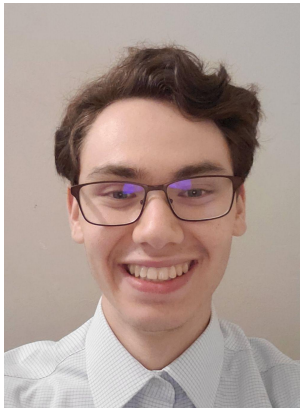
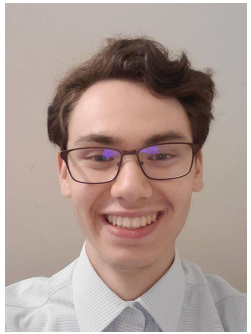
Using machine learning to measure the polarisations of same-sign W boson pairs from proton-proton collisions at the Large Hadron Collider



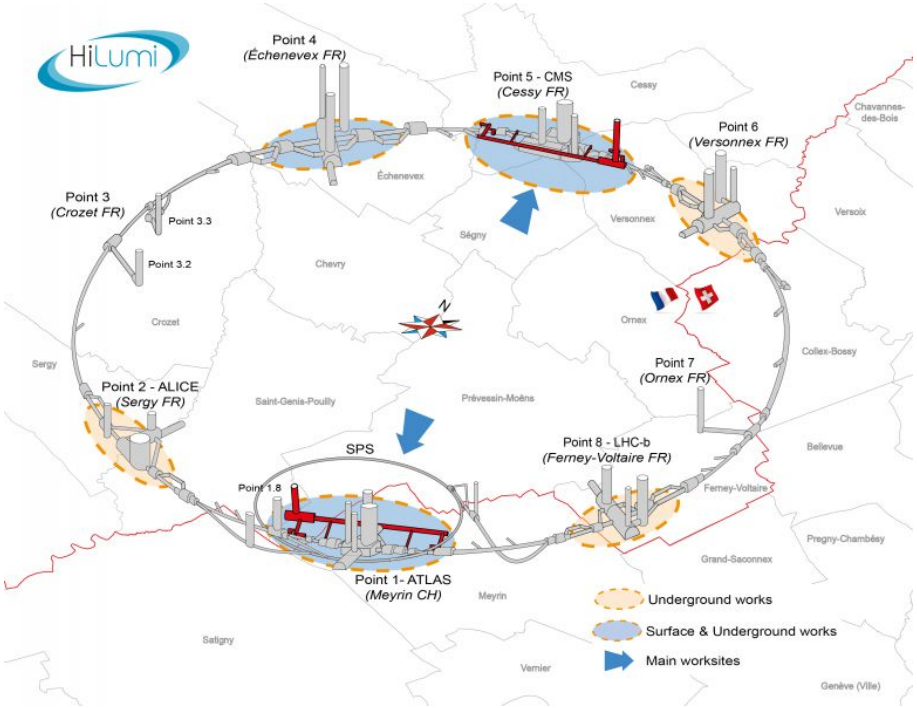
Sam Kelson

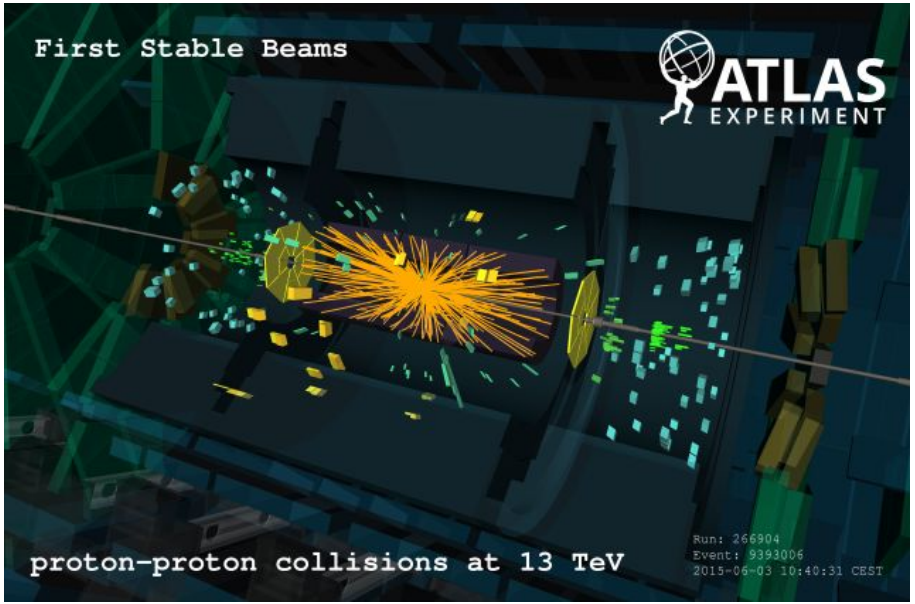
A little about me …

- Brandeis University class of 2025
- Physics major, math and cosi double minor
- Brandeis Rock Climbing Team
- Joel Kelson's son



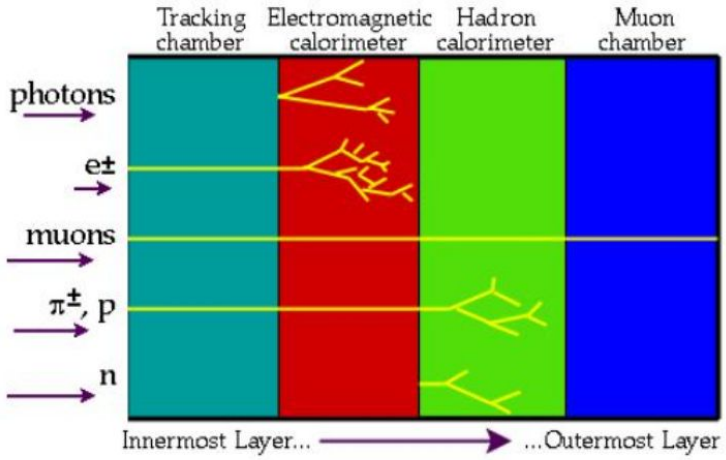
Proton-Proton Collisions at the LHC

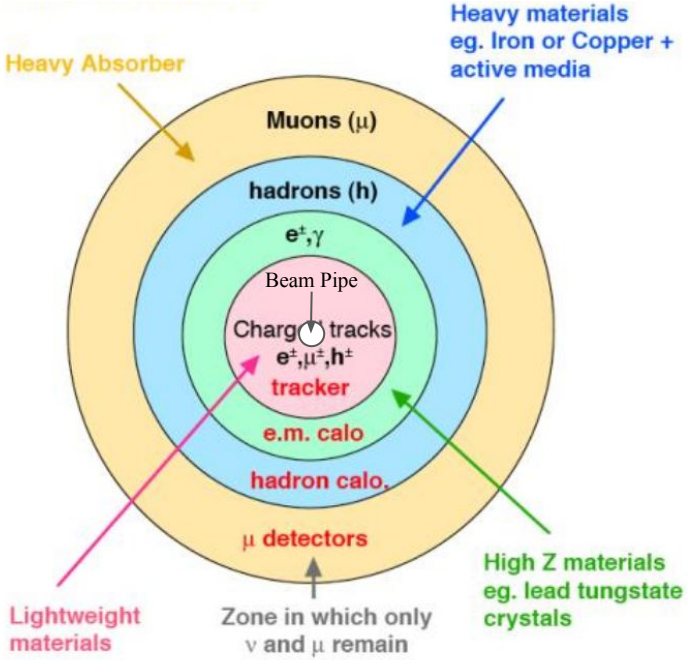




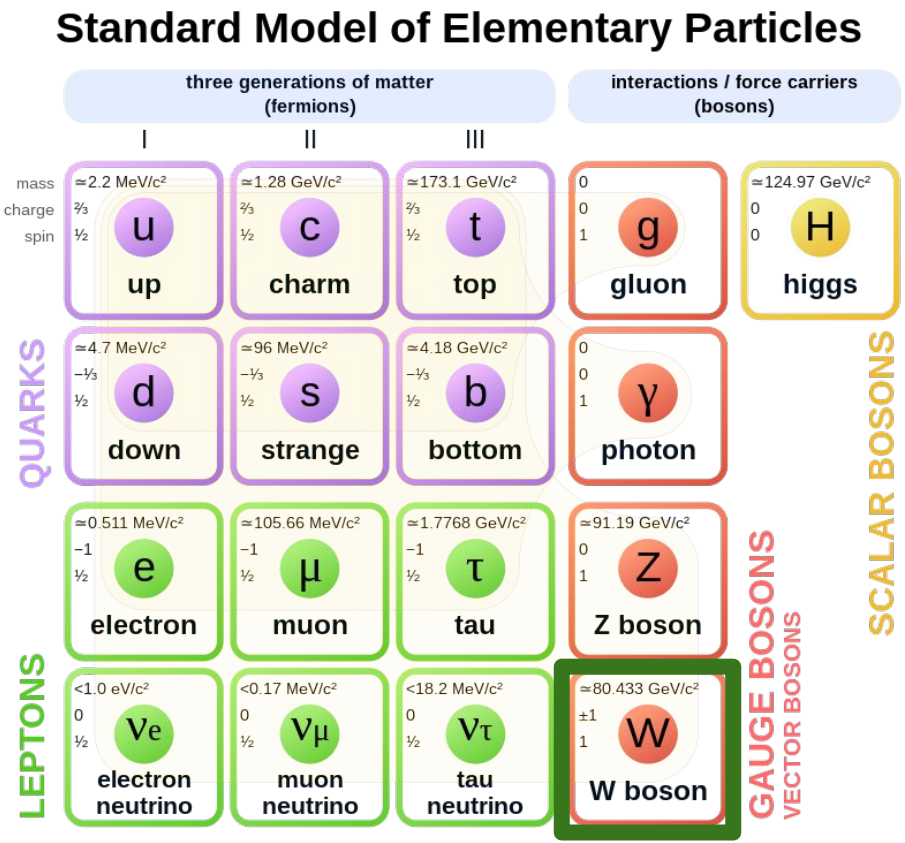
Proton-Proton Collisions at the LHC

- To distinguish between the different types of objects ($e^{\pm}$,γ, hadrons, etc.) most detectors use an onion like structure





