Loss Functions for BDTs in TMVA

By Andrew Carnes

Intro

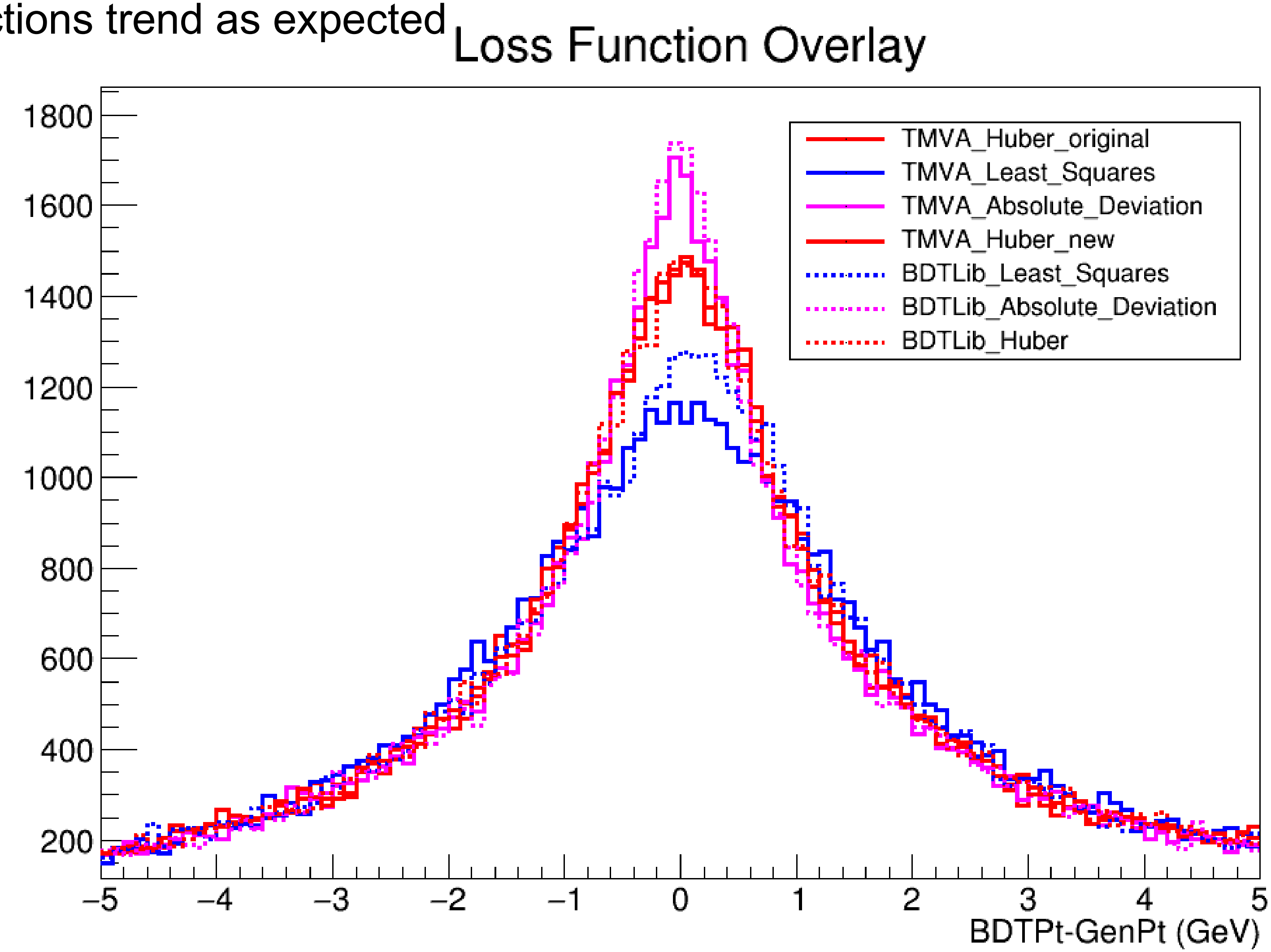
- Wrote a Boosted Decision Tree (BDT) package, BDTLib
  https://github.com/acarnes/bdt
    - Multiple loss functions
    - Consensus to integrate this functionality into TMVA
- Would also like to parallelize the BDTs
- Done with multiple loss function implementation

This Time

- **Message of this talk:** Implemented multiple Loss Functions into TMVA
    - Huber, Least Squares, Absolute Deviation
- Present new functionality on an available dataset
- Then future plans

Benchmarking on CSC Pt Assignment

- TMVA Huber Loss Function before and after work the same
- Loss functions for Huber, and Absolute deviation match almost exactly b/w TMVA/BDTLib
- Least Squares differs between TMVA and BDTLib
    - Why? BDTLib and TMVA choose split points differently
    - BDTLib doesn't work based on max depth, just max terminal nodes
- Anyways, loss functions trend as expected



