A Look at the Multilayer Perceptron

With application in particle physics

Justin Anguiano

University of Kansas

July 16, 2020

Introduction

Presenting : A look at the elementary concepts of an Articficial Neural Network (ANN)

Goal : to build intuition on how to use ANN

Outline:

1. Familiarize concepts of Multilayer Perceptron
2. Introduce basic particle physics classification problem
3. Highlight some challenges in real life application

ML Concepts – Introduction

What is an ANN?

Computing systems vaguely inspired by the biological neural networks that constitute animal brains.

What is a MLP?

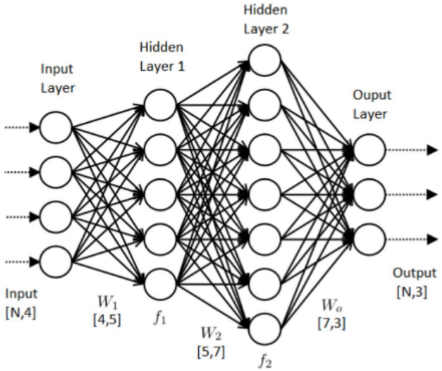
- a class of feedforward artificial neural network (ANN)
- at least three layers of nodes: an input layer, a hidden layer and an output layer
- each node is a neuron that uses a nonlinear activation function

What does it do?

- Its multiple layers and non-linear activation enable it to distinguish data that is not linearly separable
- Typically used in classification problems

ML Concepts – MLP Breakdown

Common depiction of an MLP
Lets dive into each element of this figure



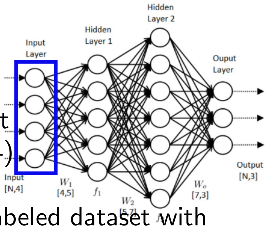
`https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4`

ML Concepts – MLP Breakdown

Input layer

This layer represents the input data matrix

- Data contains N entries
- Each node represents a feature of a data element
- A data entry is described by 4 variables $\Rightarrow (N, 4)$
- Data traverses network 1 row at a time



Toy Example: (Identifying fish) Suppose we have a labeled dataset with various fish – We want to use the network to distinguish the fish

Feature Matrix (X):

$$\begin{bmatrix} Size & Color & FinType & Location \\ Size & Color & FinType & Location \\ \vdots & & & \\ Size & Color & FinType & Location \end{bmatrix}$$

Data Labels (Y):

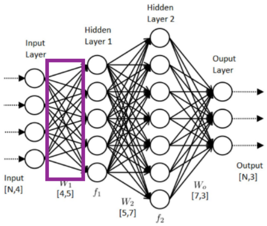
$$\begin{bmatrix} Tuna \\ Salmon \\ \vdots \\ Shark \end{bmatrix}$$

ML Concepts – MLP Breakdown

Network Weights

The lines between nodes are simply matrix multiplication

- This represents the linear combination part of the network
- Each feature is weighted and mapped to nodes in the hidden layer
- Weight matrix dimension is such that it maps to the desired hidden layer dimension $(4, 5)$



Toy Example:(Identifying fish, i-th data element) $X_{[i,:]}W$

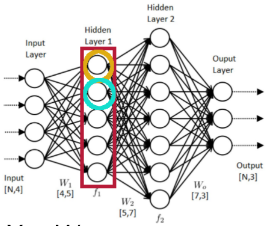
$$\begin{bmatrix} Size & Color & FinType & Location \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} & W_{15} \\ W_{21} & W_{22} & W_{23} & W_{24} & W_{25} \\ W_{31} & W_{32} & W_{33} & W_{34} & W_{35} \\ W_{41} & W_{42} & W_{43} & W_{44} & W_{45} \end{bmatrix}$$

ML Concepts – MLP Breakdown

Hidden Layers

Hidden nodes are weighted linear combinations of data features

- Can have as many hidden nodes as you want
- Each linear combination gets a nonlinear transformation
- Transformation through an activation function



Toy Example: Identifying fish, Hidden node contents $X_{[i,:]}W_{[:,j]}$

$$X[i,:]W[:,1] = W_{11}Size + W_{21}Color + W_{31}FitType + W_{41}Location$$

$$X[i,:]W[:,2] = W_{12}Size + W_{22}Color + W_{32}FitType + W_{42}Location$$

$\vdots$

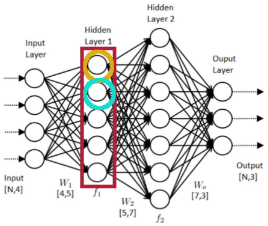
$X_{[i,:]}W_{[:,j]} = \sum_k W_{[k,j]}X_{[i,k]}$