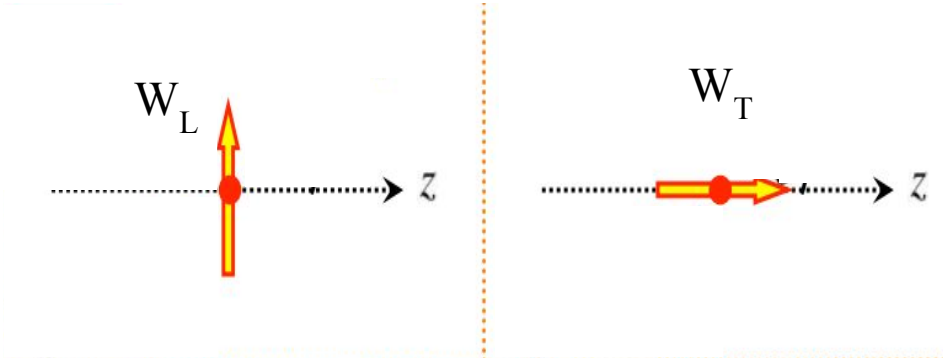
New Physics



- Searches for ‘new’ physics
  - W boson has interactions with the recently proven Higgs boson
  - One such interaction is the role of the Higgs in the **polarization** of vector bosons, such as the W boson, during/after **Vector Boson Scattering (VBS)** events

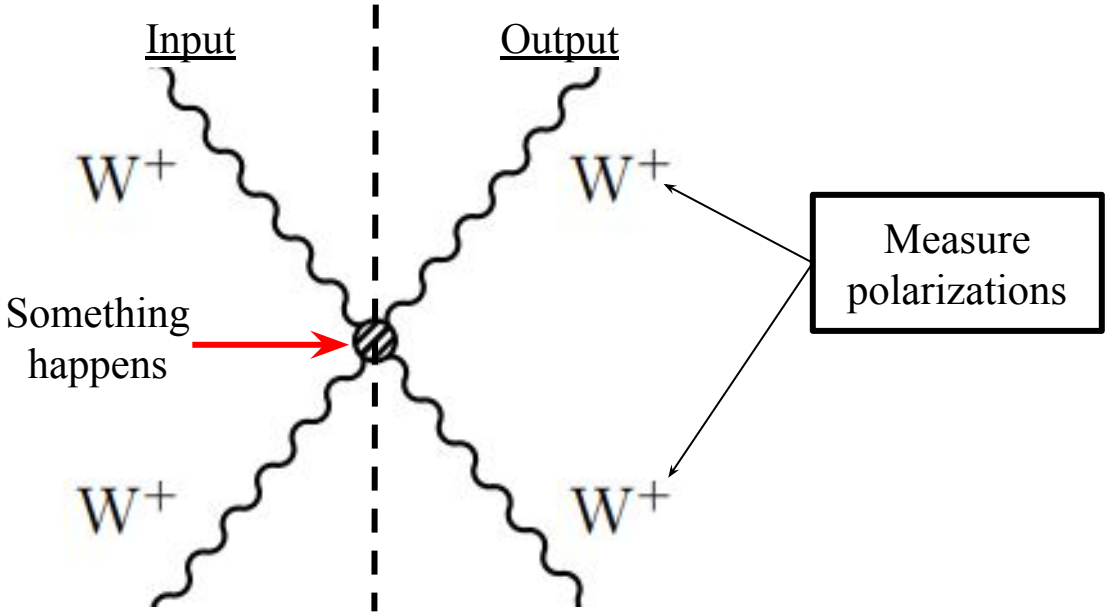
Polarization States of the W Boson

- Polarization - describes how the spin (property of subatomic particle) is related to its direction of motion



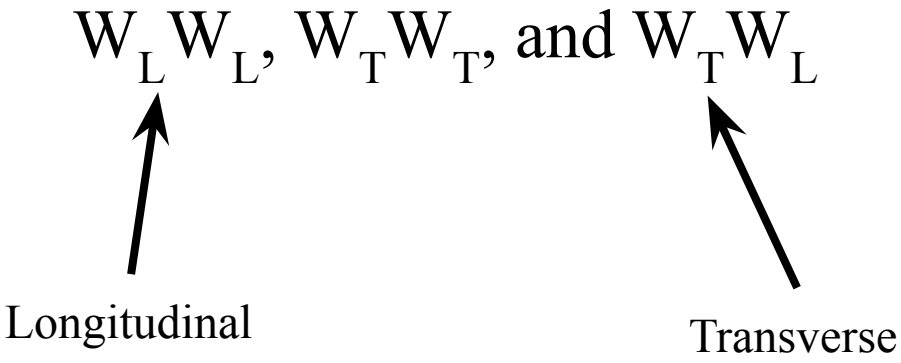
Vector Boson Scattering (VBS)

- Collision/Interaction of two vector bosons
- The focus of this study is the collision of two same charge W bosons

