Pulse Shape Analysis of Gamma rays from NaI (Tl) Detector and Gamma-Neutron Differentiation using Machine Learning.

Harsh Kumar
hkumar2@ph.iitr.ac.in
En No – 18122008

Utkarsh Parkhi
uparkhi@ph.iitr.ac.in
En No – 18122030

Nikhil
nikhil@ph.iitr.ac.in
En No – 18122014

Abstract

This project presents the extraction of pulse shape, plot of pulse height distribution due to the pulses generated by Gamma and use of Machine Learning algorithms for classification of pulses due to Neutron and Gamma. The efficiency of classification system has been tested by three different sets of algorithms. The best overall accuracy achieved by the system is 99.71% percent which shows the potential of the system to be used for practical purposes.

1. Main Objectives

- Plot of pulse shape and pulse height distribution due to Gamma
- Use of ML algorithms to classify pulses due to Gamma and Protons.

2. Status and Other Details

- Completed
- Total time spent on project – 3 weeks

3. Major Stumbling Blocks

- Selection of ML algorithms for classification and adjustment of Hyperparameters.

4. Introduction

Scintillates are one of the oldest types of radiation detector because measurements could be made with photographic film. Images could be collected or intensity measurements could be made. Measurements were also made with the human eye observing the brightness of frequency of flashes in the scintillator. Nowadays the light output is converted into voltage pulses that are processed in the same way as pulses from proportional counters, semiconductor detectors etc. The whole point of scintillation detectors is that we want to produce a large light output in the visible range.

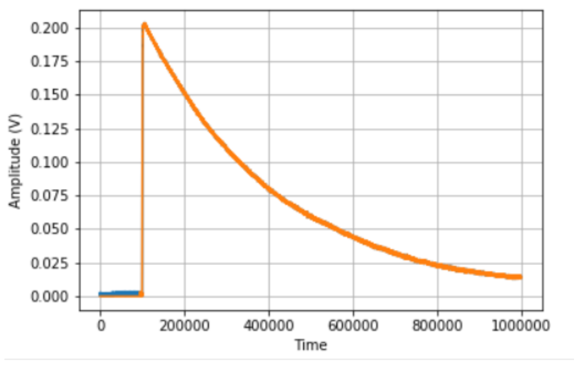
In this project we first present the plot of pulses due to Gamma incident on the Scintillation Detector and the plot of its pulse height distribution. In second part we present three different classification system for separation of pulses due to Neutrons and Gamma. The algorithms used for classification are Logistic Regression, Support Vector Machine and 8 different models of Artificial neural Networks each with a different set of hyperparameters.

The rest of the project goes as follows.

Section 5 discusses the plot of pulse and pulse height distribution due to gamma pulses. Section 6 describes the theory of Logistic Regression. Section 7 describes basic theory of SVM. Section 8 describes the theory of Artificial Neural Networks. Section 9 and 10 present the Confusion Matrix and ROC curve. Section 11 summarizes the experimental results and Section 12 concludes the entire project.

5. Pulse shape and pulse height distribution

The pulse detected by the scintillation detector cannot be directly used for machine learning applications; it must be sanitized and converted in a clean and clearly distinguishable form. This happens mainly due to noise in the data and multiple pulse pileup in the same graph. This report demonstrates a way to tackle this problem and applications of pulse-height distributions.



➢ Removing Noise

The raw data received from the scintillator detector contains noise which needs to be removed. The noise was removed by the savgol filter which is a useful function in signal processing and can be found in {library}.

➢ Adjusting The Baseline

The data received from the detector has a zero error which must be removed to do this all the data points are shifted up or down so that the lowest point corresponds to zero amplitude.

➢ Detecting Pileup Pulses

Our detector receives multiple pulse inputs but we consider only the first pulse so to remove the other pulses we must detect the other pulses.

