MACHINE LEARNING IN HIGH ENERGY PHYSICS

Giles Strong

VBS Workshop, LIP-Lisbon - 05/12/2019

giles.strong@outlook.com

twitter.com/Giles_C_Strong

Amva4newphysics.wordpress.com

github.com/GilesStrong

OVERVIEW

1. Motivation for machine learning in searches
2. Overview of machine learning
3. Possible applications of NNs to VBS searches
4. Examples
5. Summary
6. Appendix: Getting started in ML

MOTIVATION FOR ML IN SEARCHES

ANALYSIS SENSITIVITY

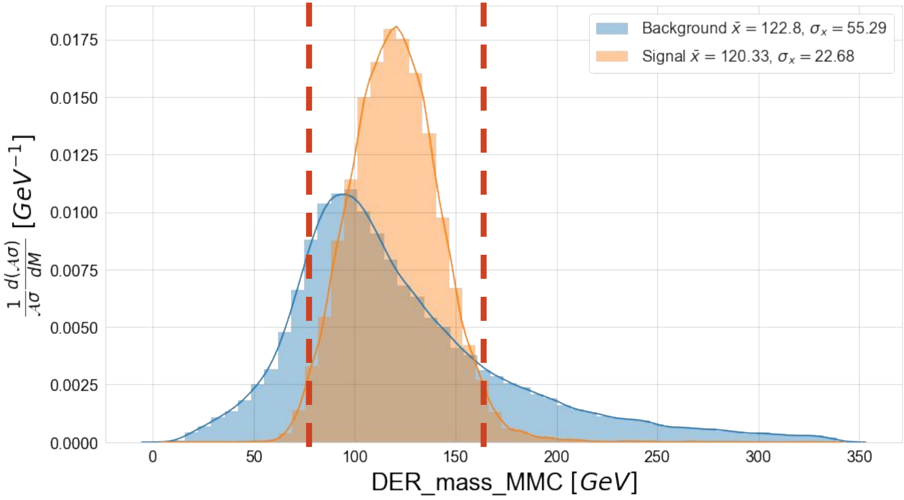
- Expected discovery significance can be approximated as $\frac{s}{\sqrt{b}}$
- Or, more accurately as the Approximate Median Significance

$$\text{AMS} = \sqrt{2\left(s+b\right)\log\left(\frac{\left(s+b\right)\left(b+\sigma_b^2\right)}{b^2+\left(s+b\right)\sigma_b^2}\right) - \frac{b^2}{\sigma_b^2}\log\left(1+\frac{\sigma_b^2 s}{b\left(b+\sigma_b^2\right)}\right)},$$

  - For expected signal and background yields of *s* and *b*
  - And background yield uncertainty σ_b
- I.e. sensitivity generally improved by defining region with higher signal to background ratio (subject to background uncertainty)

TRADITIONAL APPROACH

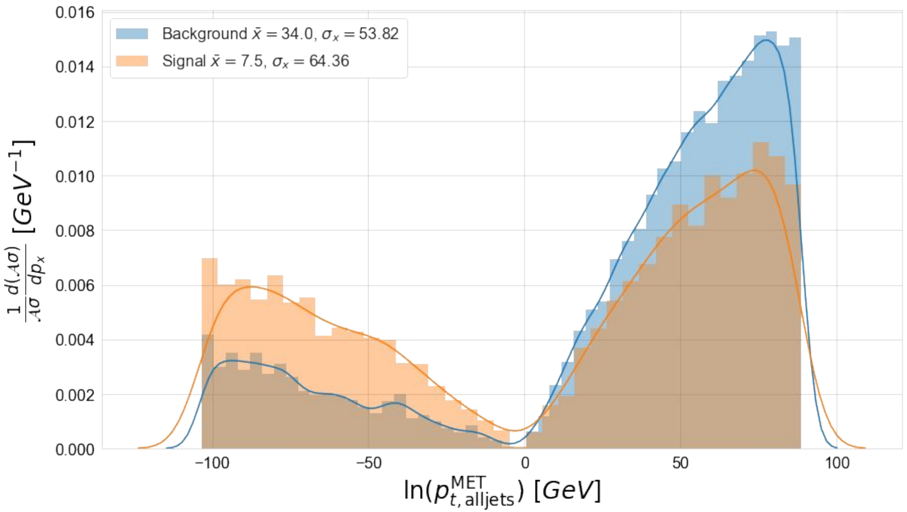
- Use physics knowledge to define subset of possible variables to check manually
- Define cut(s) to optimise sensitivity
- The variable(s) used are often complex combinations of basic information
    - E.g. masses, specific angles in a certain rest-frame, (s)transverse masses