ML Concepts – MLP Breakdown

Hidden Layers

Hidden nodes are weighted linear combinations of data features

+ Each linear combination gets a nonlinear transformation
+ Can be any non-linear function



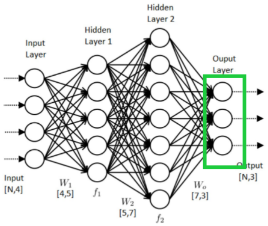
A popular activation function: RelU $f\left(X_{[i,:]}W_{[:,j]}\right)$

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

ML Concepts – MLP Breakdown

Output Layer
Performs the classification task on the transformed input data

+ Typically a logistic regression layer
+ Outputs probability that data element belongs to a certain class(label)
+ Can scale to any number of classes



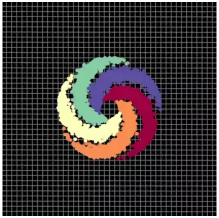
Toy Example: Identifying fish
Input $X_{[i,:]}$ Layers Label Prob.

$$Salmon \longrightarrow [HL] \rightarrow [OL] \longrightarrow \boxed{\begin{bmatrix} 0.95 & Salmon \\ 0.025 & Tuna \\ 0.025 & Shark \end{bmatrix}}$$

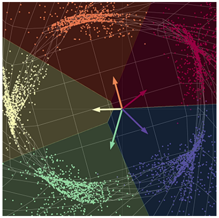
ML Concepts – MLP Recap

- Data feeds through the network row by row
- The element then undergoes a series of linear combinations and non linear transformations in each layer
- The output layer performs the classification task

Toy Example: Data Visualization



Pre-Network



Post-Network

https://atcold.github.io/pytorch-Deep-Learning/en/week01/01-3/

ML Concepts – Training

Training is finding the set of weights W in each layer that minimize the chosen loss function

+ Derivatives between consecutive layers are dependent
+ Must also be calculated backwards through the network "back propagation"
+ Popular loss function: cross entropy

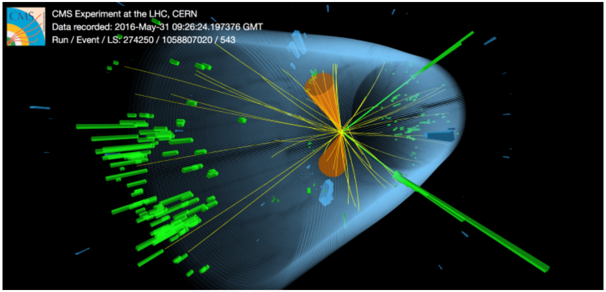
Cross Entropy:
$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$

- c represents class label of M classes
- $y_{o,c}$ is the true one hot label for data element o
- $p_{o,c}$ is the predicted probability of class c

Outline:

1. ~~Familiarize concepts of Multilayer Perceptron~~
2. Introduce basic particle physics classification problem
3. Highlight some challenges in real life application

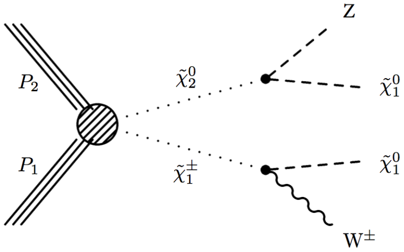
Physics Application – Introduction


CMS Experiment at the LHC, CERN
Data recorded: 2016-May-31 09:26:24.197376 GMT
Run / Event / LS: 274250 / 1058807020 / 543

- MLP can be used to aid in the search for particle dark matter
- Soft charged particles could be produced in rare dark matter events
- We use an ANN to distinguish charged particles of interest with particles that can imitate the desired signature

Physics Application – SUSY

KU CMS analysis in progress – searching for particle dark matter via compressed SUSY models



- Protons P_1, P_2 collide producing SUSY $\tilde{\chi}_1^\pm$,$\tilde{\chi}_2$
- SUSY $\tilde{\chi}_1^\pm$,$\tilde{\chi}_2$ decays to D.M. $\tilde{\chi}_1^0$ and known particles $W^\pm$,Z
- $W^\pm$,Z immediately decay into charged particles($\mu^\pm$) that we see in the detector

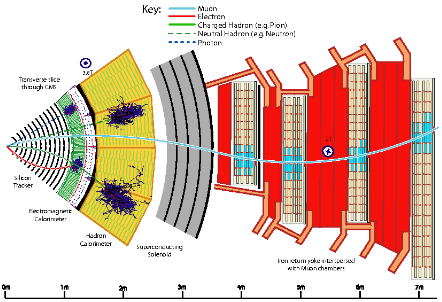
A compressed scenario implies $\tilde{\chi}_1^\pm$,$\tilde{\chi}_2$ and $\tilde{\chi}_1^0$ are very close in rest mass