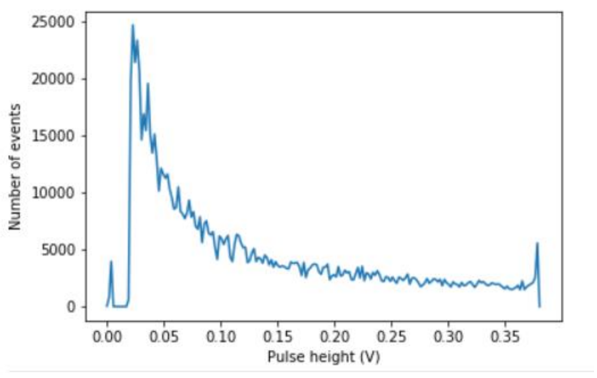
The abovementioned task was done by detecting a sudden increase in the amplitude.

➢ Removing Pileup Pulses

Once the pulses are detected the task to obtain the graph after considering only the first pulse still remains, this task can be done by observing the nature of exponential graphs and realizing the decay constant of both the pulses is equal.

The direct consequence of above statement is that both the graphs decrease at same rate i.e during the time interval the first graph decreases by a factor of $\boldsymbol{\gamma}$ the second graph also decreases by a factor of $\boldsymbol{\gamma}$.

So we can calculate a ratio **r** for each point. To calculate this ratio we just need to find this ratio when a new peak occurs and use the same ratio till the next peak. This statement is true because both the pulses have the same decay constant, as mentioned above.



6. Logistic Regression

Logistic regression is a statistical model that uses a logistic function for binary classification and comes under the Supervised learning technique. It outputs a number between 0 and 1 which represents the confidence of the model in classifying a certain input as class with label 1. Then depending on the manually selected threshold value, the probability is converted to 0 if it is below the threshold value and 1 if it is above it. A loss function is used to calculate the deviation of model prediction from actual results and backpropagation is applied to tune the parameters in the direction which reduces the loss function.

Let a training set S with features X_i and classification output d_i, of the form

$$S = \{(X_1, d_1), (X_2, d_2), \dots\dots, (X_n, d_n)\}$$

Where $X_i \in m$ and $d_i \in \{1,0\}$

Here n represents total number of samples and m represents total number of features.

Let $w = \{\theta_1, \theta_2 \dots, \theta_m\}$ be the set of parameters. First the independent variables are multiplied with the corresponding model parameters to get a variable z. Then this value is applied to sigmoid activation function which returns a number between 0 and 1 and it is considered as the probability that y = 1.

$$z = w^T X = \Sigma \theta_i X_i$$

$$P(y = 1 | x_1 .. x_m) = \frac{1}{1 + e^{-z}}$$

This probability is converted to either 0 or 1 depending on the threshold value as below:

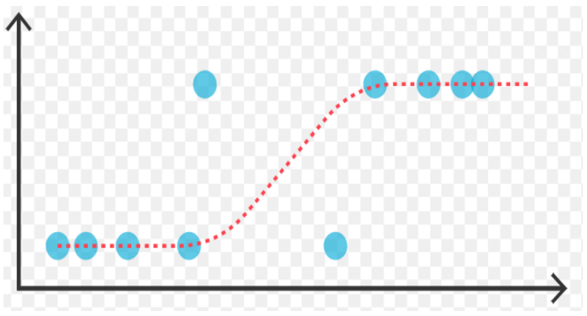
$$y_p = \{0 \; if \; P(y = 1 | x_1 \dots x_m) < Th$$

$$1 \; if \; P(y = 1 | x_1 .. x_m) > Th\}$$

Where Th is the threshold value generally kept as 0.5. The loss function used is log loss function as defined below:

$$L = \sum -y log y_p - (1 - y) \log(1 - y_p)$$

Where y is the actual label and y_p is the predicted label by the model. The goal of minimising the loss function in backpropagation is achieved by changing the weights and biases at each step in the direction of steepest descent of cost function.



Let $\frac{\partial L}{\partial w}$ be the derivative of the loss function w.r.t certain weight then the weight update is given by:

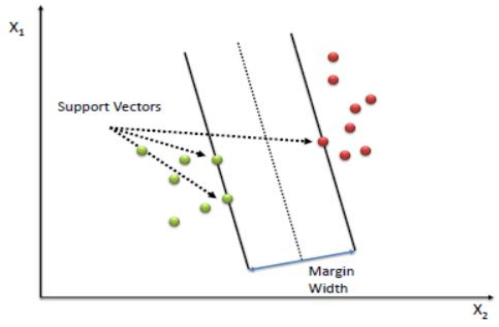
$$w = w - n\frac{\partial L}{\partial w}$$

where n is arbitrary learning rate. This process is repeated for certain number of steps called epochs until the parameters are tuned to the desired amount and accuracy of the model has improved significantly.

