Machine Learning in Particle Physics

A B. Tech Mini-Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Technology

by

Aman Kumar
(180121002)

under the guidance of

Dr. Bipul Bhuyan

to the

DEPARTMENT OF PHYSICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

Contents

Chapter 1

Introduction

High-energy physics is a fertile area for applied research in machine learning and deep learning. The Large hadron collider generates humongous amount of data by colliding hadrons at very high velocities and recording the events by various detectors. The data about the events are extensively used by algorithms to classify particles and also find new exotic particles. Machine learning techniques have made significant progress in the classification metric which uses the best new approaches without the manual assistance .

1.1 Problem Description

Differentiating collisions which induce particles of interest a.k.a. Signals and those which produce other already known particles is called Background. Discovery of these particles is of paramount interest to the research community in physics [6].

The goal of the particle identification (PID) is to identify a type of a particle associated with a track using responses from different subdetectors [9] (detector systems).

1.2 Tracking System

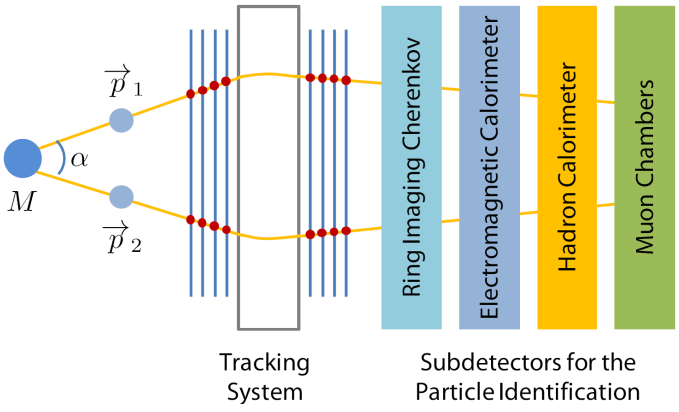


Fig. 1.1 tracking system and subdetectors

Tracking devices reveal the paths of electrically charged particles as they pass through and interact with suitable substances. Most tracking devices do not make particle tracks directly visible, but record tiny electrical signals that particles trigger as they move through the device. A computer program then reconstructs the recorded patterns of tracks.

Once a particle has passed through the tracking devices and the calorimeters, physicists have two further methods of narrowing down its identity. Both methods work by detecting radiation emitted by charged particles. When a charged particle travels faster than light does through a given medium, it emits Cherenkov radiation at an angle that depends on its velocity. The particle's velocity can be calculated from this angle.

$$\cos\theta = \frac{1}{n\beta} = \frac{\sqrt{p^2 + m^2c^2}}{np}$$

momentum of a particle:

$$p = \frac{mc\beta}{\sqrt{1-\beta^2}}$$

Velocity can then be combined with a measure of the particle's momentum to determine its mass, and therefore its identity. When a fast charged particle crosses the boundary between two electrical insulators with different resistances to electric currents, it emits transition radiation. The phenomenon is related to the energy of the particle and so can distinguish different particle types.

Collating all these clues from different parts of the detector, physicists build up a snapshot of what was in the detector at the moment of a collision. The next step is to scour the collisions for unusual particles, or for results that do not fit current theories.

1.3 Organization of The Report

This chapter introduces us to the use of machine learning in HEP and to the problem statement. The next chapter provides details of the problem and the ROOT CERN data for constructing a classification model. After that, we discuss TMVA and the top performing algorithms and others in brief for modelling our classification algorithm. And finally, we conclude our analysis with result, discussion and implemented code.