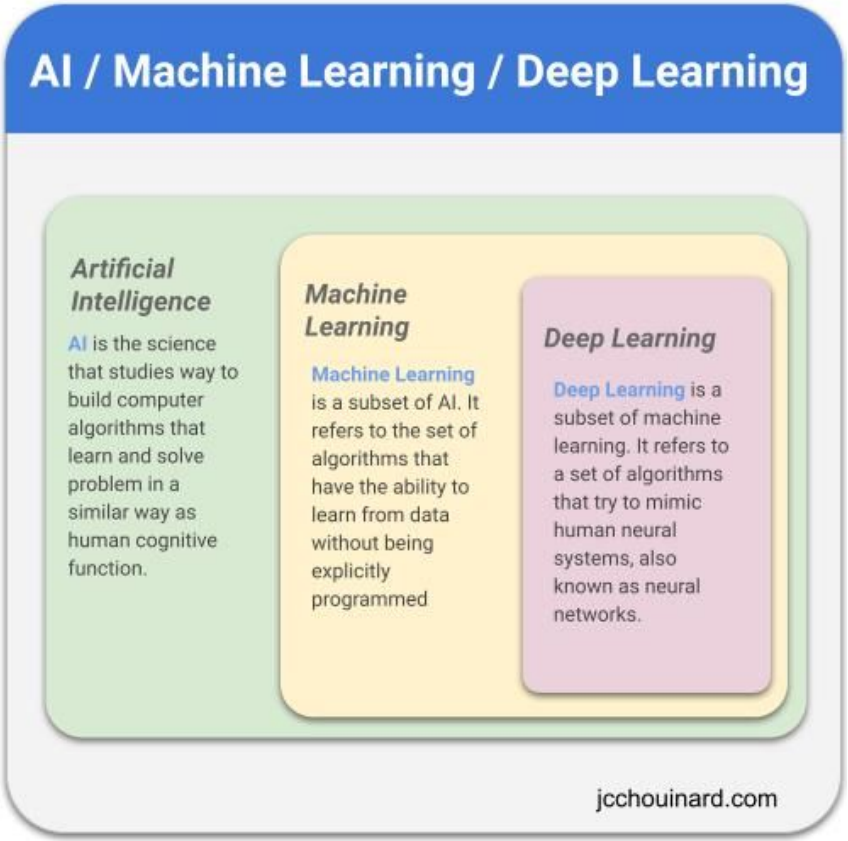


Objective

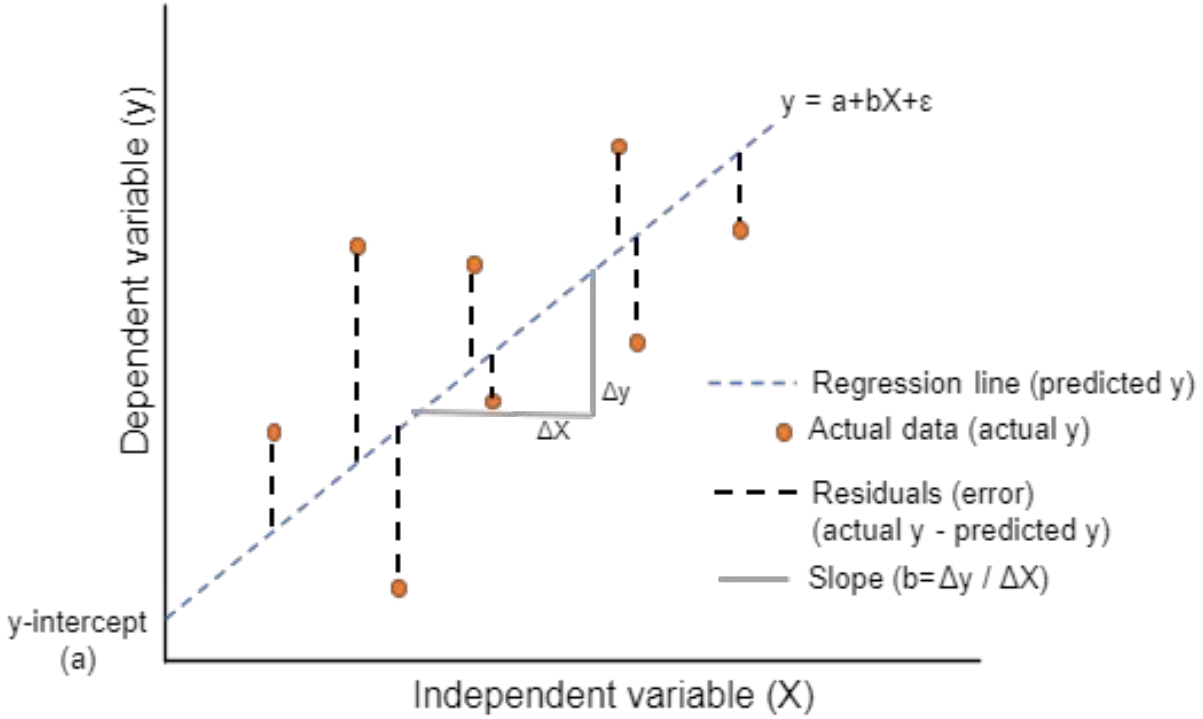
- From theoretical predictions, the amount of each category (LL, TT, TL) produced should follow a certain distribution
- **Goal**: Use machine learning techniques to differentiate between the polarization states of WW bosons from their leptonic decays in VBS events



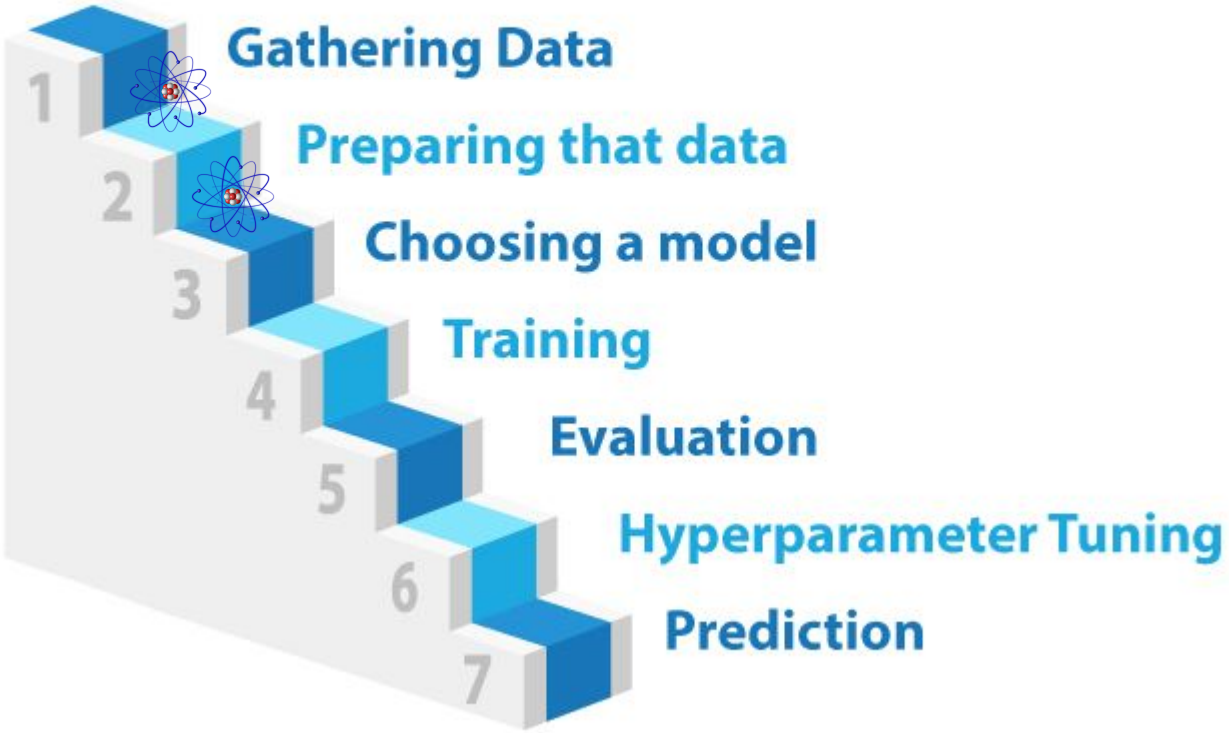
What is Machine Learning (ML)?



ML Ex: Regression



Application of Machine Learning



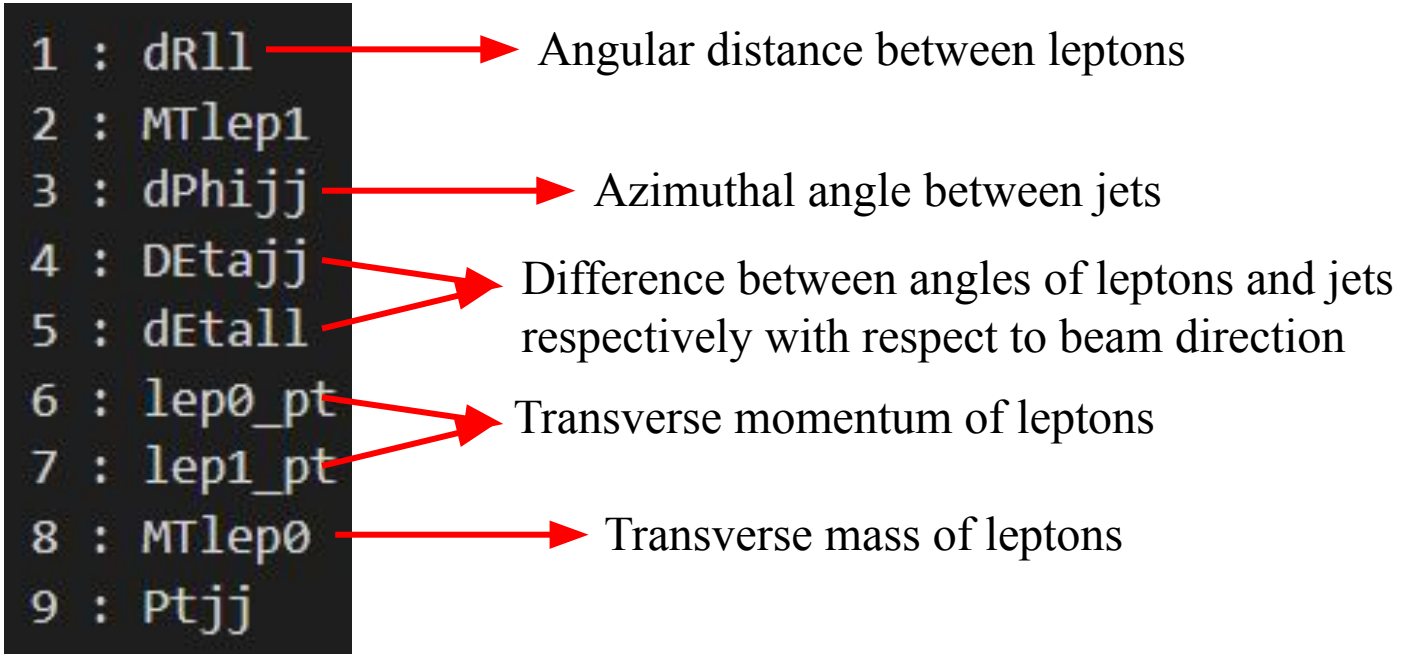
Gathering Data

- Take advantage of Monte Carlo(MC) simulations
  - Simulation models used to predict the probability of a variety of outcomes when the potential for random variables is present