Regularisation is also used while tuning the parameters of the model. It prevents the parameters from assuming extremely high values and without regularisation, the asymptotic nature of the logistic regression would keep driving loss towards 0 in high dimension.

7. Support Vector Machines

The concept of Support Vector Machines (SVM) was originally introduced for binary classification problem. It finds a decision boundary or hyperplane such that it separates the two sets sets in such a way that the distance between the hyperplane and nearest point of each of the data sets (support vectors) is maximum [1].



The binary classification problem may be formulated as follows.

Given a training set S with input features X_i and classification output d_i, of the form

$$S = \{(X_1, d_1), (X_2, d_2), \dots \dots, (X_n, d_n)\}$$

Where $X_i \in R^n$ and $d_i \in \{+1, -1\}$

Here N represents total number of samples and m represents total number of features.

In SVM method, optimal margin classification for linearly separable input patterns is achieved by finding a hyperplane in m dimensional space [2]. The hyperplane must linearly separate the two classes {+1,-1} on its either side. The equation of the decision surface is given by

$$w^T X + b = 0$$

Here ***w*** is the weight vector and **b** is the hyperplane bias. The equations corresponding to the two classes can be represented mathematically as:

$w^T X + b \leq -1$ for $d_i = -1$

$w^T X + b \geq +1$ for $d_i = +1$

The support vectors are the training data points for which

$$w^T X + b \geq +1$$

that is, the points for which the corresponding inequalities are binding.

The distance between hyperplanes is $2/||w||$ and the problem is of maximizing this distance or minimizing $||w||^2$.

Thus, we can formulate the quadratic optimization problem:

Find w and b such that

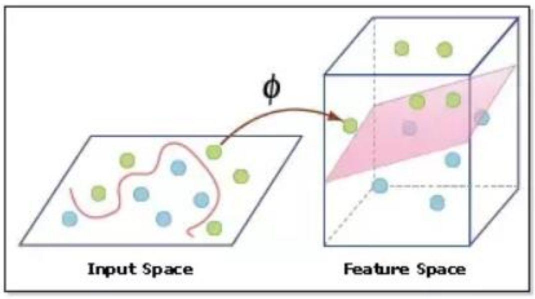
$$\phi(w) = ||w||^2 = w^T w$$

is minimized,

subject to the constraint

$$d_i(w^T X + b) \geq 1; i = 1 \; to \; N$$

The non-linear classification problem can also be solved by introducing concept of Soft Margin and mapping data to a High-Dimensional space by finding some kernel function Φ(x) for this purpose such that the problem changes into a linear classification problem (Figure 2).



Now the mathematical formulation becomes:

Min $\frac{1}{2}||w||^2 + C\Sigma\xi_i$, such that

$d_i(w^T\phi(X_i) + b) \geq 1 - \xi; \xi_i \geq 0$

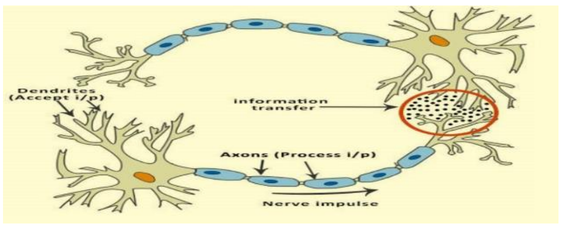
In soft margin problem, we introduce extra cost term to penalize for misclassified instances and those within the margin.

Though the concept of SVM was originally proposed for binary classification, various methods have been proposed to use SVM for multi-class problems also. “One against One” and “One against All” methods are among the most popular methods for multi-class classification problems [3]. The former involves constructing $^pC_2$ binary classifiers, one for each pair of a total of p classes. The final class of the test point is determined by a pre-defined voting mechanism. In the “One against All” method, there is a binary classifier for each class to separate the members of that class from all other classes.