Chapter 2

Data-set Description

The given data consists of *signal* (S) and *background* (B) root files. Each has a tree "h1" containing 188 branches both for the signal and the background files. There are 931219 entries of each branch present in the background; on the other hand, there are 144804 entries per branch in (S)

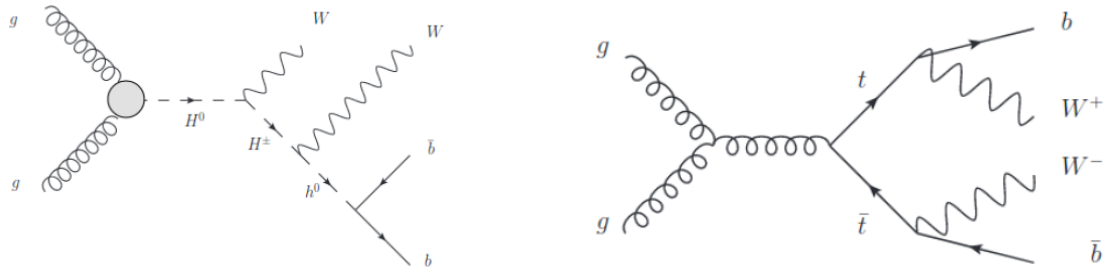


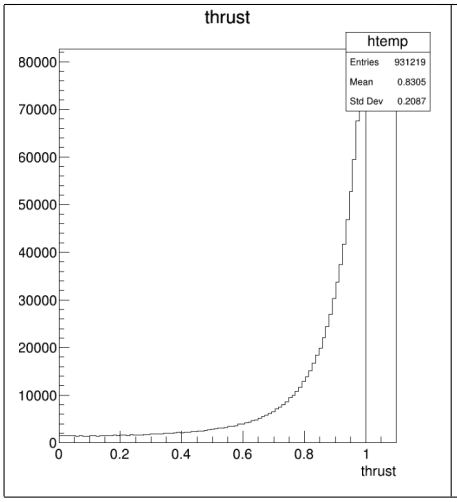
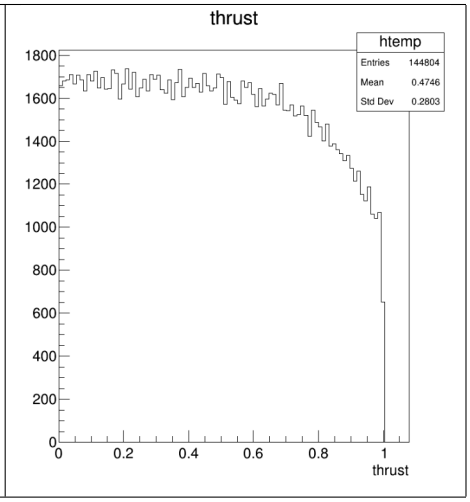
Fig. 2.1 signal and background processes

2.1 Analysis

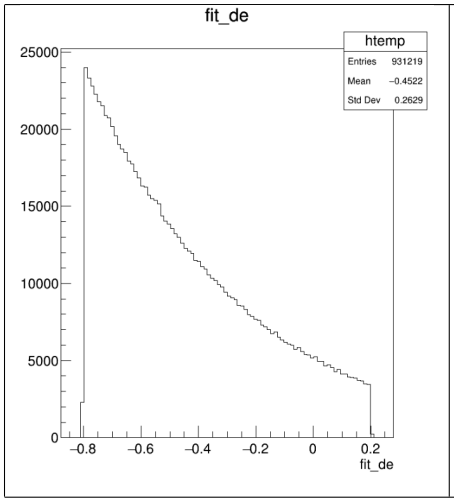
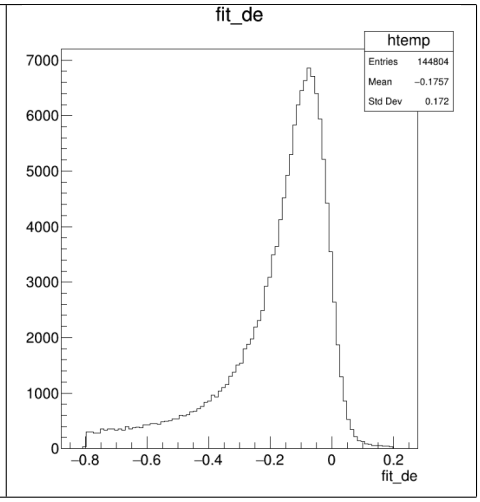
The distributions of these 188 features are carefully observed with the help of *THistogram* [4] present inside the ROOT framework. The distribution of the properties for the signal data which distinct from the distribution of the background data, by mean and std deviation [2], are chosen for the purpose of training the model.