Signal region

TRADITIONAL APPROACH

- These high-level variables are inspired by theory or experience
  - Limits the number of variables to check
- But, limits the performance of the analysis if a better feature exists but is not considered
  - E.g. the natural log of the transverse momentum of all jets raised to the power of the missing transverse energy can be computed, but who would?



TRADITIONAL APPROACH

- *Ad absurdum*: you can only know your analysis is as sensitive as possible if you have tested every single possible combinations of variables
- Clearly infeasible, but what if we can get close?

FUNCTIONAL APPROXIMATION

- Ideal scenario: a single variable with perfect signal/background separation *y*
  - Where *y* = some value for signal, and a different value for background
- This variable will be a combination of other variables <u>*x*</u> using some function *f* :

$$y = f(\bar{x})$$

  - Where <u>*x*</u> is a vector of other variables such as 3-momenta, masses, and jet multiplicity

FUNCTIONAL APPROXIMATION

- If we know the analytic form for *f*, then great! But we're here because we probably don't…
- Instead we can model *f* with a parameterised function to get an approximation of *y* :

$$\hat{y} = f_\theta(\bar{x})$$

  - Where θ are the parameters of our model
- This approximator is what a machine learning algorithm aims to fit

OVERVIEW OF MACHINE LEARNING

MACHINE LEARNING: TASK & DATA

- Signal-background separation is a *binary classification task*
  - Trying to determine which of two classes each event belongs to
- Normally performed using *supervised learning*
  - The model is provided with input data and the targets
  - Model predictions improved via an iterative learning process
  - Requires datasets with known labels, e.g. Monte Carlo simulation

MACHINE LEARNING: MAIN STAGES

- Training: model adjusts parameters θ to solve the problem using inputs $\underline{x}$ and targets y from training data
    - Model sees both $\underline{x}$ and y and uses them to update θ
- Validation: model with fixed θ is applied to data new data to check performance
    - Model only sees $\underline{x}$, but the score is used by user to guide training and compare models
- Application/testing: model with fixed parameters is applied to new data for which the target may not be known
    - Model only sees $\underline{x}$, the score may be computable but is not used to improve model

