Term Project Report: Distinguishing between background and signal data in H→ $\tau\tau$ decay using Artificial Neural Networks(ANN)

Akhilesh Tayade
email: s6aktaya@uni-bonn.de
Rheinische Friedrich-Wilhelms-Universität Bonn

Abstract

ANN is a machine learning(ML) technique used in both astronomy and particle physics to separate data and noise. Implementation of ANN to distinguish between background and signal data for Higgs to $\tau\tau$ is discussed, and the efficiency for various activation functions and cost functions is presented.
Keywords: ANN, Higgs, cost function, activation function.

1 Introduction

A percepton is a crude model of a biological neuron. It takes an array of input, weighs them and gives a scalar output. An ANN is a directional network of perceptons. The input layer takes the input, passes it's outputs onto next layer of perceptons(called the hidden layer), which in turn, generate inputs of the final layer. The final layer can have a single (in case of binary classification) or multiple outputs. If the ANN has more than 1 hidden layer, it is called a deep neural network. The weights of each node in the ANN are 'learned' using gradient descent method, which uses a *cost* or *loss function* J . Learning constitutes two important steps: forward pass of outputs and back propagation of errors.

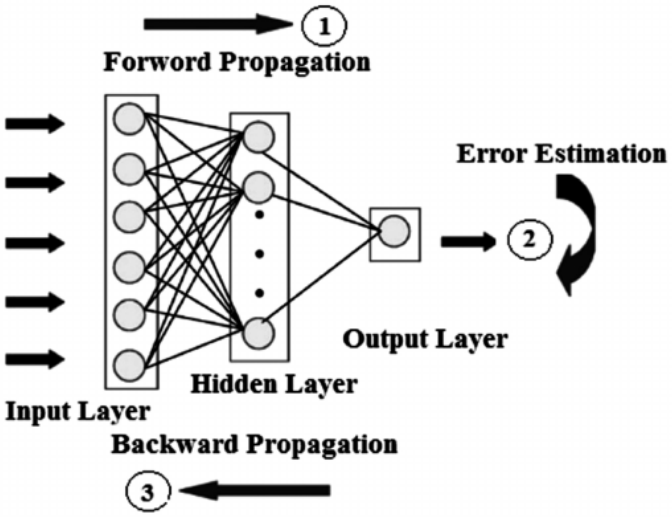

Figure 1: Visualization of steps in back propagation algorithm.[1]

1.1 Feature Selection

The data provided has been collected from the ATLAS experiment at CERN. We divide the data into training(≈ 70k) and testing sample(≈ 30k). We use training sample to train our ANN and the testing sample to evaluate it. Each of the 100k data points have 31 Features(columns) and one of them has the label 'class' = 1 if it is signal or 0 if it is background. Notwithstanding, here we use only 3 most discriminating of the other 30 features as input to ANN. Discriminating power can be determined using the Kolgomorov- Smirnov test [2]. Higher is the KS statistic between two distribution, less likely are they to be from the same pdf. In our case the most discriminating features are

`m_pp,m_wbb,m_wwbb`

ANN has another advantage - it is not necessary to completely understand the physical process in order to separate background and signal. In order to not bias the weights heavily towards any of the features, the data is also normalized before feeding it to the network.

2 Algorithm

2.1 Forward Pass

For each neuron k in a layer ℓ the output is

$$y_k^\ell = f(z_k^\ell),$$

where z_k^ℓ is the weighted sum of it's inputs

$$b_k^\ell + \sum_{i=1}^{n} y_i^{\ell-1} w_{k,i}^\ell$$

b are the biases equivalent to zeroth weight. They can be though as weights whose input is always one. The first layer takes data(features) as a input. The consequent layers have outputs of the previous layer as input.

Over a certain value of z, f should 'activate' it's output. Weight update (discussed in §2.2) is proportional to derivative of f, so f' has to be high where input is low and vice versa. For these reasons it is chosen to be sigmoid or sigmoid-like. Similarly, J should be proportional to output error.

2.2 Back Propagation

The weights are changed once J has been calculated for the entire data set. This makes one epoch. A common choice for cost function is $J = \frac{1}{2}(\hat{y} - y^L)^2$, where $\hat{y}$ is expected value and y^L is the output of the last layer. The weights are updated according to the equation

$$w_k^{*\ell} = w_k^\ell - \gamma \cdot \frac{\partial J}{\partial w_k^\ell}$$

where γ is the learning rate and $y^L \equiv y^L(y^{L-1}, y^{L-2}, ..)$. Learning determines how fast the network learns. Chain rule can be used to write J in terms of w_k^ℓ.

$$\frac{\partial C}{\partial w_{k,i}^\ell} = y_i^{\ell-1} \delta_k^\ell$$

,

$$\delta_k^\ell \equiv \left(\sum_j w_{j,k}^{\ell+1} \delta_j^{\ell+1} \right) \cdot f'(z^\ell),$$

where the index j runs over all the nodes in the $(\ell+1)$th layer [3]. Consequently, $w^{\ell+1}$ has to be updated before w_k^ℓ. The weights closer to last layer are updated first and then previous layers' weights are updated. Hence this step of using the cost function gradient is called back propagation. Weights are updated ϵ (number of epoch) times until the cost function is minimized and reaches a plateau.

2.3 Regularization

Regularization is performed to prevent over-fitting of weights. The cost function is rewritten as

$$J \equiv J + \frac{1}{2}\lambda \cdot \Sigma w^2,$$

where the summation runs over all weights. The weight update equation(§2.2) also gets modified accordingly. λ is a tuning parameter. Heuristically speaking, the ANN is penalized in proportion to the square of it's weights. This also prevents biasing towards an input.