The selected set of input features are listed:

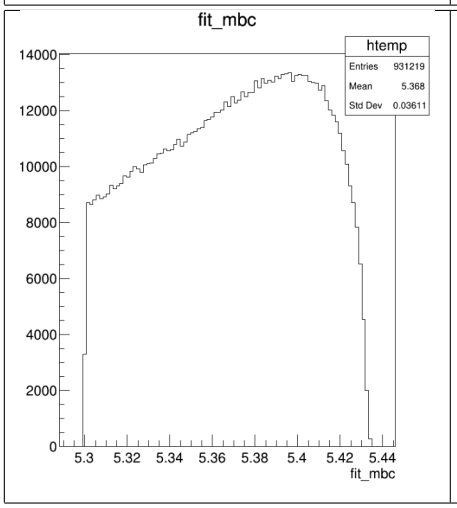
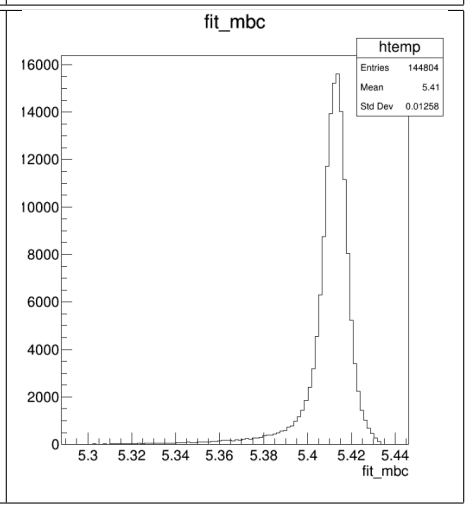
1. thrust

Background	Signal
	

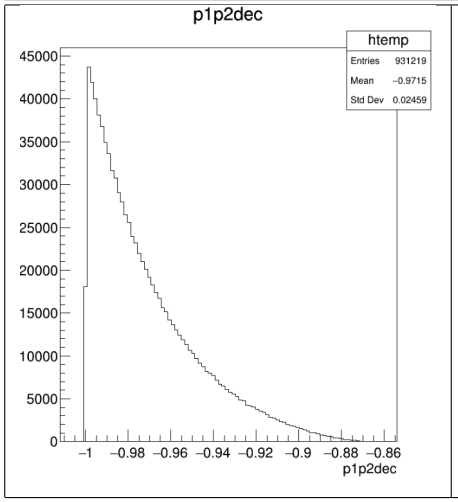
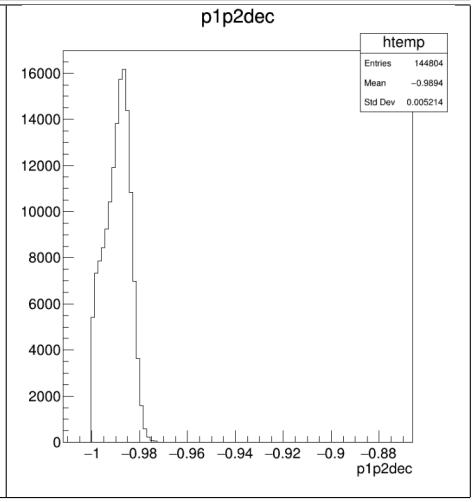
2. fit_de

