Introduction to Regressor NN

Offshell Workshop Tutorial

John Rotter (Rice)
Graduate Student

RICE

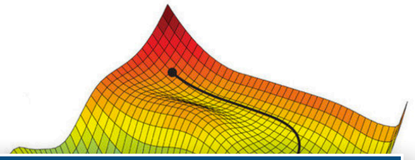
Previous Tutorial

What is Machine Learning?

- Data driven general solution to a complex problem - Universal Function Approximators!
- The name “Learning” comes from personification of the algorithm but it is really just an optimization problem and **not a sentient being**
- Supervised Learning is when the algorithm is given a deterministic task

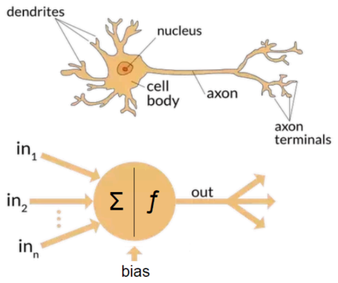
How to Reward the Machine

- To motivate an algorithm to perform as intended, its performance will need to be quantified
- Often this is done using a Loss Function which gives a penalty to wrong answer



Origins of Neural Networks

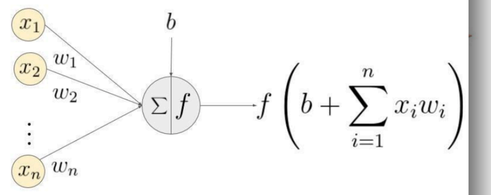
- Neural Networks were developed in an attempt to mimic how neurons in the brain function
- Neurons in the brain are wired to each other with each connection having its own weight
- Neurons also have their own internal behaviors and can send signals of their own to other neurons in the brain
- By building a simplified model of neurons and allowing them to wire together, we get a Neural Network



“Neurons the fire together, wire together”

The Neural Network Model

- Neural Networks (NN) have an input layer
- Each sequential layer is fully connected (dense) to the previous layer
- Each neuron’s value is computed as:
  - Sum of all previous neuron’s values, x_i, weighted by w_i
  - Plus an internal bias of the neuron, b
  - Then passed through an activation function, f
- This computation is done for every neuron in ever layer!
  - Since it is essentially matrix multiplication, this can be done very fast by the GPU!
- The final layer is the prediction of the network



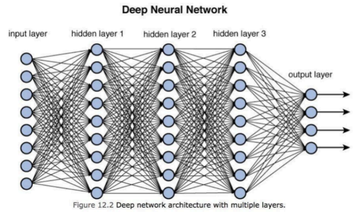
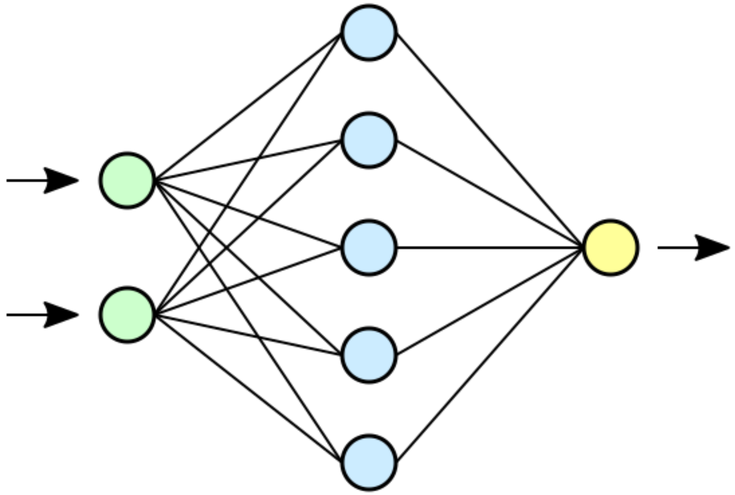


Figure 12.2 Deep network architecture with multiple layers.

