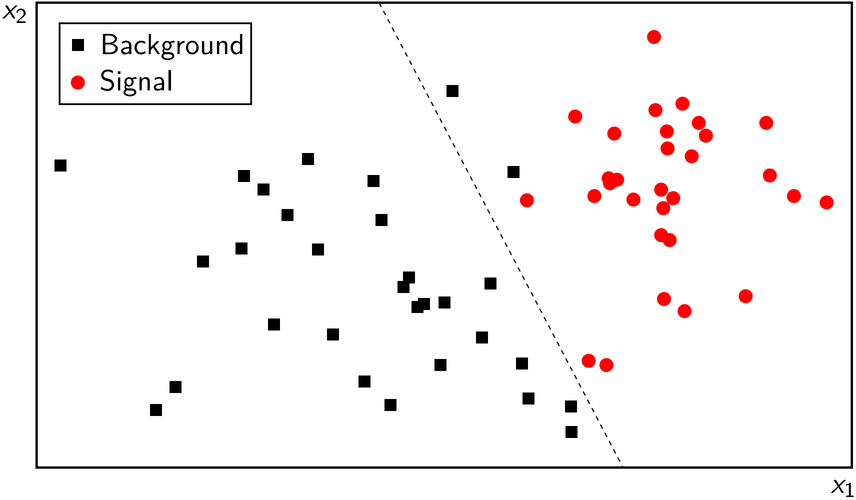


Figure: Artificial neurons solve linearly separable problems.

ANNs/Multi-Layer Networks

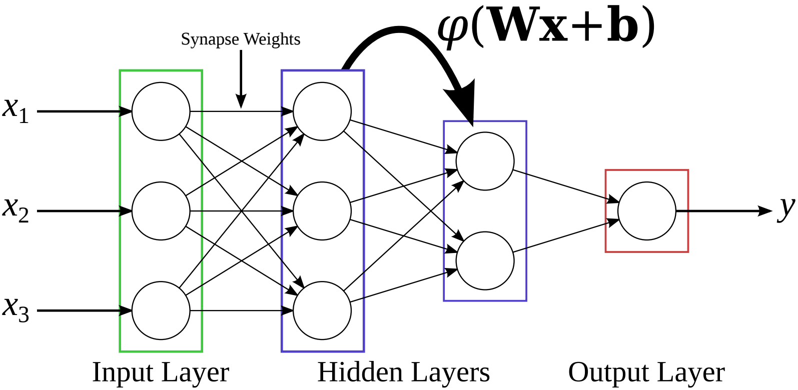


Figure: Each layer performs a linear transformation and then applies the activation function element-wise.

Training/Error Function

Error Function

- measures distance between desired and actual input
- e.g. mean-squared error: $E(\mathbf{w}) = \frac{1}{2}\sum_i (y(\mathbf{x}_i, \mathbf{w}) - t_i)^2$
- training = minimization of $E(\mathbf{w})$

Challenges

- many dimensions: $N_{\text{weights}} \in \mathcal{O}(10^3)$
- many local minima

Training/Procedure

Procedure

1. start with random initial synapse weights $\mathbf{w}_0$

   $\Rightarrow$ breaks up symmetries of $E(\mathbf{w})$

2. minimize $E(\mathbf{w})$ iteratively in *epochs* τ:
   - calculate gradient $\nabla E(\mathbf{w})$
   - use it to modify weights: $\mathbf{w}(\tau+1) = \mathbf{w}(\tau) + \Delta\mathbf{w}$

3. stop training if $E(\mathbf{w})$ stops decreasing

Training Algorithms

Gradient Descent: straightforward approach, $\Delta\mathbf{w} \sim -\nabla E(\mathbf{w})$

- vulnerable to local minima
- many parameters to be tuned correctly

SARPROP: modified gradient descent with additional statistical noise

- better results
- only one tunable parameter

BFGS: approximates Hessian of $E(\mathbf{w})$ using $\nabla E(\mathbf{w})$

- same performance as SARPROP
- no fine-tuning
- computationally more expensive

Training/Score Histogram

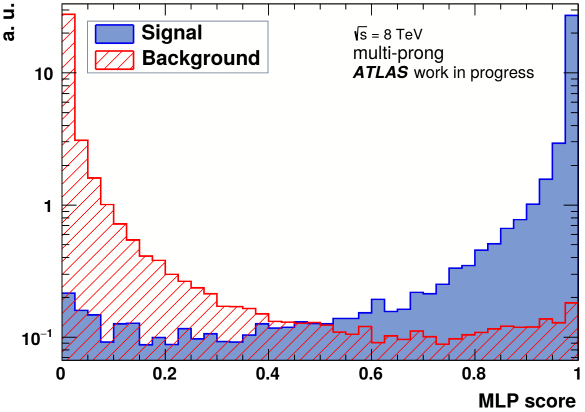


Figure: Score histogram of an ANN for multi-prong tau candidates. The peaks are very narrow and give numerical difficulties.