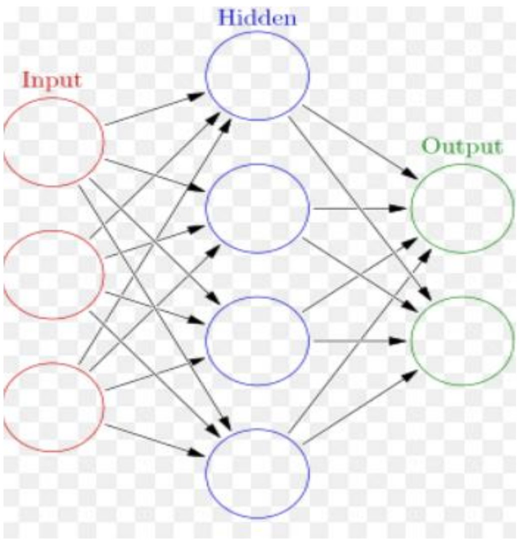
8. Artificial Neural Networks

Artificial Neural Networks are basically a computational model that is based on the structures and functions of biological neural networks.

Generally, the working of Human brain by making the right connections is the idea behind ANN. In brains the neurons are used to transmit messages and inputs are received from sensory organs.



Similarly, in ANN’s there are different layers consisting of various number of nodes. The first layer is the input layer and last layer is output layer. The remaining layers in between are hidden layers whose nodes contain specific information. All the nodes in certain layer are connected to all the nodes in the previous layer i.e the entire network is fully connected.



The ANN's can be used to perform variety of takes form regression to binary classification and multi classification. The processes performed by ANN are:

- Training Process
    - Feed Forward
    - Calculation of Loss function
    - Backpropagation to update weights.
- Testing process
    - Feed forward for predicting values or classes.

The classification problem can be formulated as:

Given a set S with input features X_i and classification output d_i of the form:

$$S = \{(X_1, d_1), (X_2, d_2), \dots \dots, (X_n, d_n)\}$$

Where $X_i \in R^m$ and $d_i \in \{0,1, \dots p\}$.

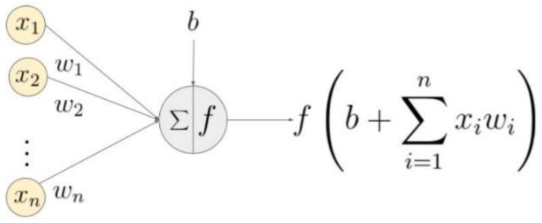
Here N represents total number of samples, m represents total number of features and p represents total number of classes.

It is to be noted that the total number of nodes in the input layer of ANN is equal to the total number of features and similarly total number of nodes in the output layer is equal to the total number of classes corresponding to each class.

Feed forward:

The feed forward has been explained in the paper [4].

Each connection between the nodes has a certain weight and bias.



Let n_{21} be the first node of second layer

n_{1i} be the nodes of first layer for $1 \le i \le k$

where k is the total number of nodes on the first layer.

The value n_{21} of is calculated as follows:

$$n_{21} = \Sigma(w_i n_{1i} + b_i)$$

Where w_i and b_i are corresponding weights and bias of the connection between ith node of first layer and 1st node of second layer.

The value of n_{21} generated is then passed to an activation function which determines whether the neuron should be fired or not [5]. It also introduces non-linearity in the network.

Some common activation functions used are:

- Sigmoid function – $f(x) = \frac{1}{1+e^{-x}}$

- Relu function – $f(x) = \max(0, x)$
- Hyper Tangent function – $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Binary Step function – $f(x) = \{0\ if\ x < 0\ and\ 1\ if\ x \geq 0\}$

The output obtained after passing through activation function is stored in the corresponding node and is used for calculation of node values of the further layers in the similar way as described above.

Since the output layer of ANN is used for determining classes, the activation function applied is such that it converts the values of all nodes in the final layer into probabilities which represents the confidence of ANN to classify a given input into certain class. The activation function most generally used for converting values of output nodes into probabilities is:

SoftMax activation function-

$$f(s)_i = \frac{e^{s_i}}{\sum e^{s_j}}$$

The node with the highest value of probability is taken as the class predicted by the ANN for a particular input.