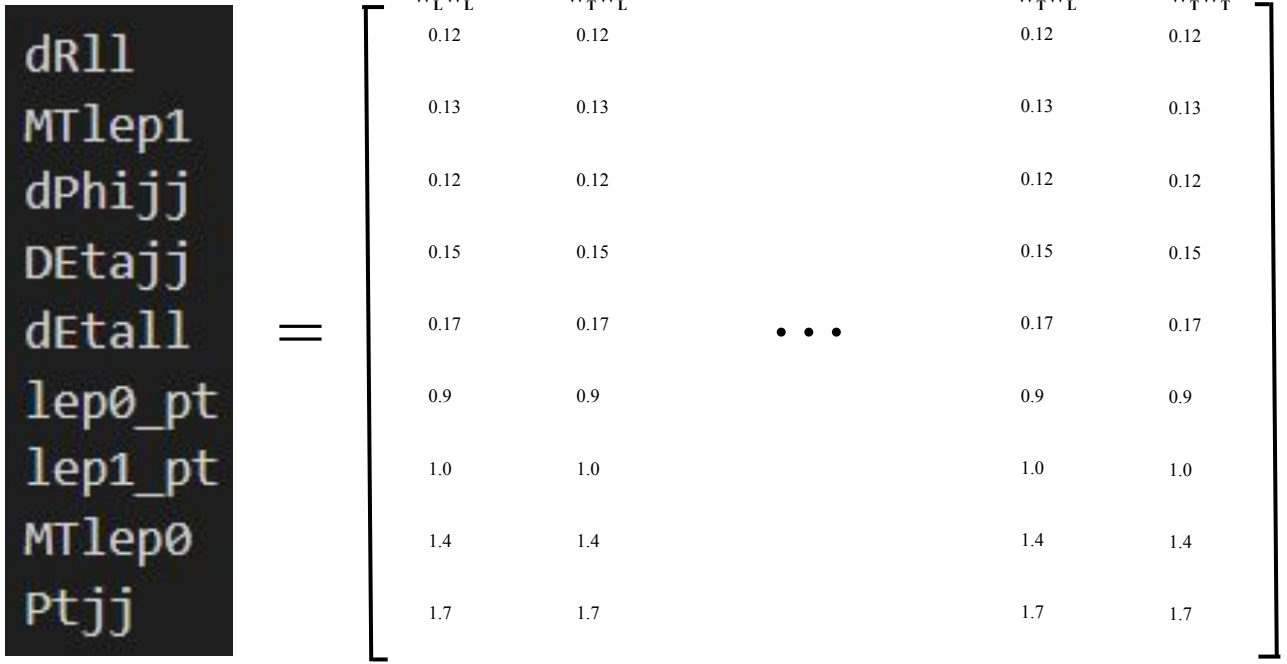
- Use MC's to simulate the VBS process and the detection of the decay products of W boson pairs

- Reason: the polarizations of each event is known

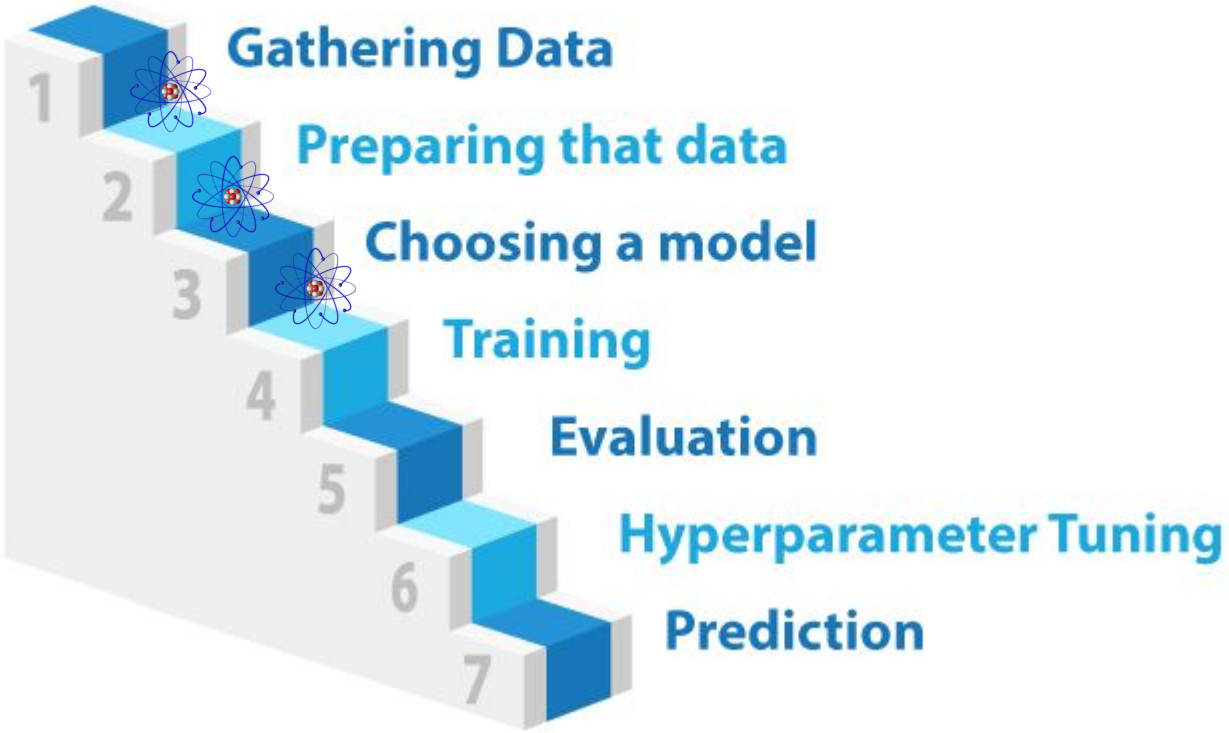
Variables/Features



Labeled Data



Application of Machine Learning



Types of ML: Supervised vs. Unsupervised Learning

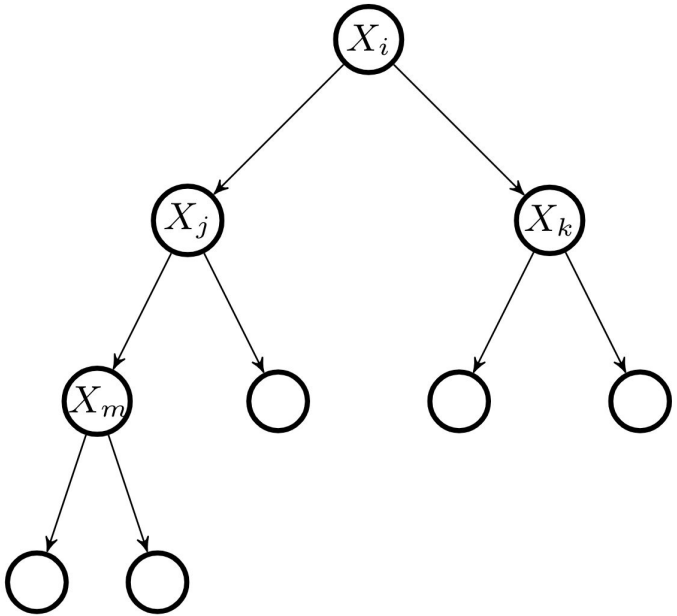
- Unsupervised
  - Given a dataset containing many features from different examples, learn useful properties about dataset (ex: clustering)
  - Training data is unlabeled

- Supervised
  - Given a dataset containing many features from different labeled examples, learn useful properties about dataset (ex: image classification, house price prediction)
  - Training data is labeled

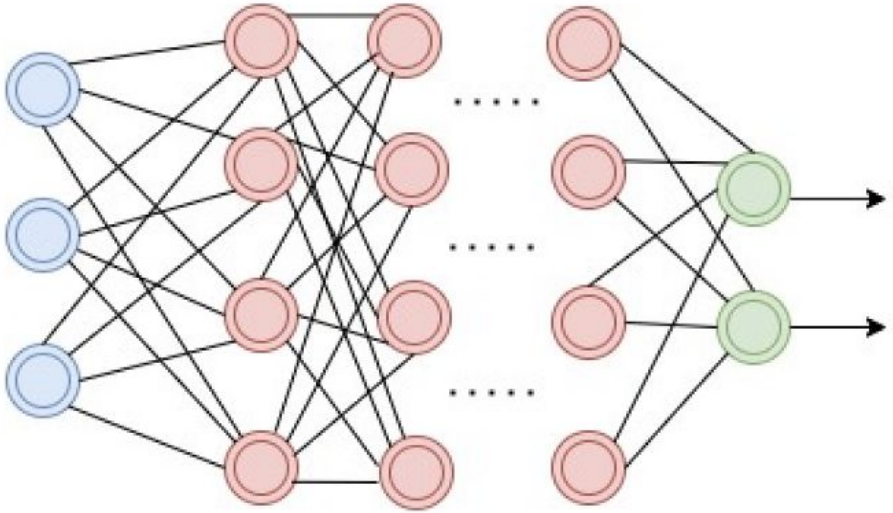
Machine Learning Models

- Two different models were used:

Boosted Decision Tree (BDT)



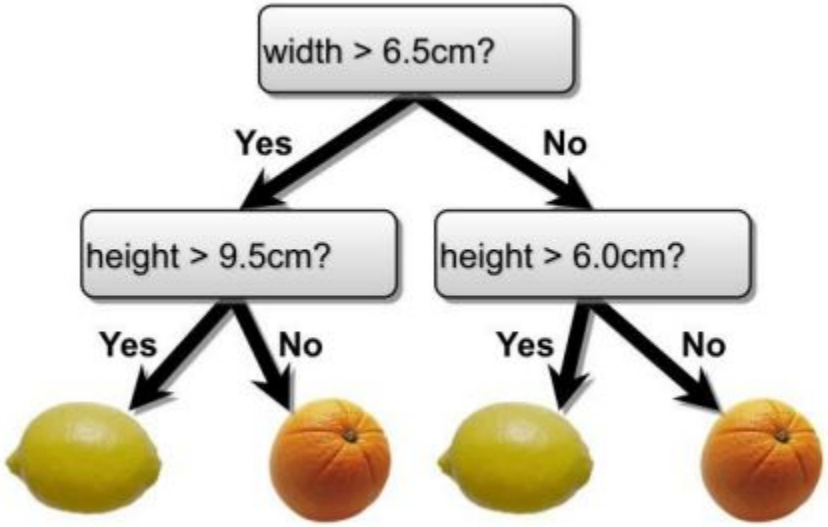
Deep Neural Network (DNN)