Instability of ANN Training

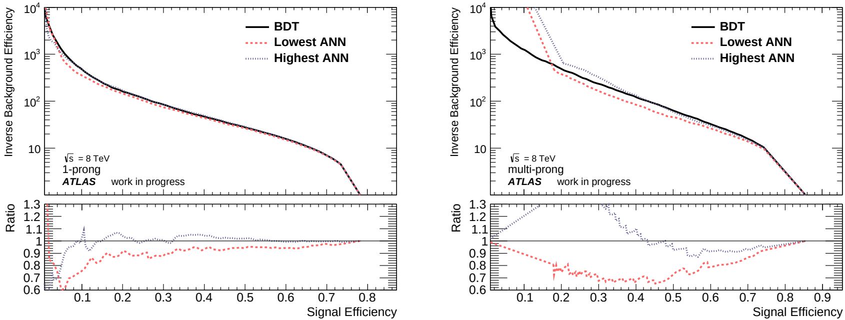


Figure: Best and worst ANN out of 50. The erratic behavior for $\varepsilon_{\text{sig}} < 0.3$ and multi-prong tau candidates is due to low statistics. (cf. previous slide)

Instability of ANN Training

Problem

- initial weights are chosen randomly

$\Rightarrow$ training ends in random local minimum

$\Rightarrow$ performance of the final classifier is random, too

Solution: Ensembles

- train N ANNs with different initial weights
- average their scores for each tau candidate: $y_{\mathrm{Ens}} = \frac{1}{N}\sum_{i=1}^{N} y_{\mathrm{ANN},i}$
- reduces variance due to initial weights
- also reduces variance due to overfitting(!)
- $N = 5$ gives good results, $N = 10$ is only marginally better

Instability of ANN Training

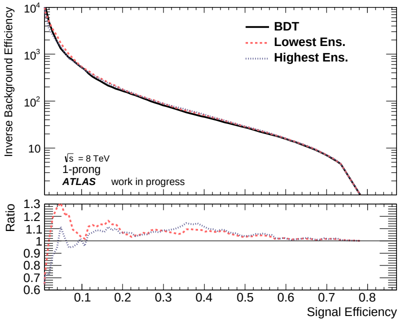


Figure: Best and worst five-member ANN ensemble. Standard deviation due to random initial weights is reduced by 71 % for 1-prong and by 47 % for multi-prong tau candidates.

Results/Optimization

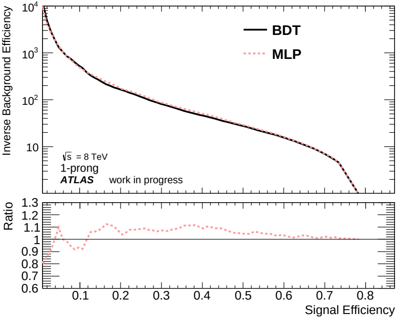
What affects the classifier performance?

- training algorithm
- error function $E(\mathbf{w})$
- network topology (number of hidden layers, neurons per hidden layer)
- activation function of the hidden layers

Procedure

- train and evaluate ANN ensembles for each configuration
- compare to theoretical predictions

Results/Final Classifier



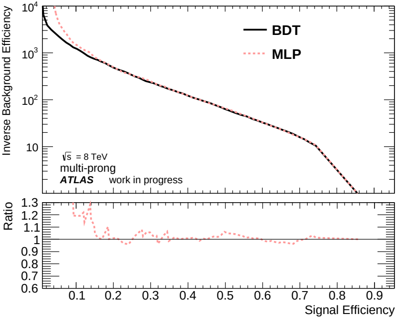


Figure: The relative difference between MLP and BDT is $+6.1\,\%$ (1-prong) and $-0.4\,\%$ (multi-prong).