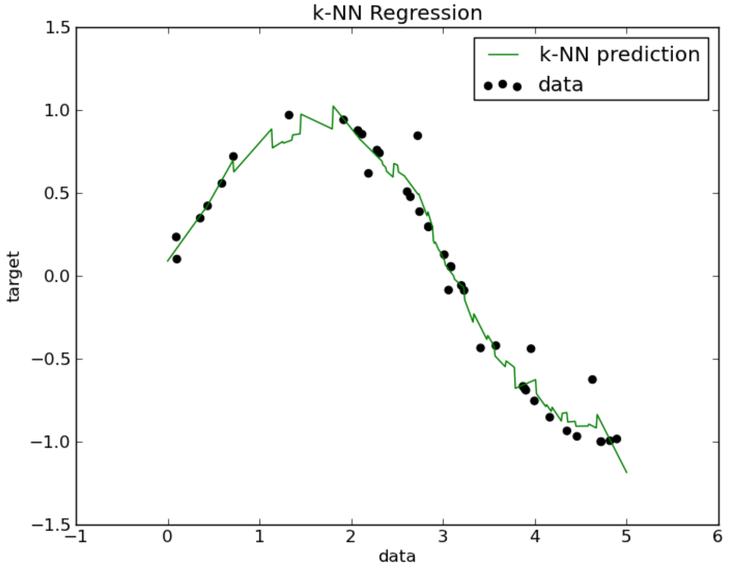
https://indico.cern.ch/event/1375252/timetable/#16-machine-learning

Regressor NN

- Regressor NN are used to predict a given variable from a set of input variables
  - Very useful when the relationships between variables are unknown or complex
- Training is relatively straight forward with a single output node, the loss is often the squared error!



$$Loss = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$



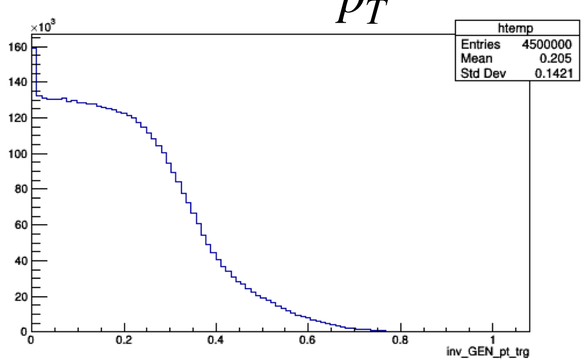
Training a Regressor NN

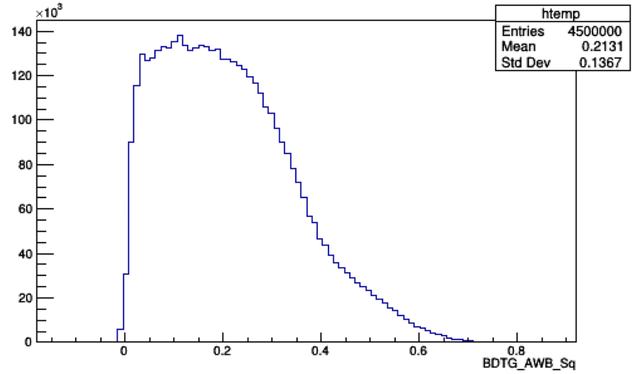
- When training any algorithm, a testing, training, and validation are randomly split to identify performance of the training
- Training data includes events whose output is already known
- The input data is fed into the NN for each event and the output layer is used directly in the loss function
- Based on the loss function, the NN will update its sets of weights and biases to minimize the loss

Target Variables

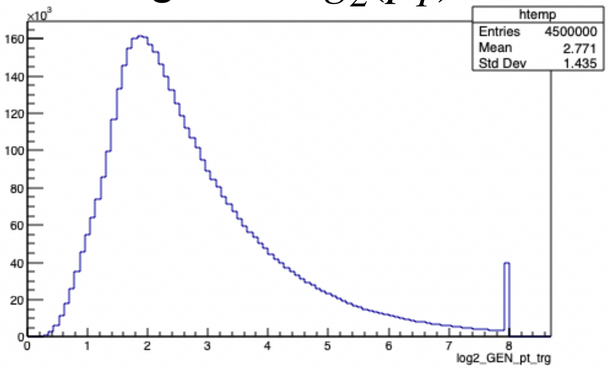
- With machine learning algorithms, the regression generally performs better with smooth output spaces
- The algorithm will tend to smooth over sharp peaks
- Playing with the target variable to choose the smoothest output space will improve performance!