Calculation of Loss function:

When modelling a classification problem, we are interested in mapping input variables to a class label and in order to improve the model's performance we define a loss function which gives a scalar measure of the deviation of model's output form the target values. Since we are dealing with classification problem here, the loss function used is:

Cross Entropy Loss function-

$$L = -\sum t_i \log(f(s)_i)$$

Where $f(s)_i$ is the softmax output of the ith node of output layer.

Our goal is to minimise the loss function in order to improve model's performance.

Backpropagation:

The backpropagation algorithm was originally introduced in the 1970s, but its importance wasn't fully appreciated until the famous paper [6]. That paper describes several neural networks where backpropagation works far faster than earlier approaches to learning, making it possible to use neural nets to solve problems which had previously been insoluble.

The goal of minimising the loss function in backpropagation is achieved by changing the weights and biases at each step in the direction of steepest descent of cost function.

Let $\frac{\partial L}{\partial w}$ be the derivative of the loss function w.r.t certain weight then the weight update is given by:

$$w = w - n\frac{\partial L}{\partial w}$$

where n is arbitrary learning rate.

The process of feed forward and backpropagation is repeated for certain number of times called as epochs, till the loss function has reduced by considerable amount.

After training the model, test inputs are fed and the class with maximum probability is taken to be the predicted value of model for a certain input.

9. Confusion Matrix

Confusion matrix is a N*N matrix which is used for evaluating the performance of a classification model, where N stands for the number of target classes. It comprises of the actual target values and those predicted by the classification model.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

- True Positives (TP) – The actual target was positive and the prediction was also positive.
- True Negatives (TN) – The actual target was negative and the prediction was also negative.
- False positives (FP) – The actual target was negative and prediction was positive.
- False Negatives (FN)– The actual target was positive and prediction was negative.

Some of the terms generally used with confusion matrix are: -

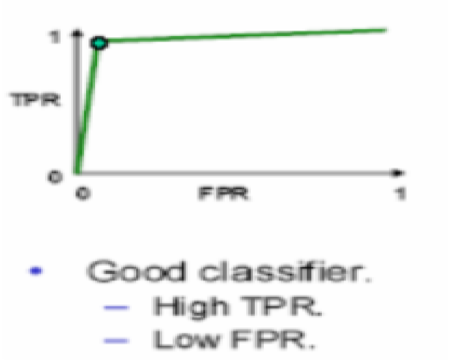
- Accuracy = (TP+TN)/N
- Error Rate = (FP+FN)/N
- Sensitivity = TP/(TP+FN)
- Specificity = TN/(TN+FP)
- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)

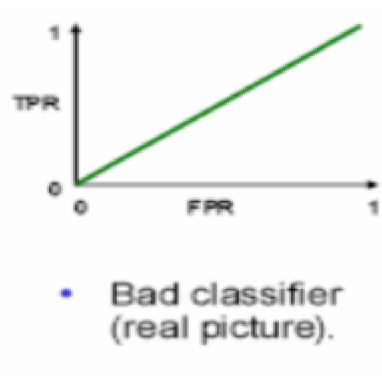
10. ROC Analysis

ROC stands for Receiver Operating Characteristic and is a plot of Sensitivity against (1-Specificity) calculated using the confusion matrix at different values of thresholds.

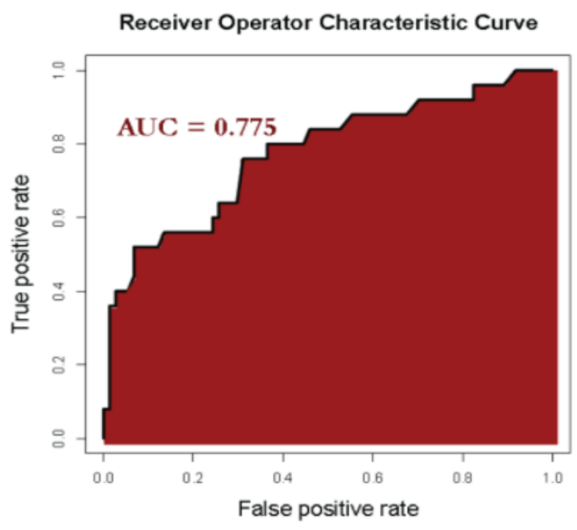
Sensitivity – True positive rate

1-Specificity – false positive rate





The area under the curve of a ROC curve is used a measure of performance of the model.