Summary & Outlook

Summary

- first implementation and study of ANN-based tau identification
- comparison with BDT-based approach

  1-prong: improved by $\sim 6\,\%$

  multi-prong: no significant difference
- ensemble formation increased stability and performance
- optimization w.r.t.
  - training method
  - error function
  - hidden layer activation function
  - network topology

Summary & Outlook

Outlook

- larger training samples (especially for multi-prong)
- substructure variables
- look into more modern training algorithms
- more sophisticated ensembles (bagging, boosting)

Backup

Backup/Implementation

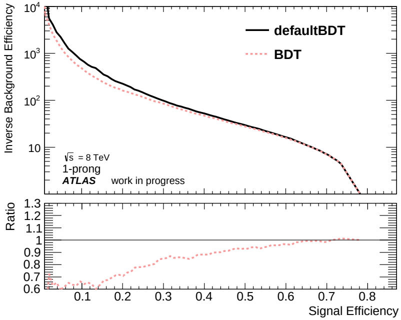
Setup

- ROOT 5.34
- TMVA's MLP method
  - supported training methods: gradient descent, BFGS
  - implemented ourselves: SARPROP
- wrapped in a Python package for parallel training of ANNs

Benchmark BDT

- new BDT trained on the same samples as the ANN for better comparison
- p_{T}- and n_{vtx}-reweighting, no cross-section weights
- p_{T}-dependent cut-off score

Backup/Implementation



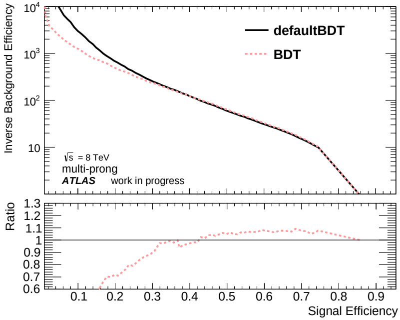


Figure: Comparison of the default and the benchmark BDT. The benchmark BDT is worse than the default, but allows more realistic comparison of ANNs.

Backup/Implementation/ID Variables

Variables for 1-prong

- tau_calcVars_corrCentFrac
- tau_calcVars_corrFTrk
- tau_pi0_vistau_m
- tau_ptRatio
- tau_seedCalo_trkAvgDist
- tau_pi0_n
- tau_ipSigLeadTrk
- tau_seedCalo_wideTrk_n

Variables for 3-prong

- tau_calcVars_corrCentFrac
- tau_calcVars_corrFTrk
- tau_pi0_vistau_m
- tau_ptRatio
- tau_seedCalo_trkAvgDist
- tau_pi0_n
- tau_massTrkSys
- tau_seedCalo_dRmax
- tau_trFlightPathSig

Table: Sample sizes and processes. (***ATLAS*** work in progress)

	Training/Validation set		Test set	
Process	1-prong	3-prong	1-prong	multi-prong
Signal samples				
$W \to \tau\nu_\tau$	24 065	7959	32 116	14 382
$Z \to \tau^+\tau^-$	26 684	7714	35 598	13 878
$Z'_{250} \to \tau^+\tau^-$	24 137	6718	32 237	11 069
$Z'_{500} \to \tau^+\tau^-$	24 713	6398	32 605	10 725
$Z'_{1000} \to \tau^+\tau^-$	25 239	5324	33 484	10 588
Total	124 838	34 113	166 040	60 642
Background sample				
`PeriodD`	66 887	86 165	89 913	259 552

Backup/Optimization/Training Algorithm

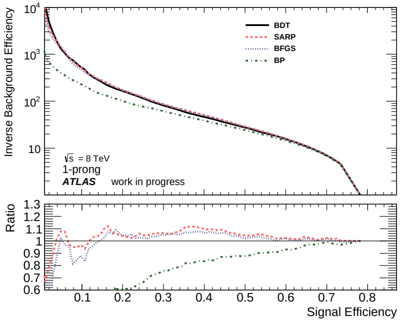


Figure: SARPROP and BFGS give about equal results. (But BFGS is very slow.) Stochastic gradient descent (BP) is much worse.

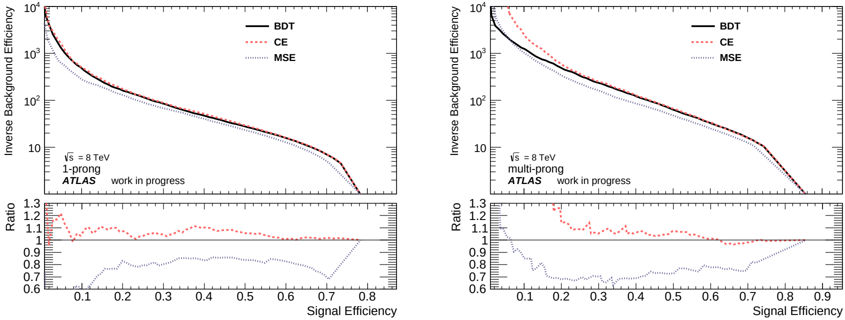


Figure: Comparison of mean-squared-error (MSE) with cross-entropy (CE) error function. CE is better because it's specialized for binary classification tasks.