Boosted Decision Tree (BDT)

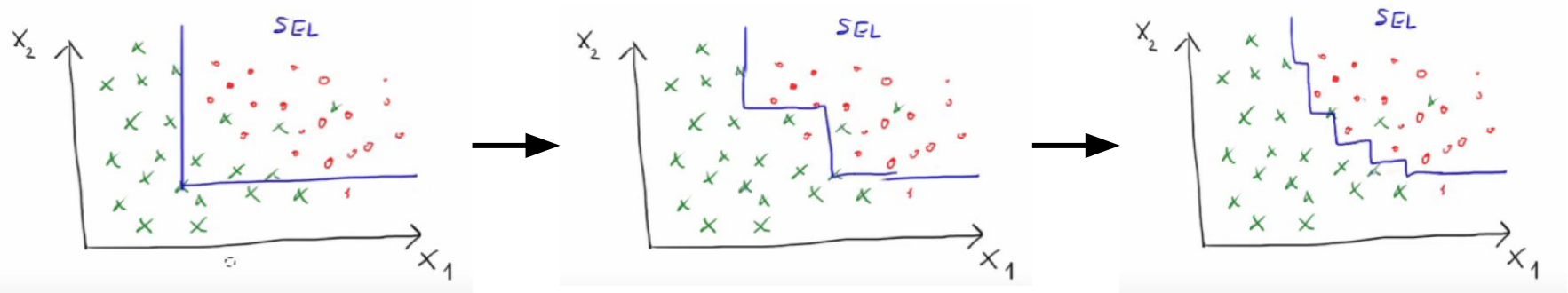
- Uses the parameters of a given event to give that event a certain score
- Use simulated (Monte Carlo) data to train the BDT

Decision Tree



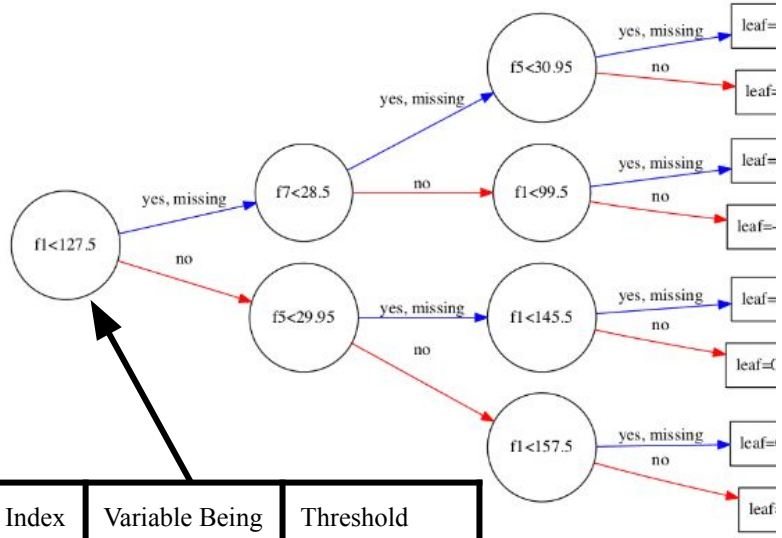
BDT Training

- BDT's make binary decisions (boolean logic)
- A different way to think about BDT's is making cuts
  - As we go down the tree:



BDT Training Algorithm

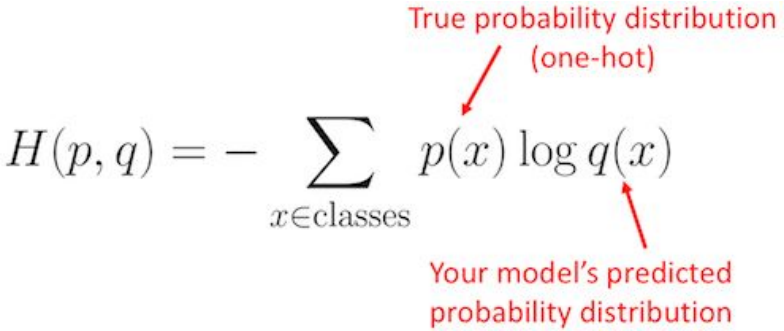
- Determine best cut(s):
  1. Loop over all variables
  2. Calculate splitting index
     - Rank of possible cuts for each variable (usually limited to 20 random cuts per variable) based on how good they are at splitting data categorically
  3. Best cut is highest ranked on the splitting index
  4. Wipe splitting index and start again



Splitting Index Rank	Variable Being Split	Threshold
1	Length	127.5 mm
2	Width	543 mm
3	Height	123 mm
…	…	…

Splitting Index

- Based on a cost/loss function (ex: Cross Entropy Loss)



- How good prediction set matches ground truth set
- The more the cut minimizes the cost function the higher it is ranked on splitting index
- Important idea I will come back to later

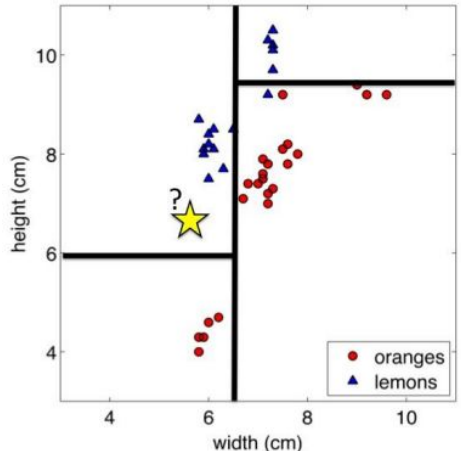
BDT Issues

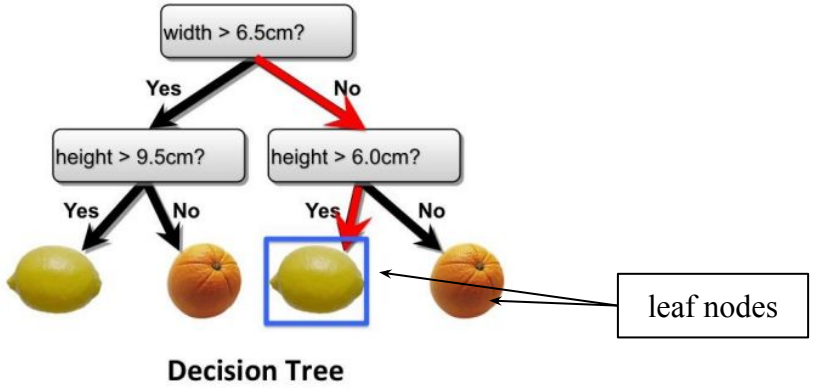
- **Overtraining** or knowing when to stop:
  - Require a certain signal purity after a cut is performed
  - Require a minimum fraction of events be in each node
  - Require a max number of branches

- Even with these steps BDT's are still weak classifiers

BDTs So Far…

- To use a trained tree:





- Leaf nodes can be designated as the signal or background depending on search target and the purity of the leaf node
- Any event that 'falls into' a leaf node can be called either a signal event (BDT score = 1) or a background event (BDT score = -1)

How to Make BDT Better?

- Add gradient boosting:
  - Add weight to events that are put into wrong category and then re-train tree on data with updated weights
  - Do this n number of times to create a certain number of trees ← hyperparameter
  - Average all trees together

- This greatly improves the BDT classifier

BDT Implementation

- Uses the BDT to assign a score from -1 to 1 on how likely each event is in the background or signal (search target) category respectively
- 2 separate BDTs:

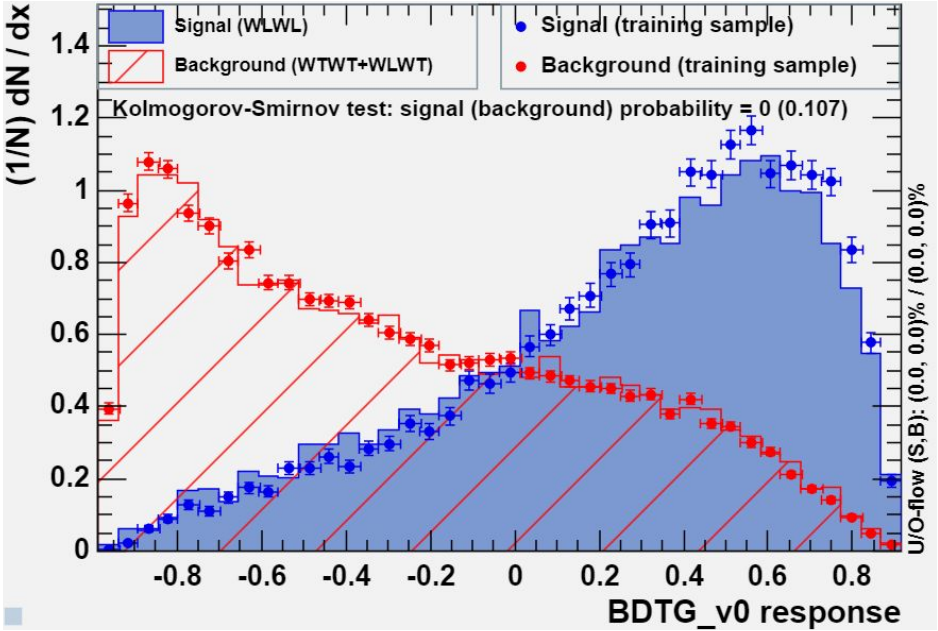
Signal (score: 1) [$W_L W_L$ vs. ($W_T W_T$ + $W_L W_T$)] Background (score: -1)