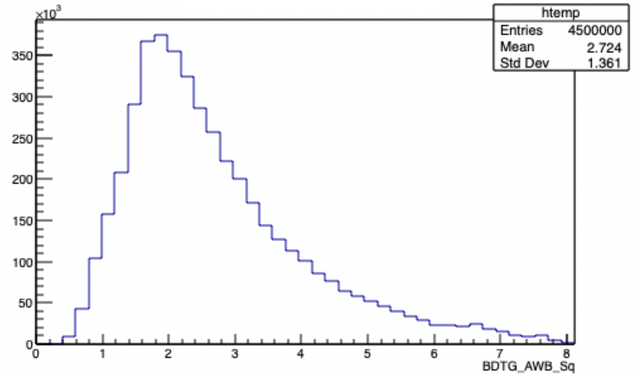
Target of $\frac{1}{p_T}$ → NN Prediction





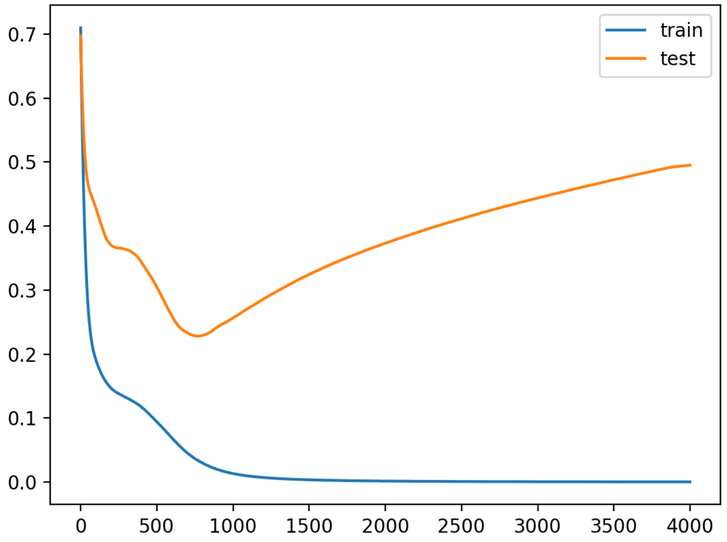
Target of $log_2(p_T)$ → NN Prediction

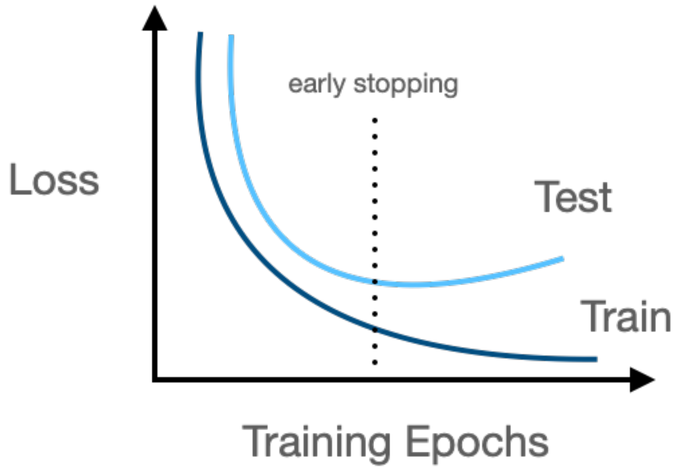




Overtraining or Overfitting

- **Overtraining** is one of the major problems that can go wrong with a NN! It is identified by a difference in loss or accuracy between training and validation sets
  - Can be caused by having too many unconstrained variables in the training (i.e. 4 input variables in a NN with 100 million connections)
  - Can also be caused by training too many iterations of the NN for the training dataset's size (i.e. training for 15,000 epochs on a dataset of 2000 events)
  - What is actually happening is the training set is being learned itself instead of the relationship between variables - Effectively the NN is memorizing the training set





Hyperparameters

- Regressor NN have many of the same hyperparameters as classifier NN with the addition of choosing the target variable!

- To name a few:
  - Batch Size
  - Number of times the NN updates (Number of Epochs)
  - Activation function of each layer (ReLu, Leaky ReLu, Tanh, etc)
  - Number of nodes in each layer
  - Early stopping (size of validation sets, patience, thresholds)
  - Event weights
  - Loss function
  - Optimizer Algorithm
  - Layer types (Dense, Sparse, Quantized)

The Tutorial

- We will be training a regressor NN to learn to regress a muons momentum in CMS based on hits in the muon chambers!
- We will be looking at simulated events that contain many variables given from the CSCs in the endcap
- The main metric we will be plotting will be the resolution of our regressor!

https://github.com/Offshell-Workshop-LPC/Offshell-Workshop-ML-Tutorials/tree/main/Tutorial_2