Background	Signal
	

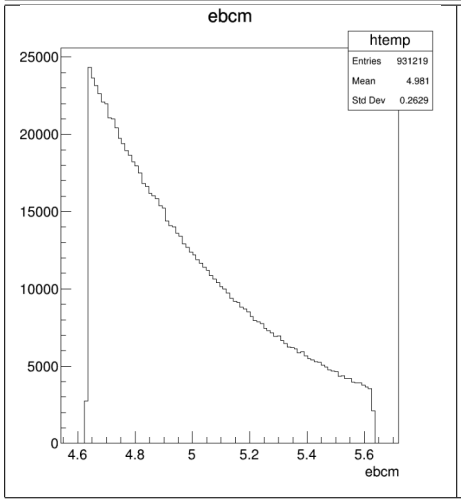
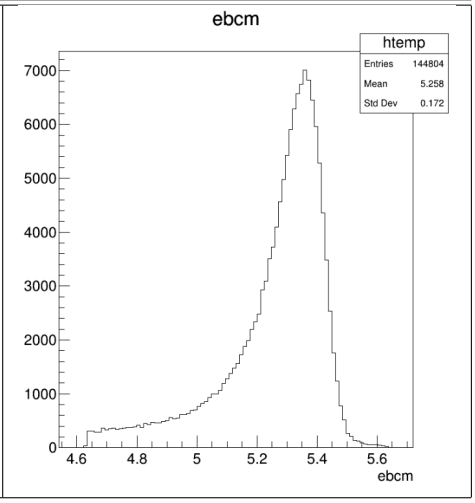
3. fit_mbc

Background	Signal
	

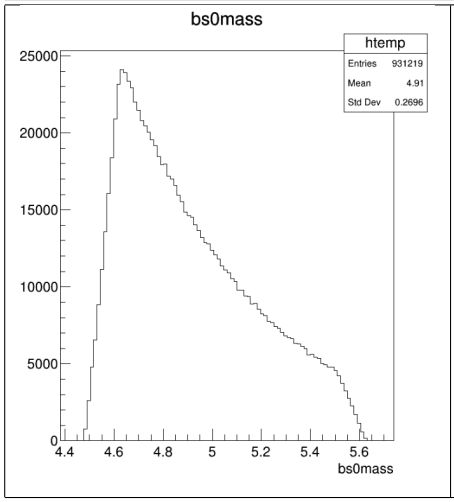
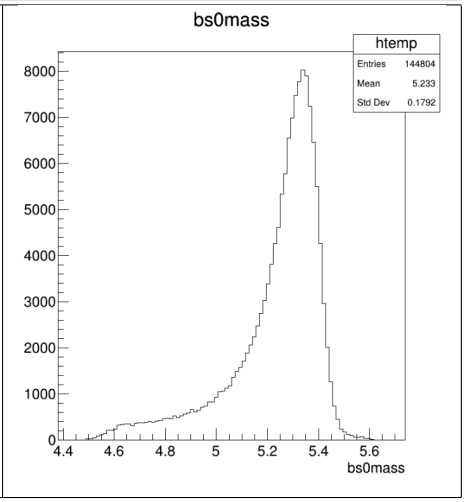
4. p1p2dec

Background	Signal
	

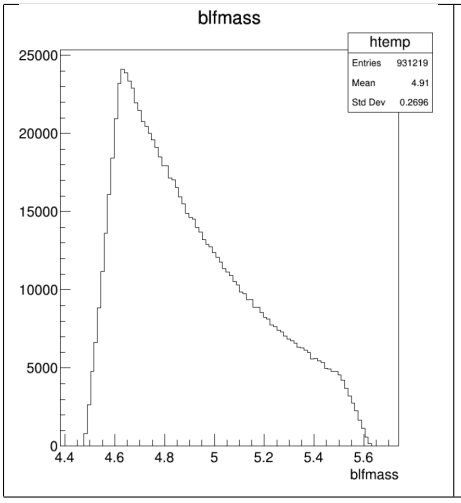
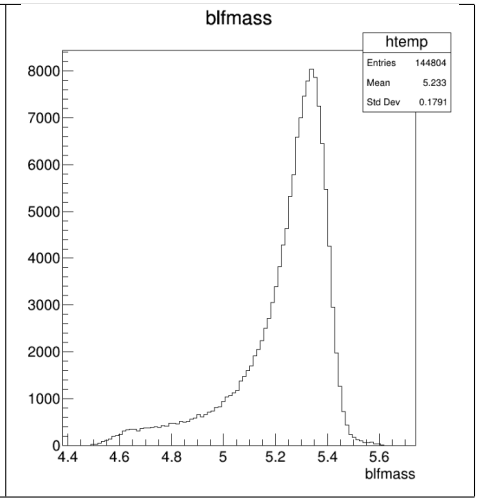
5. ebcm