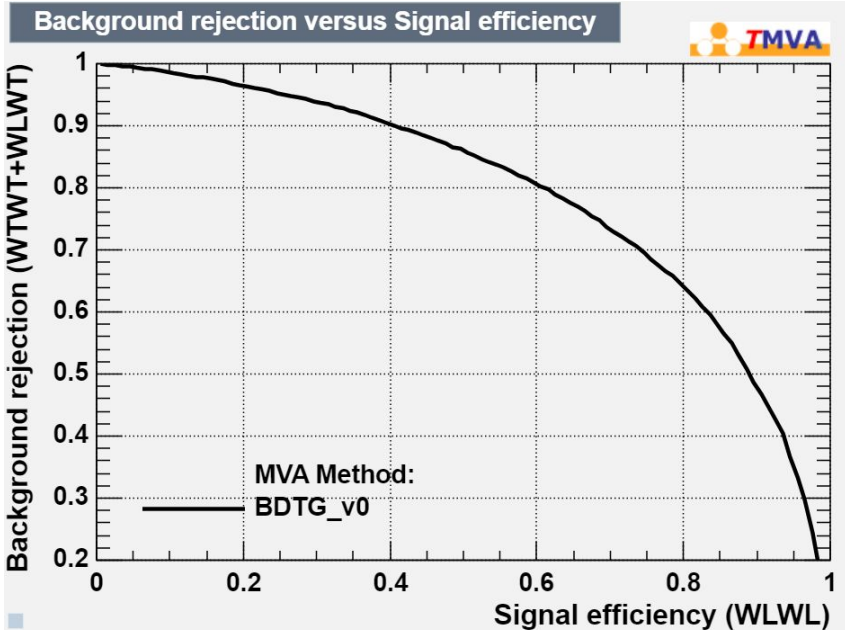
and

[$W_T W_T$ vs. ($W_L W_L$ + $W_L W_T$)]

- ROOT's Toolkit for Multivariate Data Analysis (TMVA) in PyROOT w/ Jupyter Notebooks
- Hyperparameters:
  - nTrees = 1000
- 70% of data was used for training and 30% was used for testing

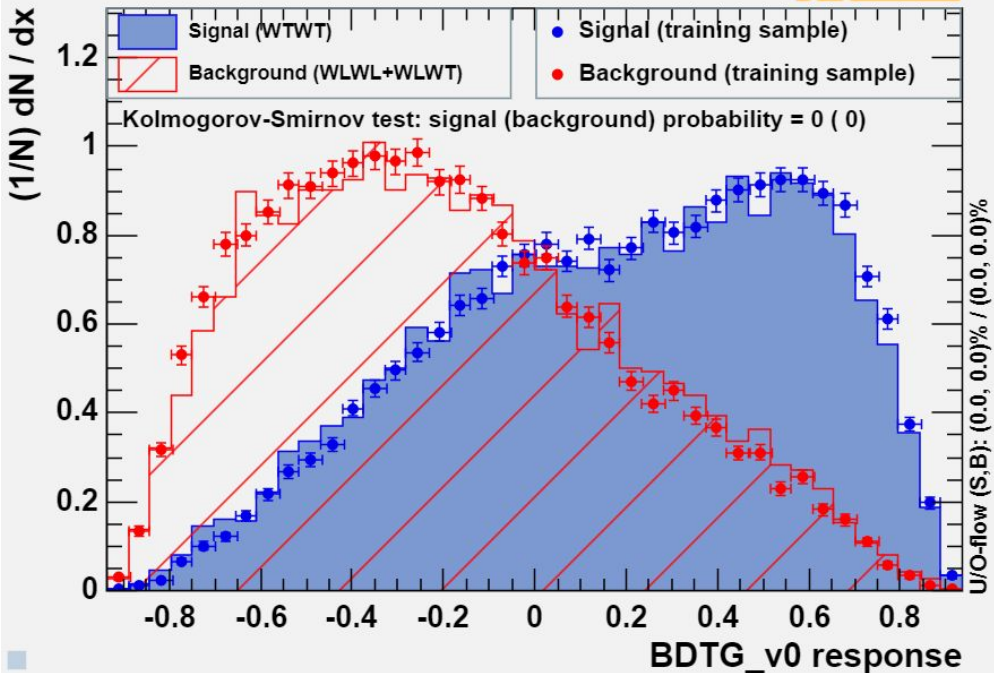
BDT Results: $W_L W_L$ vs. $(W_T W_T + W_L W_T)$