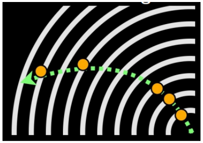
With compression the decay products of $\tilde{\chi}_1^\pm$,$\tilde{\chi}_2$ are soft (low momentum), including ending charged particles

The current CMS detector is less optimized for correctly identifying soft $\mu^\pm$

If we can optimize soft charged particle classification, we have a better chance of discovering compressed $\tilde{\chi}_1^\pm$,$\tilde{\chi}_2$, and $\tilde{\chi}_1^0$

Physics Application – Charge Particle Reconstruction


Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
2T
0m
1m
2m
3m
4m
5m
6m
7m



Charged particles bend in Mag. field and create "tracks"

Tracks are connecting the dots: "hits" that are fit with a curve

Main particle of interest is the Muon($\mu^{\pm}$)
– at high energies μ is easily correctly identified
– low energies leaves room for ambiguity

Sometimes other particles can be reconstructed incorrectly as a muon
– Common fakes: Pion($\pi^{\pm}$), Electron($e^{\pm}$), Kaon($K^{\pm}$), Proton(p), or non physical junk particles
– created from punch through
– junk particles are a result of hit combinatorics

Physics Application – ML Model

- Use fully simulated processes to get collections of reconstructed muons
  - This collection has reconstructed labels and truth(Gen.) labels
  - Sometimes reconstructed can't be matched to a truth label, these are treated as a separate class "Unmatched"
- Attempt to identify true muons(Class 0) against Unmatched(Class 1)
- Network inputs are measured quantities and track quality metrics

Physics Application – Network Architecture

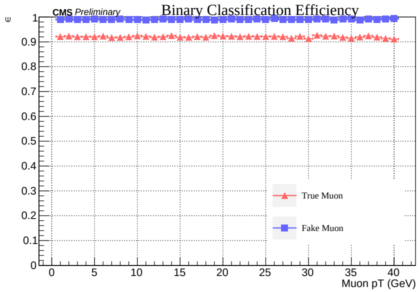
Architecture

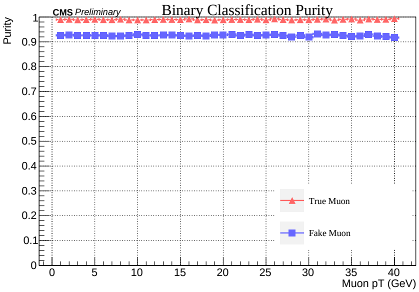
- 4 hidden layers
- 128 neurons/layer
- ReLU activation
- Softmax output layer
- Adam optimizer
- Categorical cross entropy loss

Training

- 10/90 validation/training split
- 35 Chunks
- 50 epochs, 2 epochs/chunk
- 100 batch size/chunk
- Class0/Class1 even sampling
  - 1.5M True Muons
  - 1.5M Unmatched

Physics Applications – Model Results





- Results shown as a function of muon momentum
- Class labeled when $P_{class} > 50\%$
- Efficiency for true muons $\sim 92\%$
- Tested on data sampled the same as training
- Efficiency $= \frac{TP}{TP+FN}$ Purity $= \frac{TP}{TP+FP}$

Outline:

1. ~~Familiarize concepts of Multilayer Perceptron~~
2. ~~Introduce basic particle physics classification problem~~
3. Highlight some challenges in real life application

Challenges – (1) Building your network

- What input variables to choose?
- How big should the network be?
- How much data do you need?
- What values for model hyperparameters do I choose?
- What loss/activation functions are best?

No good answer – just guess and check many configurations

Challenges – (2) Normalization

Variables in the input data need to be small

After a sequence of summations, inputs become large

In the final layer, with logistic regression, that large number is exponentiated

Easy to go beyond the double precision limit and break the model

$$P(C_i = j) = \frac{e^{X[i,:]\, W[:,\, j]}}{\sum_{\ell=1}^{k} e^{X[i,:]\, W[:,\, \ell]}}$$

Challenges – (3) Really Big Data

For our case we needed very large statistics to properly train

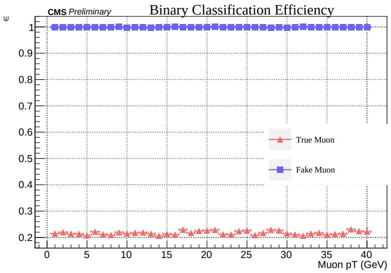
Data is too big to load into memory and train all at once

Load data in "chunks"

Chunks are trained iteratively

Model is tuned to learn more slowly

If weights change too rapidly model will have forgotten the first chunk when it gets to the last



Example:

Model trained asymmetrically

Chunk0 90% class 0 and 10% class 1

Chunk1 10% class 1 and 90% class 0

The End.