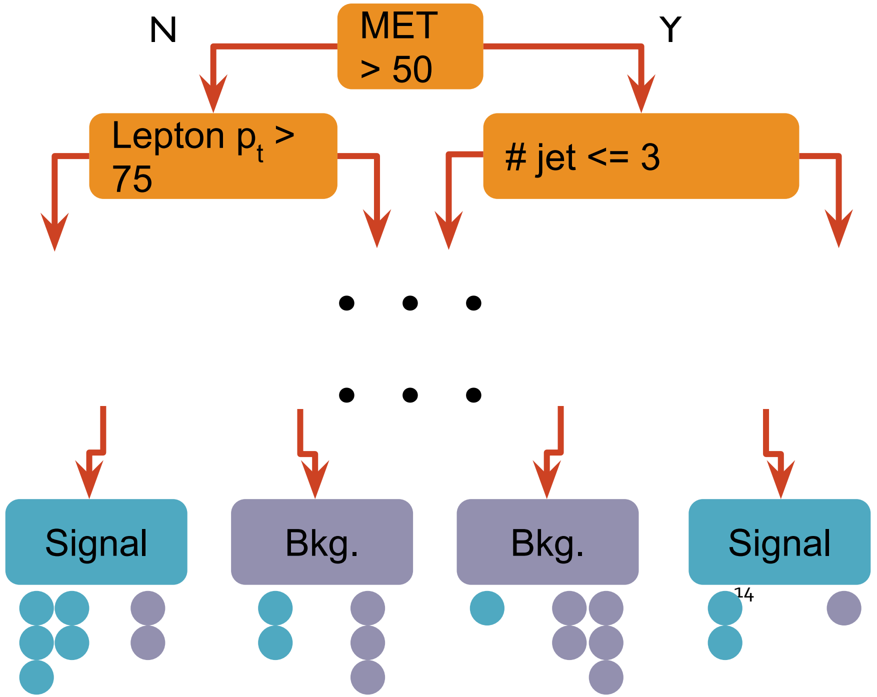
DICTIONARY

Machine Learning	Physics
Class / target	Signal or background
Feature	Variable
Ensemble	Averaging predictions of many models

Interactive Demo

ML ALGORITHMS: DECISION TREES

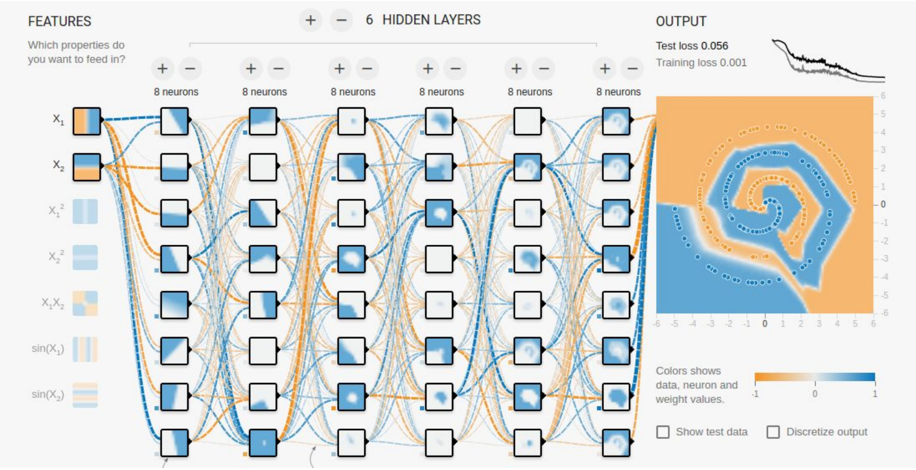
- Decision trees recursively split training data by cutting on the variables in $\underline{x}$
- End nodes of tree assigned class probabilities based on training data population
- Can be further improved by ensembling tens or hundreds of such trees:
  - Random Forests: learn a set of decorrelated trees by bootstrap resampling data and subsampling training features
  - Boosted Decision Trees: train each tree based on the residual prediction of the prior tree



Interactive Demo

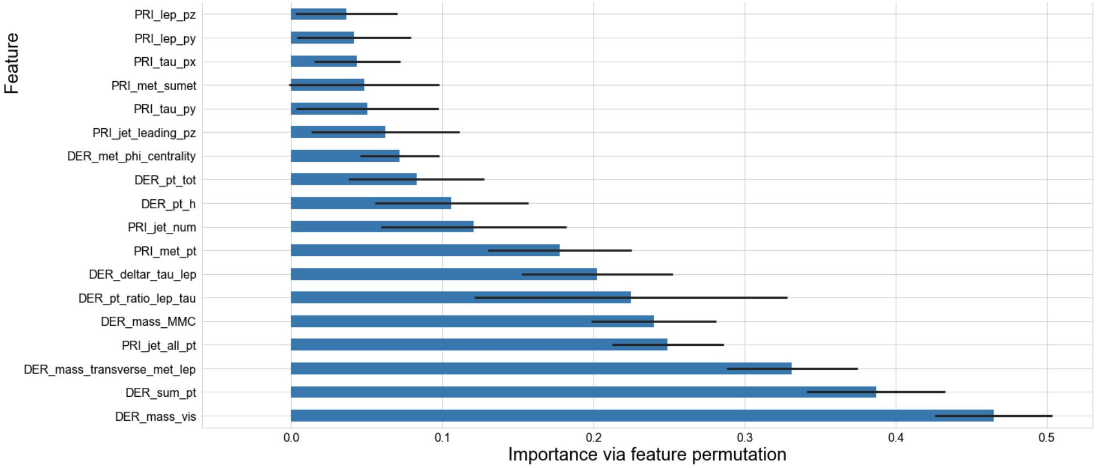
ML ALGORITHMS: NEURAL NETWORKS

- DTs consider many features at each split, but act only on one each time
  - Linear response
  - Can approximate nonlinear responses via ensembling
- See all features simultaneously
- Apply series of linear and nonlinear transformations based on learned parameters
- Direct access to nonlinear responses