Background	Signal
	

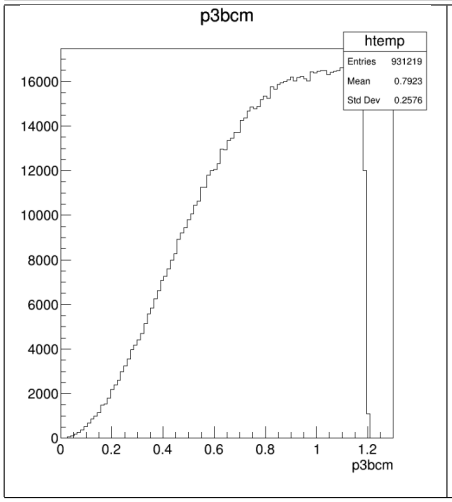
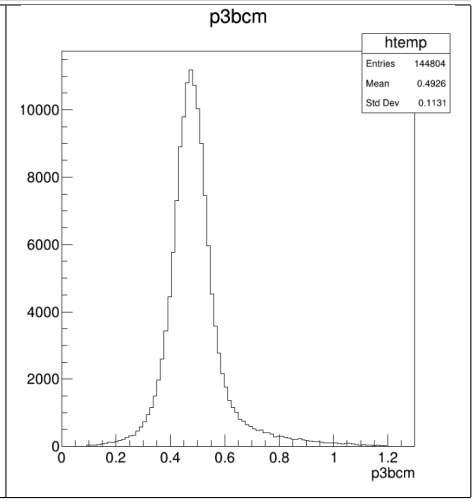
6. bs0mass