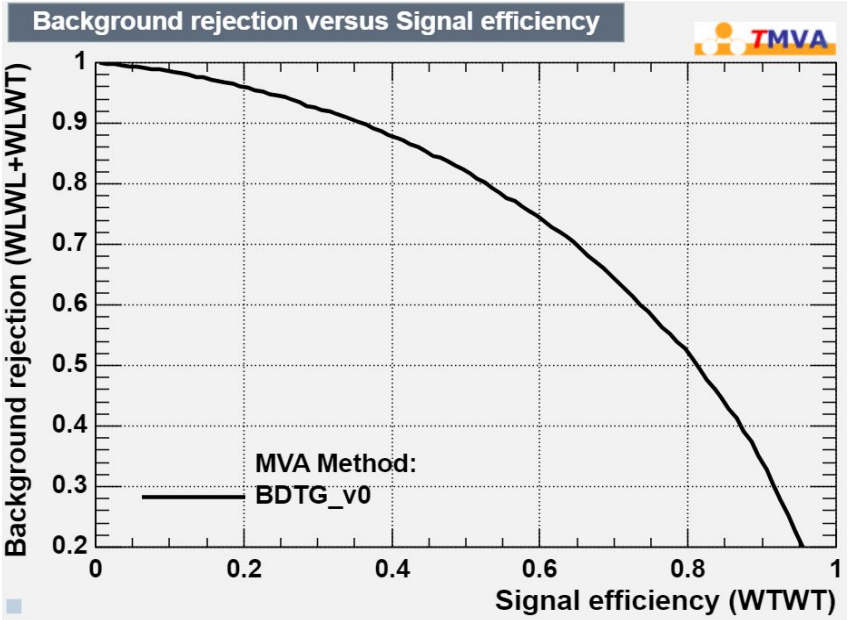




AUC: 0.791

BDT Results: $W_T W_T$ vs. $(W_L W_L + W_L W_T)$



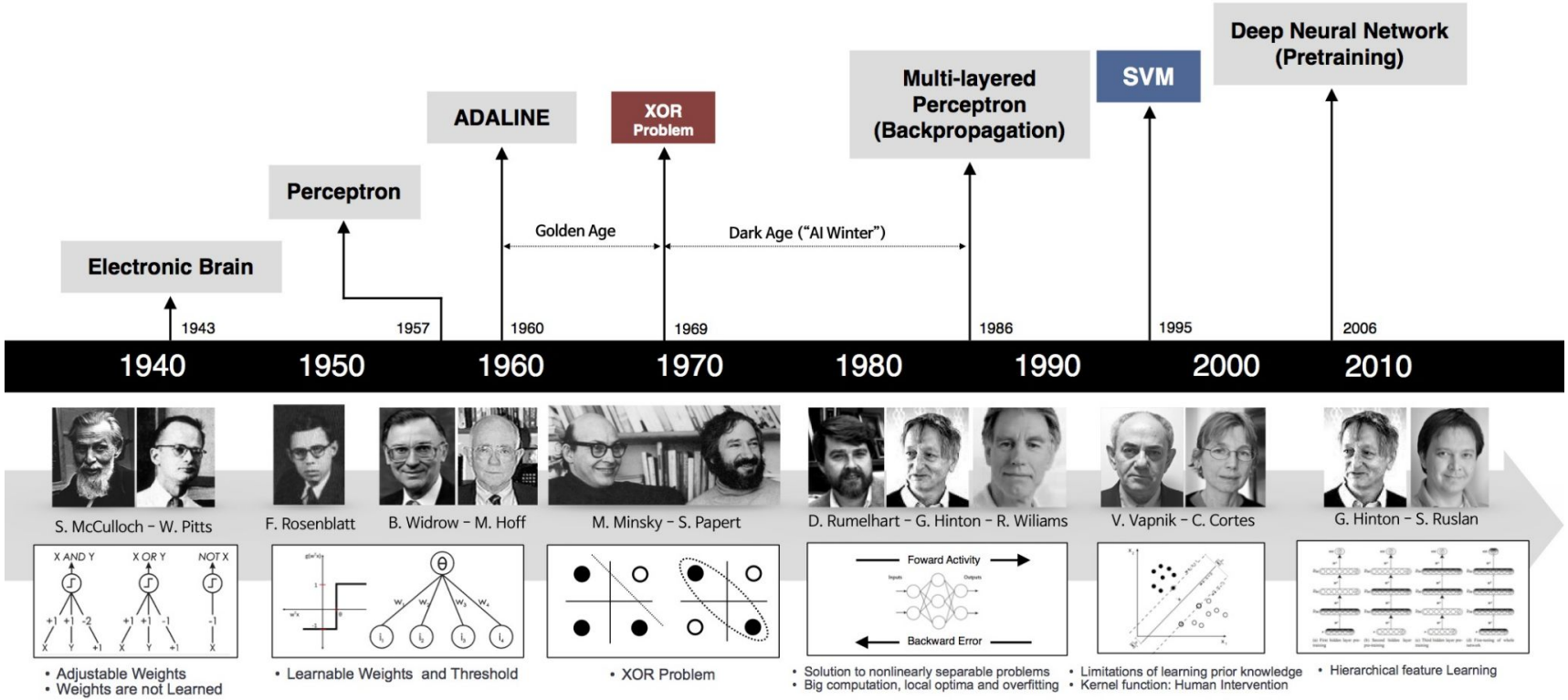


AUC: 0.736

Now on to Deep Learning

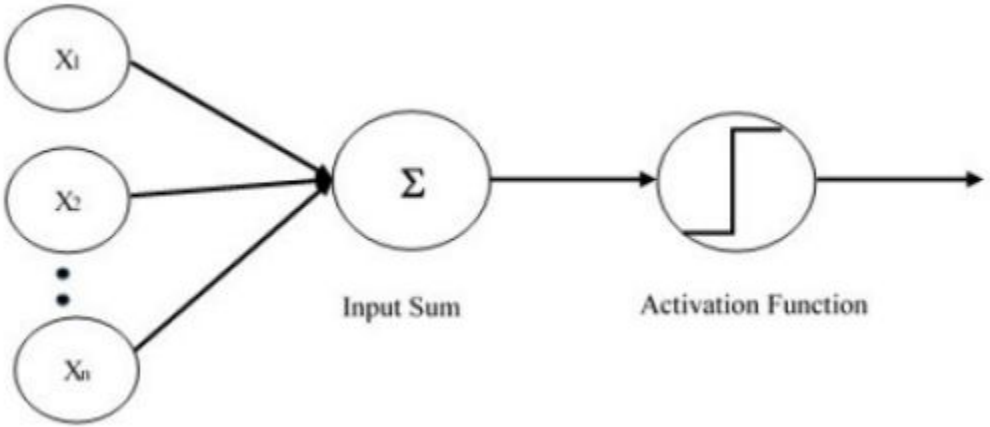
- Deep Learning vs Classic ML
  - Classic ML: Feature Engineering
    - Use features to represent data (ex: house price prediction)
    - Typical Problems:
      - missing features, improper features, domain knowledge, time consuming
  - Deep Learning: Representation Learning
    - learn feature representation automatically
      - high level/abstract features that are not directly rep. by data (specific objects, heads, body parts, signs, etc.)
    - little human/domain knowledge needed
    - strong performance

Quick History (depending on time)



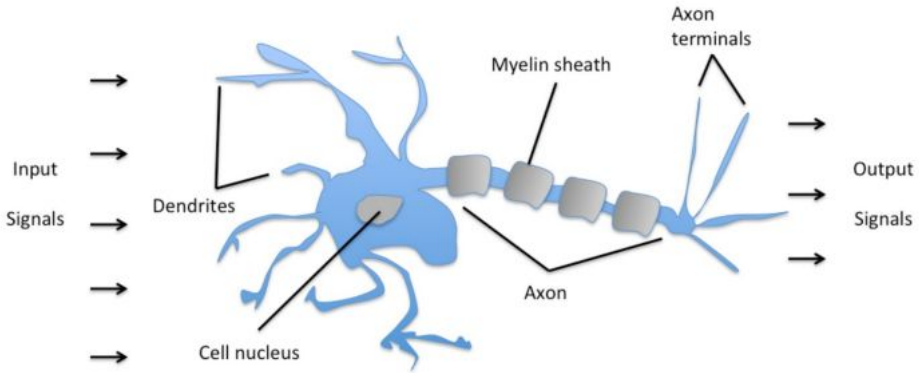
The Perceptron

- First-generation Neural Networks

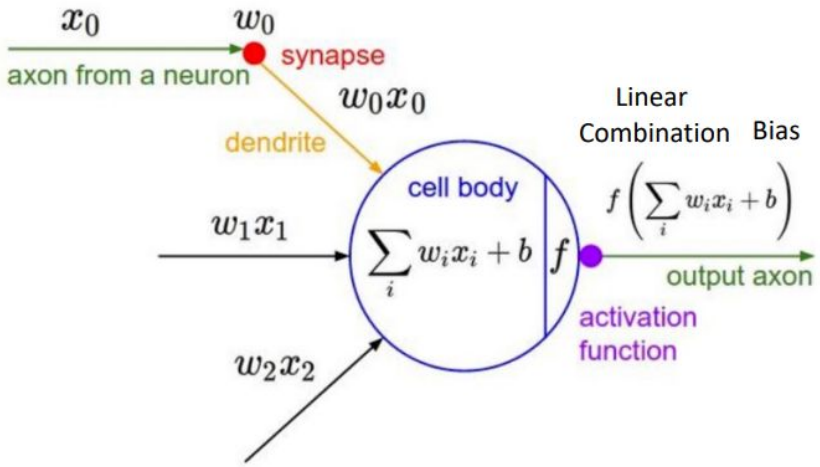


The (Advanced) Perceptron

Biological Inspiration

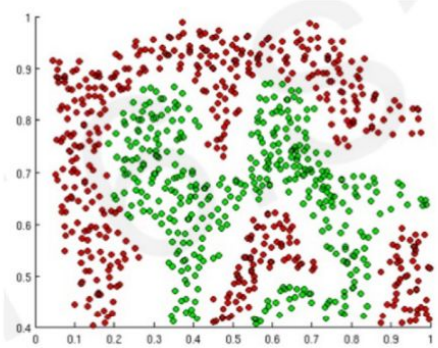


Artificial Neuron



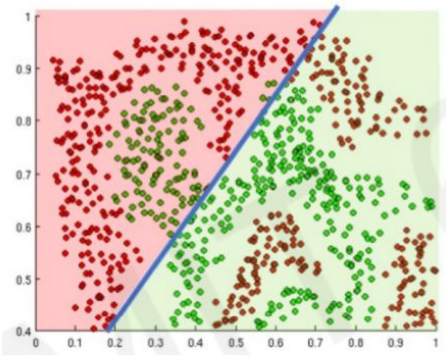
Activation Functions

- Non-linear activations allow us to approximate arbitrarily complex functions

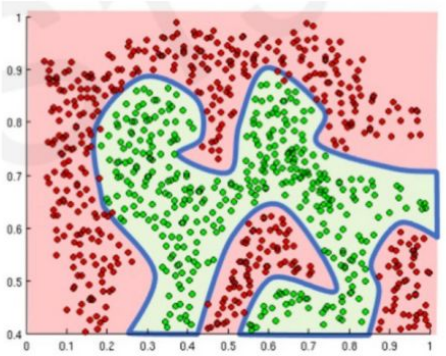


Classify green and red points

(Pictures from MIT 6S191)

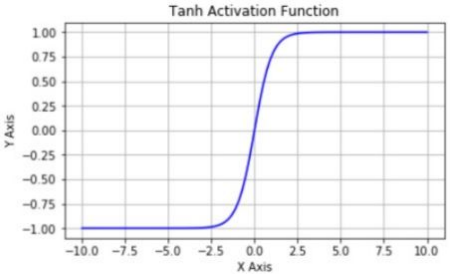


Linear Action



Non-linear Action

ex:

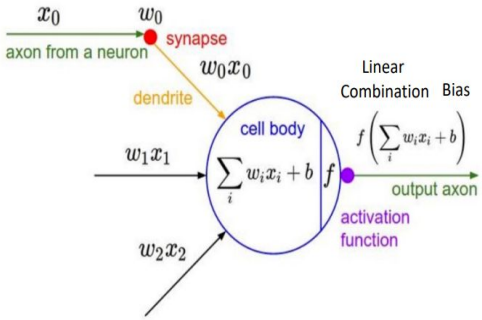


$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

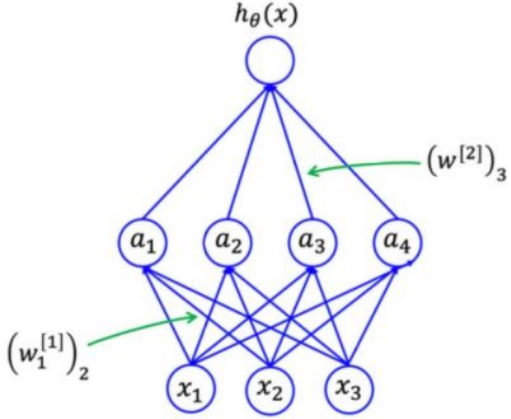
Neural and Deep Neural Networks

- A composition of perceptrons into layers:
  - ‘Deep’ when more than one hidden layer

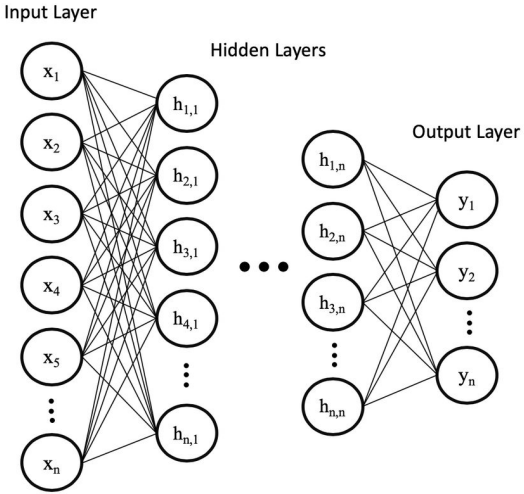
Artificial Neuron



Single Hidden Layer Neural Net.