INTERPRETATION

- Although complicated, ML algorithms can be understood via *interpretation* methods
- Useful to verify training and response
- Can help identify problems in model or data



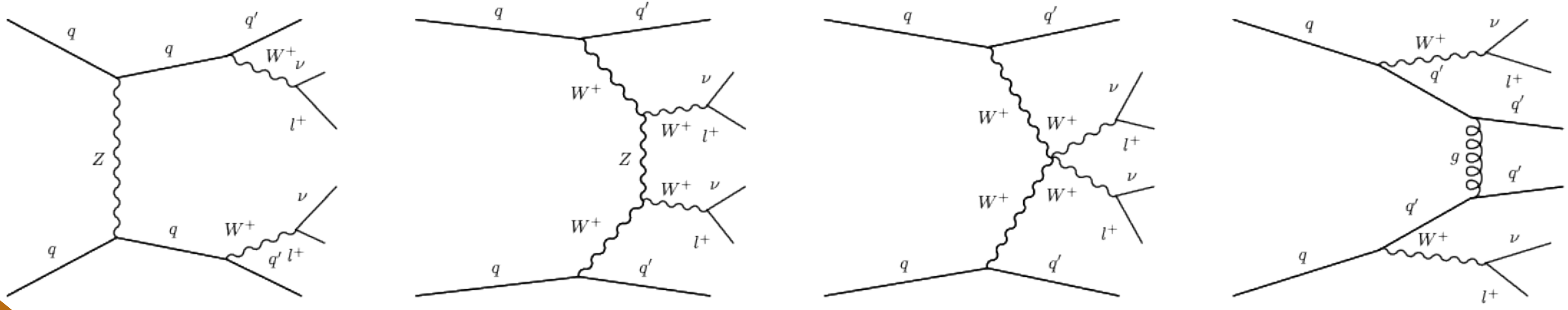
ML ALGORITHMS

- Summary:
  - Decision trees cut on features
  - Neural networks combine features together
- These are all things which a physicist could do
  - Machine learning is not doing anything strange or magical
  - Just automates the task of finding *f*, the function of the data which provides the most discriminating high-level feature
  - Well modelled input = well modeled output
  - Uncertainties can be propagated by evaluating on perturbed inputs

POSSIBLE APPLICATIONS TO VBS

EXAMPLE:
SAME-SIGN WW

- Published in arXiv:1709.05822
- Main diagrams:



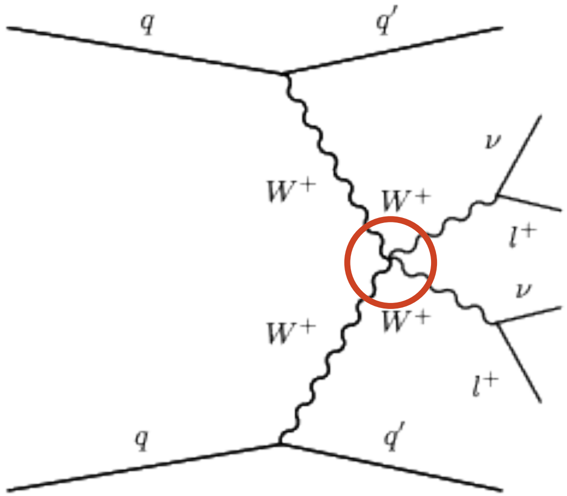
- 5 final-states: 2 jets, 2 leptons, and MET

EXAMPLE: SAME-SIGN WW

- Possible input features:
  - 4-momenta of final-states (p_x,p_y,p_z,E), for leptons and jets
  - Total MET and transverse components
  - Di-lepton & di-jet invariant masses
  - Angles between final-states
  - Number of jets in event
  - Transverse masses
  - Flavours of leptons & jets
- Target: Signal (EW WW) or background (WZ, non-prompt, others)