Background	Signal
	

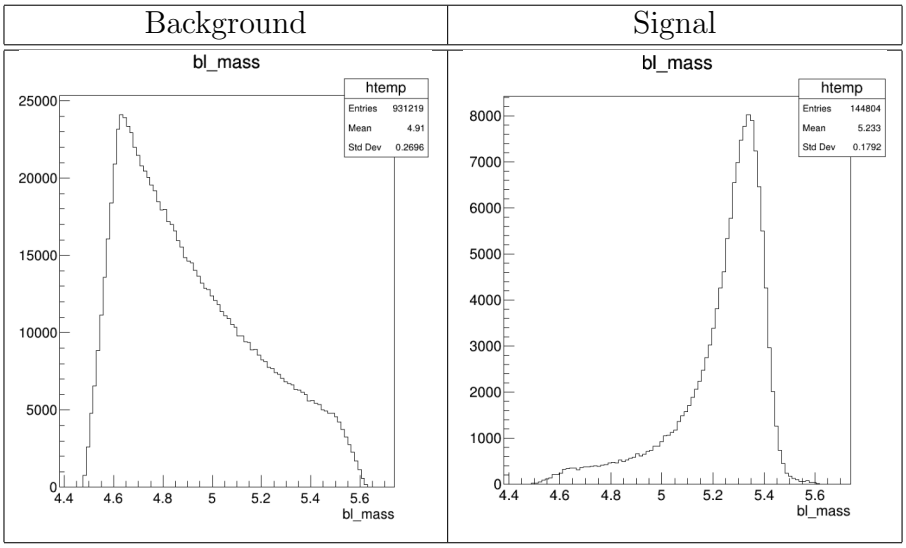
7. blfmass

Background	Signal
	

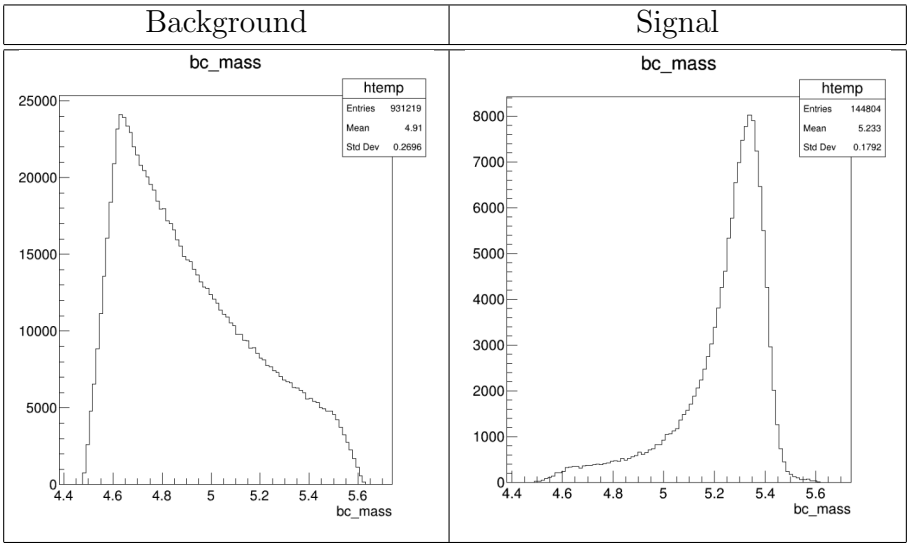
8. p3bcm

Background	Signal
	

9. bl_mass



10. bc_mass



Chapter 3

Algorithms

3.1 TMVA

Toolkit for Multivariate Data Analysis (TMVA) is a toolkit which performs a large variety of multivariate classification algorithms. It has been manipulated to multivariate regression of a real valued target vector. TMVA toolkit is constructed for machine learning applications in the field of high-energy physics. TMVA methods have highly boosted algorithms which results in a possible powerful committee method that consolidate exquisite properties[3].

The Toolkit for Multivariate Analysis (TMVA) provides a ROOT-integrated environment for the processing, parallel evaluation and application of multivariate classification and multivariate regression techniques. All multivariate techniques in TMVA belong to the family of "supervised learning" algorithms. They make use of training events, for which the desired output is known, to determine the mapping function that either describes a decision boundary (classification) or an approximation of the underlying functional behaviour defining the target value (regression). The mapping function can contain various degrees of approximations and maybe a single global function, or a set of local models [1]. TMVA is specifically designed for the needs of high-energy physics (HEP) applications.

3.2 MLP

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN [10], sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation).

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer, when they have a single hidden layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. In our case, we have used

$$y(v_i) = tanh(v_i)$$

activation function to have symmetry around zero, and the input, output and hidden layers have mean values of 0 and standard deviation of 1. Choosing tanh activation function also helps us avoid vanishing gradient problem [5] and enhances the training speed. MLP utilizes

a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

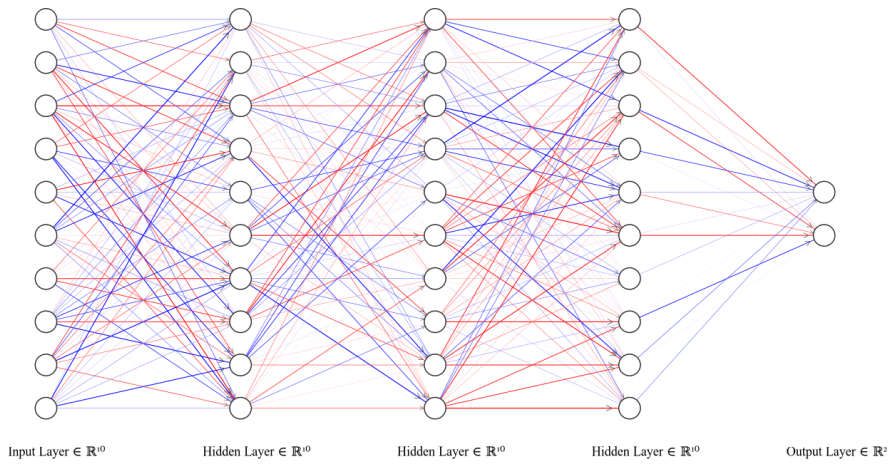


Fig. 3.1 multi layer perceptron

3.2.1 Learning

Our MLP model consists of an input and an output layer with three hidden layers of nonlinearly-activating nodes. Since MLPs are fully connected, each node in one layer connects with a certain weight w_{ij} Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. to every node in the following layer, and is carried out through backpropagation. [8] the degree of error in an output node can be represented by $e_j(n) = d_j(n) - y_j(n) e_j(n) = d_j(n) - y_j(n)$

$$\mathcal{E}(n) = \frac{1}{2} \sum_j e_j^2(n) \mathcal{E}(n) = \frac{1}{2} \sum_j e_j^2(n).$$

Using gradient descent, the change in each weight is

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

where $y_i y_i$ is the output of the previous neuron and η is the learning rate, which is selected to ensure that the weights quickly converge to a response [10], without oscillations.