Deep Neural Net



Cool Aside: Universal Approximation Theorem

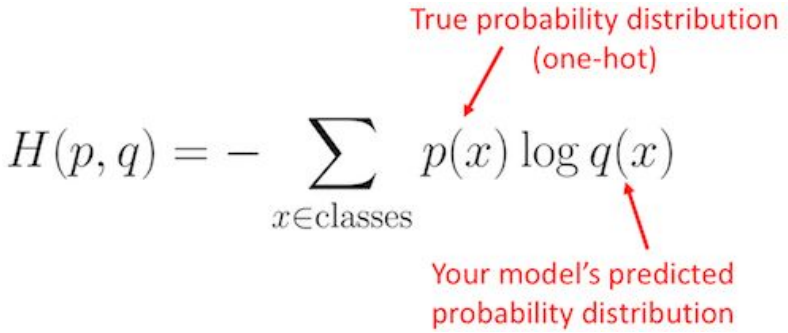
A feedforward network with a linear output layer and at least one hidden layer with non-linear activation function (e.g., sigmoid) can approximate any measurable function, provided that the network is given enough hidden units.

- Fact: A large MLP will be able to represent any function.

- Fact: We are not guaranteed that the training algorithm will be able to learn that function.
  1. The optimization algorithm used for training may not be able to find the optimal parameters that corresponds to the desired function.
  2. The training algorithm might choose the wrong function due to overfitting.

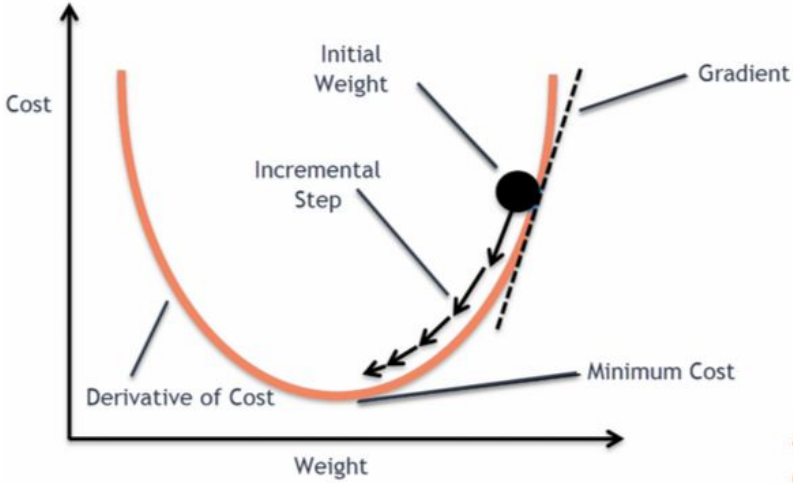
Training a DNN

- How do we learn the weights and biases?
  - By telling rewarding model when it is doing good and punishing it when it is doing bad
  - ➢ A cost/loss function to ‘grade’ the model
- The most popular loss function for classification is the Cross-Entropy Loss mentioned earlier:



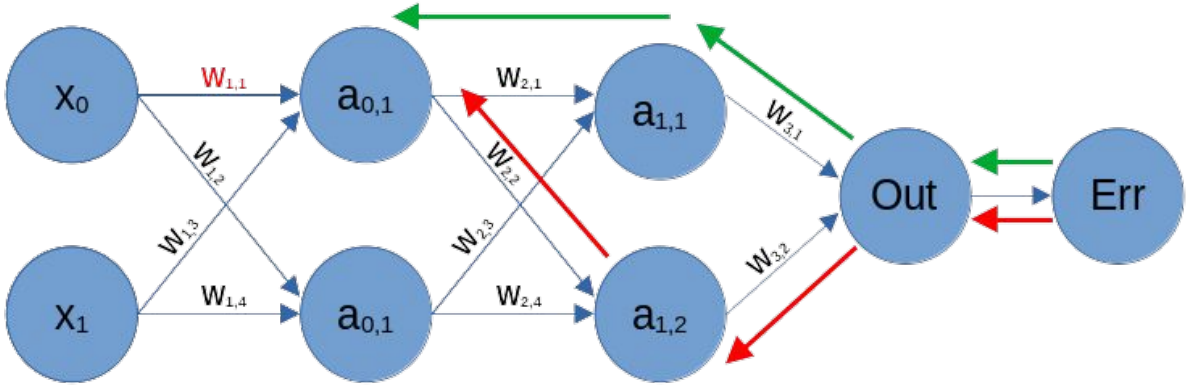
Training a DNN: Gradient Descent

- Most minimization/optimization algorithms are based on the process of gradient descent
- Minimize the cost function w.r.t each parameter (weights and biases)
- Basically taking a bunch of partial derivatives w.r.t the cost function to slowly update all parameters



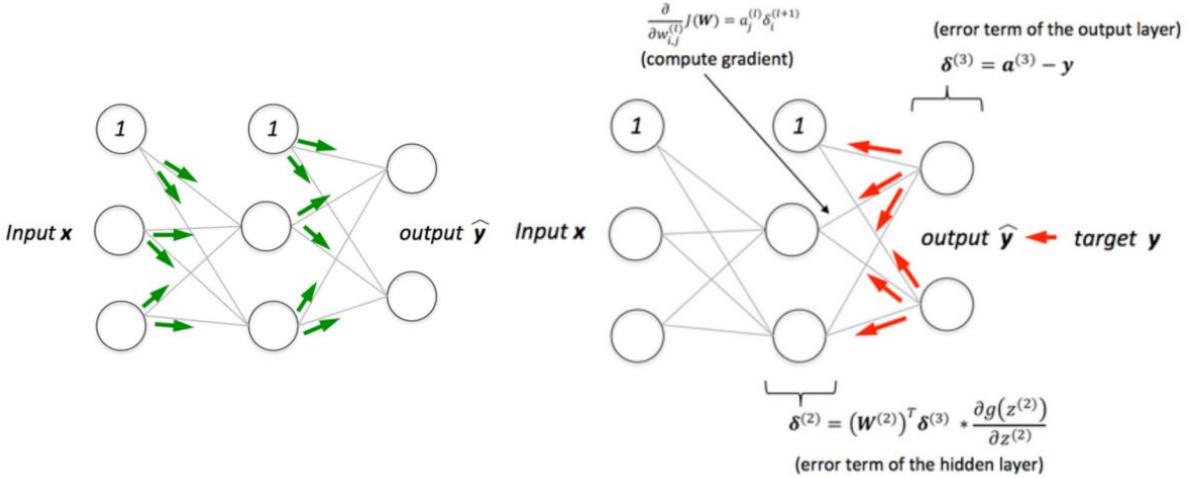
Training a DNN: Backpropagation

- However, the partial derivatives depend on the outcome of the computations that occur in the forward pass
- Therefore parameters must be updated backwards:



DNN Training Loop

- Loop until tired:
  1. Pass some data through the NN to get predictions
  2. Calculated loss and backpropagate errors
  3. Update weights



DNN Implementation and Results

Sorry, maybe next time…

DNN vs. BDT Theoretically

- Tree methods, while simpler, often work better for tabular data
  - Features are simpler, no need for abstract features
- With tabular data often statistics are better than compared to an image
  - Perfect cuts can be made using statistics instead of trying to descend a gradient
- Trees are deterministic while NN are probabilistic

Summary and Next Steps

- Two BDTs was created to separate the polarizations of ssWWjj VBS events in proton-proton collisions
  - Classification separation looks promising
- I learned …
  - Scratched the surface of particle physics
  - LHC Data
    - ROOT
    - nTuples
  - Machine learning (BDT & DNN)
  - Interpret data and results

Acknowledgements

- Special Thanks to:
  - Prof. Aram Apyan
  - Daniel Camarero Muñoz
  - Todd Zenger
  - Brandeis High Energy Group
  - ATLAS SUPER Program
  - My family

Thanks!

Thanks for listening!