PARAMETRISED LEARNING

- Reference also considers modifications to the quartic couplings
- If these then modify the feature distributions or relative weighting of events the model may not work well on data with different couplings
- arXiv:1601.07913 presents method of *parametrised learning*
- Trains a single model on many different datasets (e.g. different couplings)
- Finds it works at least as well as many dedicated models trained for each coupling

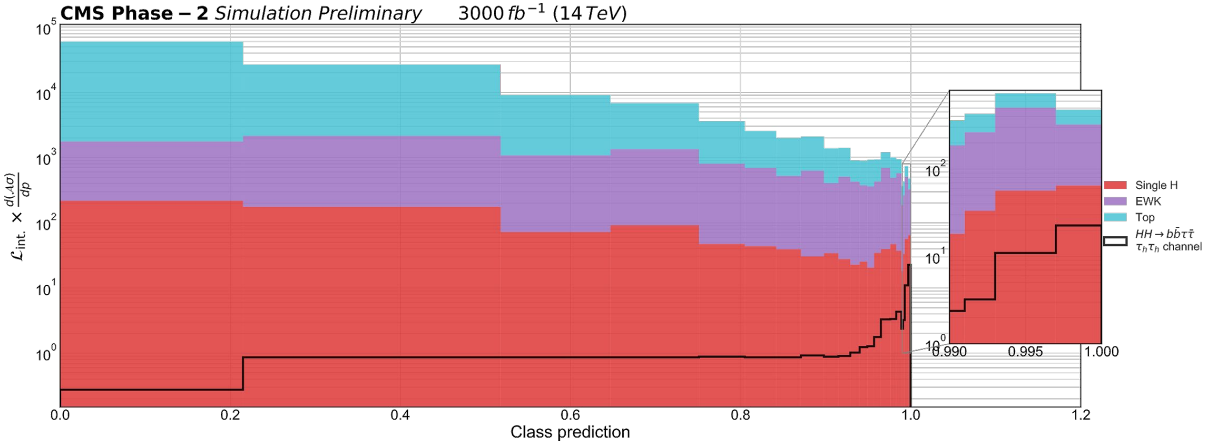


EXAMPLES OF ML IN HEP SEARCHES

Try it yourself

SEARCH: DI-HIGGS @ HL-LHC

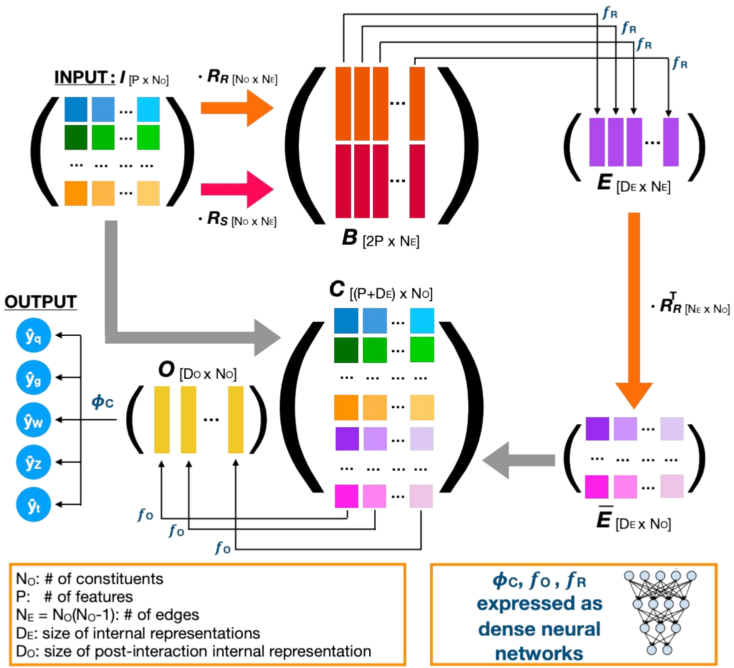
- CMS: CMS-PAS-FTR-18-019
- $hh \rightarrow bb\tau\tau$ search
- Neural network used as event-level classifier
- Able to discriminate well against large backgrounds
- Advanced training methods further improved NN performance by 20%