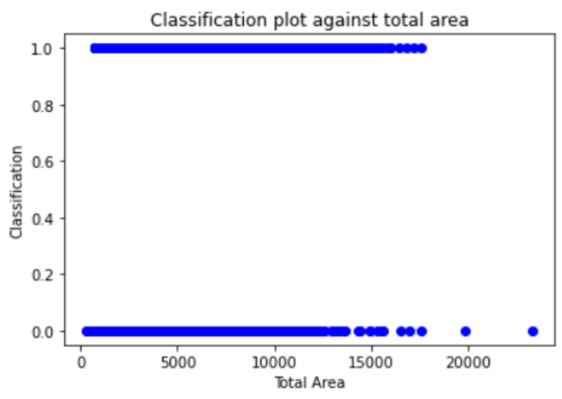
The greater the area under the curve, the better is the model.

11. Experimental Results

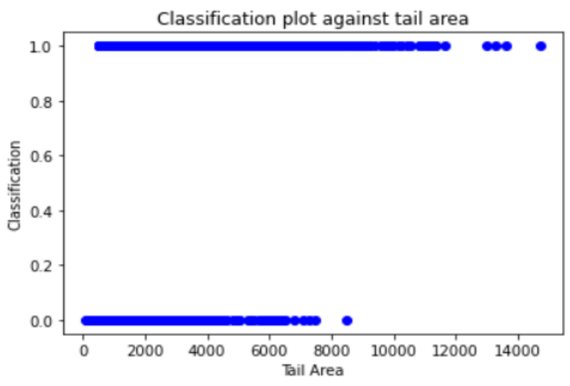
Dataset

The dataset used for our model has a total of 58122 samples. There are two features namely Total Area and Tail Area which have been extracted from the pulse. There are 2 classes namely Neutron and Gamma. The values of features were standardized. After encoding 0 represents Gamma and 1 represents Neutron. The dataset was split such that 75% of data was used for training and 25% data was used for testing all the various algorithms used.

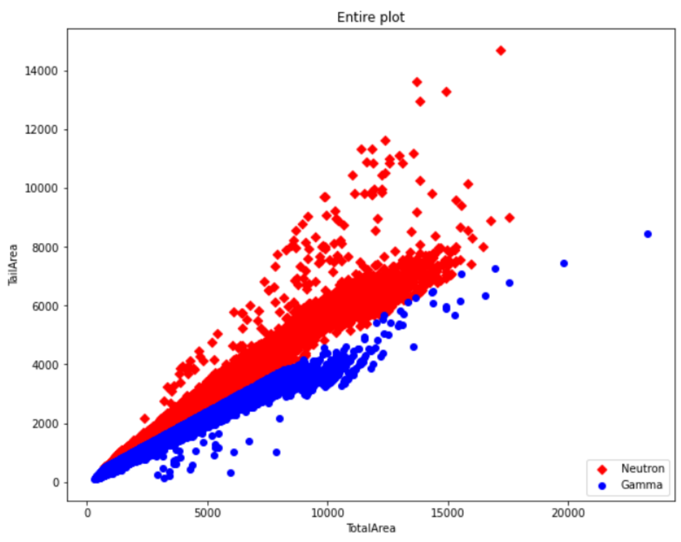
The plot of Classification against total area is as follows:



The plot of Classification against total area is as follows:



The plot of classification with total area on x-axis and tail are on y-axis is as follows:



12. Experimental Results

We deployed 3 different classification models and compared their accuracy.

- Logistic Regression
- Support Vector Machines
- Artificial Neural Networks

The confusion matrix for the test set obtained when Logistic regression model was used with threshold = 0.5 is:

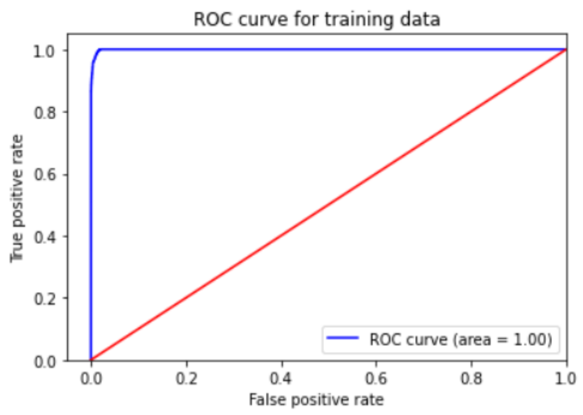
Pred | Actual	Actual = 1	Actual = 0
Pred = 1	10857	8
Pred = 0	428	3238