3 Analysis

3.1 Network Configuration

The network's performance is evaluated for various choices of J and f while keeping the number of inputs(=3), hidden layer(=3) and nodes per hidden layer(=5) constant. Since a binary classifier is trained, the number of output nodes is 1. Increasing γ makes the learning faster at the risk of missing extrema in cost function and can cause it to oscillate. It was fixed at $\frac{2}{datasize}$. We increase number of epochs to 5000 as the computational cost is sufficiently low. Ideally, it should be much lower. λ was set at 0.05.

The network was trained for three Activation functions and one Cost function:

Cost Functions	Activation Functions
Quadratic: $\frac{1}{2}(\hat{y} - y^L)^2$	Sigmoid: $\frac{1}{1+e^{-z}}$ Softplus: $\ln(1+e^z)$ Gaussian: $\exp\frac{z^2}{2}$

3.2 Results

Introducing regularization stops the network from over-fitting it's weights by shunting the weight update amount. It's effect can be seen in figure 2. Including weights explicitly in the Cost Function makes it vulnerable to input biasing, which manifest as peaks in the right plot.

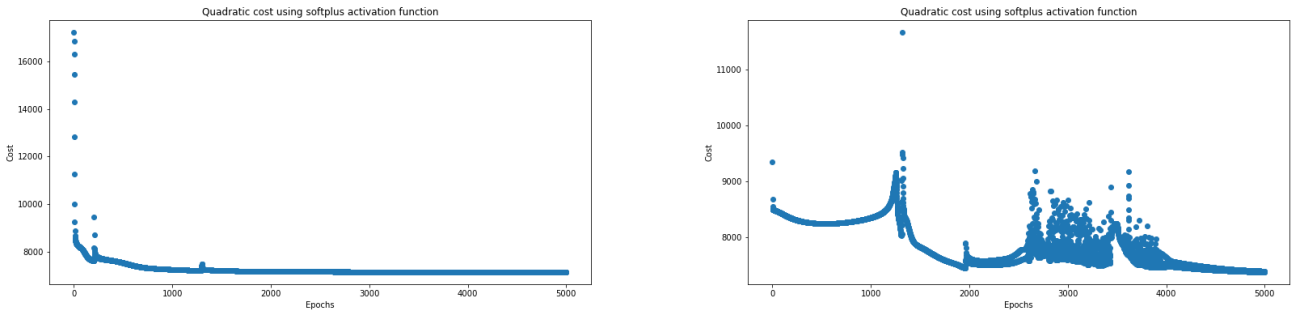

Figure 2: Cost Function minimization with(right) and without(left) Regularization.

We use Receiver Operating Characteristic (ROC) curves to study efficiency of the plots. It is a curve traced by True and False positive rates (on test data) over all thresholds on the predicted values. Area under the ROC curve (AUC) also provides an aggregate of the ANN's performance. AUC of a random classifier is 0.5 so an AUC for a network can also be used to compare it's performance with a random classifier.

ROC curve for the three most discriminating features have also been plotted. It is easy to see that for purity of signal, ANN prediction is far more efficient than any of them for all cost functions. For a higher TPR, all of them perform comparably.

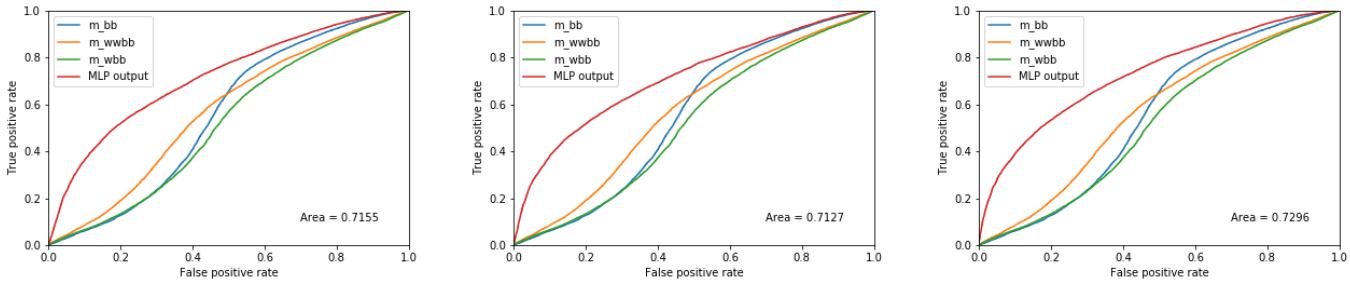

Figure 3: ROC curves (left to right) for Sigmoid, Softplus and Gaussian activation function

Gaussian has the maximum AUC, at 0.7296.

4 Conclusion

An ANN with regularization was implemented from scratch[4] to distinguish between noise and signal in Higgs to $\tau\tau$ decay. It's performance was analysed for various activation function using ROC and AUC. We found out that it performs better than performing cuts on features, especially for high signal purity.

ANN can be further improved by dynamically changing γ and λ as a function of epochs and cost. Different cost functions, such as binary cost entropy and various optimisers other than gradient descent can be implemented.

References

[1] https://smarterbeta.wordpress.com/2017/11/24/back-propagation-for-artificial\-neural-network-part-one/

[2] https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test

[3] Exercise 9, AMDA lectures.

[4] Code Github repository visible publicly at https://github.com/Ak-Bonn/ann_project