Github

OBJECT ID: JEDI-NET

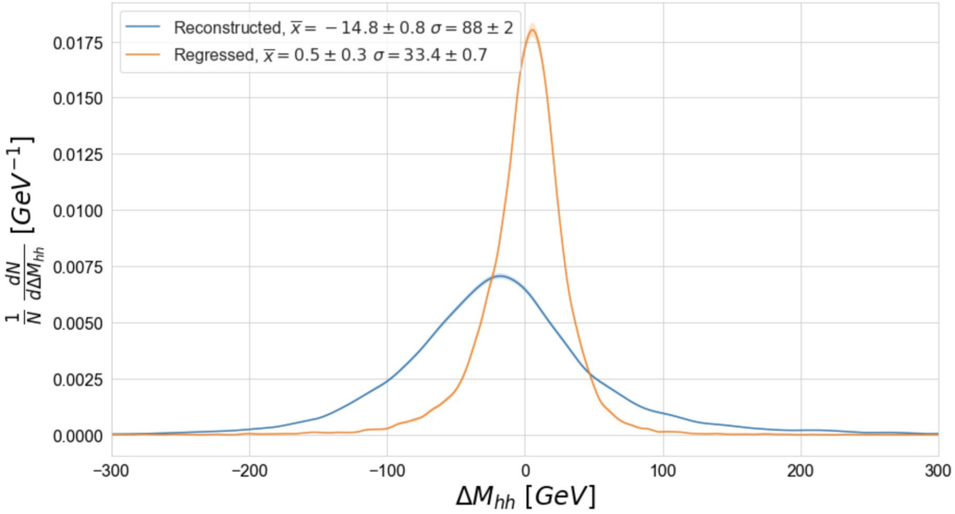
- Moreno *et al.* arXiv:1908.05318
- Sophisticated *interaction*-based neural network able to tag jets by flavour:
  - Light quark
  - Gluon
  - W-boson
  - Z-boson
  - Top quark
- Beats other approaches for inputting jet constituents (RNN, CNN, & DNN)
- Similar approach later used for double-b jets in Moreno *et al.* arXiv:1909:12285



REGRESSION: DI-HIGGS MASS

Try it yourself

- NNs can be used to regress quantities, e.g.:
  - Invariant masses
  - Particle momenta
  - Energy corrections
- Can also be used in place of transfer functions for applying the Matrix Element Method



SUMMARY

SUMMARY

- ML is a powerful & practical technique to automate the search for a specific (set of) variable(s)
  - Does not do anything a physicist couldn't do given the time and patience
  - ML algorithms are not *black boxes*; can be interpreted
- ML has many applications within HEP and is already being used to give significant improvements
- Requires some extra knowledge, but courses, software, and papers are freely available (see next section)

GETTING STARTED

LIBRARIES

- Most ML development done in Python 3
- Two main libraries: PyTorch & TensorFlow
- Both relatively low-level = need good understanding of NNs to use directly; but wrapper libraries exist to provide high-level APIs, e.g.
  - Keras - no longer developed standalone, but now included in TensorFlow 2.x
  - Fast.AI - PyTorch wrapper with best practices for image, text, & tabular data but doesn't support weighted data
  - LUMIN - My own library (in beta) - PyTorch wrapper with best practices for weighted tabular data, plus utilities for HEP, statistics, and interpretation

THEORY & PRACTICE: COURSES

- Fast.AI - free, practical courses; videos + library; top-down experiment first, theory later teaching style:
  - Machine learning - Fundamentals for data science + Python programming
  - Deep learning I - Best practices for image, text, & tabular data
  - Deep learning II - Building DNNs from scratch
- Stanford course - YouTube lecture series on theory of NNs
- Yandex MLHEP course - annual week-long intensive introduction to ML for HEP

THEORY & PRACTICE: EXPERIENCE

- Kaggle - data science challenge platform; wide range of challenges, get to see how others approach problems
- Paper reimplementation - helps get more familiar with library, and comfortable changing parts of it, e.g. SELU activation, categorical embedding, learning-rate annealing, and weight averaging