The best threshold value was calculated and the corresponding confusion matrix is:

Best threshold value = 0.316208

Pred | Actual	Actual = 1	Actual = 0
Pred = 1	10727	138
Pred = 0	47	3619

The ROC curve obtained:



The best accuracy obtained by using Logistic regression is: **98.726**

We used 3 kernels in SVM namely linear, rbf and polynomial. The accuracy obtained using each of the kernels are:

- Linear: **98.39**
- Rbf: **98.107**
- Poly: **90.51**

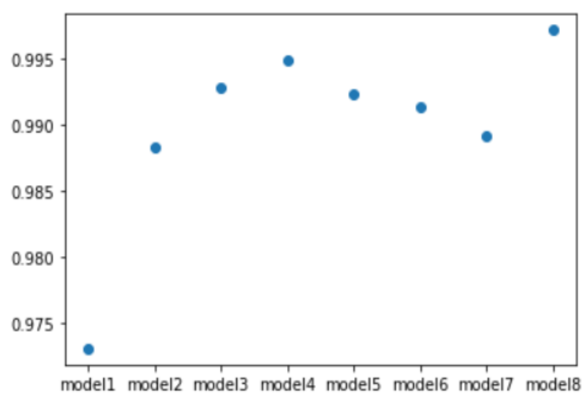
As linear kernel outputs the best accuracy, we select it as our best SVM classifier. The corresponding Confusion matrix obtained is:

Pred | Actual	Actual = 1	Actual = 0
Pred = 1	10826	5
Pred = 0	228	3438

The best accuracy obtained by using SVM classifier is: **98.39.**

We used 8 different models in Artificial Neural Networks each with different number of hidden layers and node in them.

The classification accuracy obtained on test samples by the ANN models are shown in graph below.



The hyperparameters of the models used are:

- Optimizer: Adam
- Learning Rate: 0.001
- Dropout Probability: 0.1
- Number of Epochs: 10

The model 8 with the following description achieves the highest accuracy of 99.71% .

The corresponding Confusion matrix obtained is:

Pred | Actual	Actual = 1	Actual = 0
Pred = 1	10852	13
Pred = 0	28	3638

Model 8 –

- Input Layer – 2 nodes
- Hidden Layer 1 – 128 nodes
- Hidden Layer 2 – 256 nodes
- Hidden Layer 3 – 512 nodes
- Hidden Layer 4 – 256 nodes

13. Conclusion

In this paper we designed three models for the classification of Neutron and Gamma depending on the Total Area and Tail Area of their pulses. The best accuracy we could obtain was **99.71** when Artificial Neural Network was used as classifier.

Refrences

1. Pedroso, J. P., & Murata, N. (2000). ‘Optimisation on support vector machines.’ *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*, July 24-27, 2000, Como, Italy,Volume 6 (pp 399-404).
2. Begg, R. K., Palaniswami, M., & Owen, B. (May 2005). ‘Support Vector Machines for Automated Gait Classification’, *IEEE Transaction on Biomedical Engineering*, vol 52, No. 5, (pp. 828-838).
3. Hsu, C.W. and. Lin, C.J.,(2002), ‘A comparison of methods for multi-class support vector machines’, *IEEE Transactions on Neural Networks*, 2002, Vol. 13, No. 2., pp 415-425.
4. Dawai Dai, Weimin Tan and Hong Zhan, *Feedforward Artificial Neural Network Model From the Perspective of Network Flow*: https://arxiv.org/ftp/arxiv/papers/1704/1704.08068.pdf
5. Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, ‘*Activation functions: Comparision of trends in practice and research for deep learning*: https://arxiv.org/pdf/1811.03378.pdf
6. Learning representations by back-propagating errors : https://www.nature.com/articles/323533a0

https://ieeexplore.ieee.org/document/467888