Backup/Optimization/Activation Function

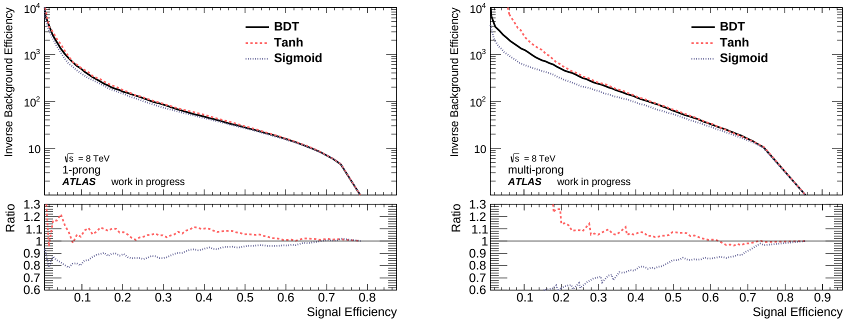


Figure: Comparison of `tanh` and logistic function (`sigmoid`) as hidden layer activation function. `Tanh` results in quicker training due to numerical reasons and thus gives better results than `sigmoid` when trained for the same amount of epochs.

Backup/Optimization/Network Topology

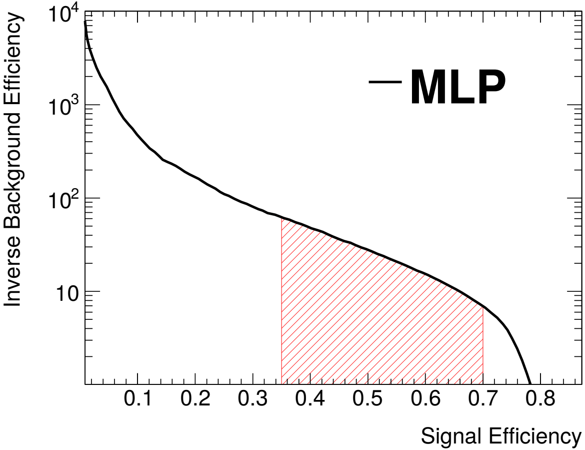


Figure: $\mathrm{AUC} = \int_{0.35}^{0.70} (\varepsilon_{\mathrm{bkg}})^{-1} \mathrm{d}\varepsilon_{\mathrm{sig}}$ is a good figure of merit.

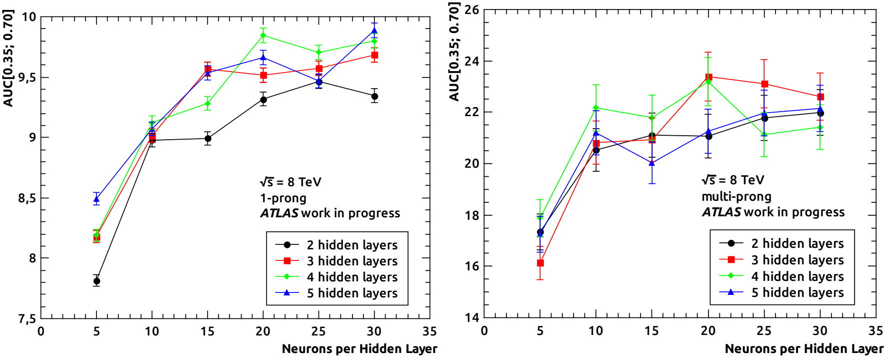


Figure: Comparison of different network topologies. The error bars give the empirical standard deviation of ANN ensembles. The optimal topology is 4 layers with 20 hidden neurons each.

Backup/Optimization/Overfitting

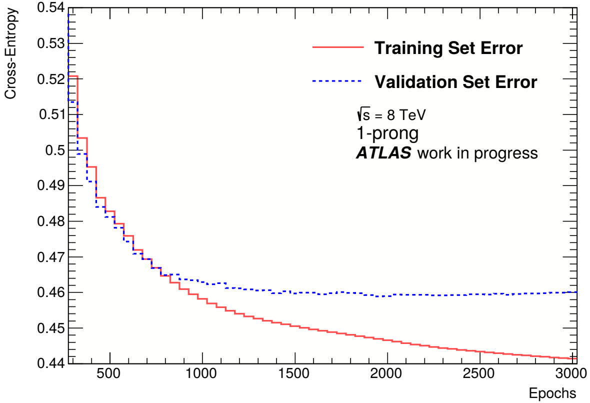


Figure: Slight overfitting is acceptable since the ensemble averages it out. (Shown: Error function during training on 1-prong sample)