Access New Functionality

- Simply choose the loss function in the options string
- factory->BookMethod(dataloader, TMVA::Types::kBDT, "BDTG", "!H:V:NTrees=64::BoostType=Grad:Shrinkage=0.3:nCuts=99999:MaxDepth=4: MinNodeSize=0.001:NegWeightTreatment=IgnoreNegWeightsInTraining: PruneMethod=NoPruning:**RegressionLossFunctionBDTG=AbsoluteDeviation**");
- Choose **AbsoluteDeviation, LeastSquares, or Huber**
- **Huber is default as before**
    - Huber has a parameter that you can set that determines the cutoff for the core and the tails of the distribution
    - Use option “HuberQuantile=0.8”, default value is 0.7 as before
    - For 0.8, the first 80% of the residuals will be the “core” and the last 20% will be the tails

Future Plans

- Need to run further unit tests
- Will make a notebook exemplifying the new capabilities
- Plans to parallelize the BDTs in TMVA
    - Can search for the best cuts along each feature in parallel
    - Can reduce the BDT training time by a factor of the number of features
    - Can also parallelize the evaluation since the contribution from each tree doesn't depend on any of the others

Backup Slides

- BDT Algorithm Overview
- References

Brief BDT Algorithm Overview

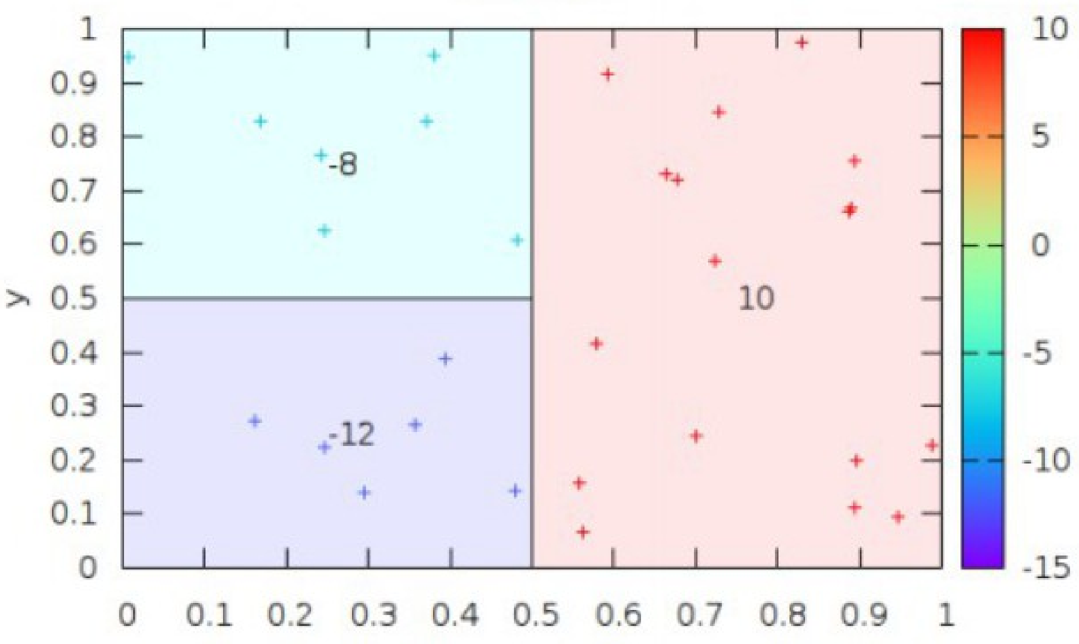


Fig 1. A decision tree with 3 terminal nodes

A Single Decision Tree

- Breaks up feature space into discrete regions using hyperplanes
- Fits a constant to each region
- The regions are greedily chosen to minimize a given Loss Function (a differentiable measure of the error)
- May be viewed as a series of decisions (shown below)

Boosting

- Make one tree, add another tree that corrects the predictions of the first
- Add another tree that corrects the net prediction of the first and second
- Continue the process
- End up with a collection of trees (Forest) and a net prediction
- $F(x) = T_0(x) + T_1(x) + T_2(x) + \ldots + T_N(x)$

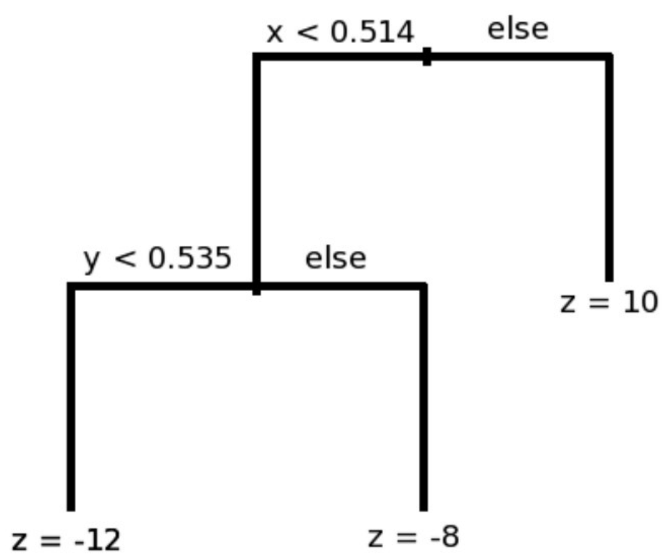


Fig 2. The same tree represented as a series of decisions

References

- Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.