The MLP neural network is booked through the TMVA [3] Factory via the command line:

```
  factory.BookMethod(dataloader, TMVA.Types.kMLP, "MLP",
"H:!V:NeuronType=tanh:VarTransform=N:NCycles=1000:HiddenLayers=N+5:\
TestRate=5" );
```

3.3 Boosted Decision Trees

Boosted decision trees have been successfully used in High Energy Physics analysis for example by the MiniBooNE experiment [11]. In Boosted Decision Trees, the selection is done on a majority vote on the result of several decision trees, which are all derived from the same training sample by supplying different event weights during the training. The phase space is split this way into many regions (nodes): the algorithm is repeated recursively on each node, classifying events as Signal or Background, depending on the majority of training events that end up in the final leaf node.

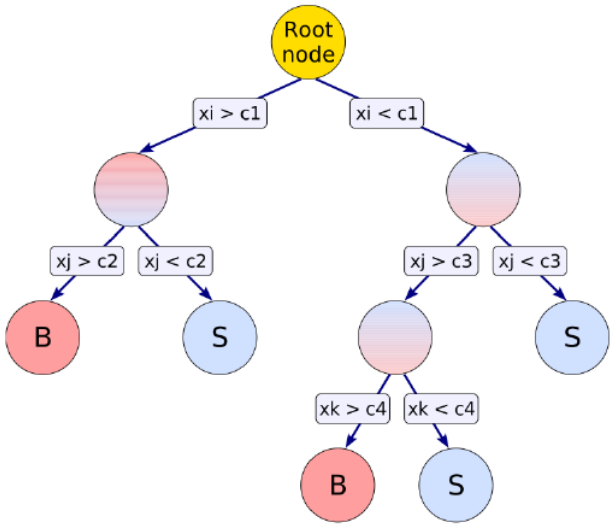


Fig. 3.2 Decision Tree

Successive decision nodes are used to categorize the events out of the sample as either signal or background. Each node uses only a single discriminating variable to decide if the event is signal-like ("goes right") or background-like ("goes left"). This forms a tree like structure with "baskets" at the end (leave nodes), and an event is classified as either signal or background according to whether the basket where it ends up has been classified signal or background during the training. Training of a decision tree is the process to define the "cut criteria" for each node. The training starts with the root node. Here one takes the full training event sample and selects the variable and corresponding cut value that gives the best separation between signal and background at this stage. Using this cut criterion, the sample is then divided into two subsamples, a signal-like (right) and a background-like (left) sample. Two new nodes are then created for each of the two sub-samples and they are constructed using the same mechanism as described for the root node.[7] The devision is stopped once a certain node has reached either a minimum number of events, or a minimum

or maximum signal purity. These leave nodes are then called "signal" or "background" if they contain more signal respective background events from the training sample.

3.3.1 Boosting

The boosting of a decision tree extends this concept from one tree to several trees which form a forest; [4] an event is classified on the basis of a majority vote done by each tree of the forest.

Four different boosting algorithms are available for decision trees in TMVA

1. Ada(ptive) Boost
2. Gradient Boost
3. Bagging
4. Randomised Trees

The idea behind adaptive boosting (AdaBoost) is, that signal events from the training sample, that end up in a background node (and vice versa) are given a larger weight than events that are in the correct leave node. This results in a re-weighed training event sample, with which then a new decision tree can be developed. The boosting can be applied several times (typically 100-500 times) and one ends up with a set of decision trees (a forest). Gradient boosting works more like a function expansion approach, where each tree corresponds to a summand. The parameters for each summand (tree) are determined by the minimization of a error function (binomial log- likelihood for classification and Huber loss for regression).[7] A greedy algorithm is used, which means, that only one tree is modified at a time, while the other trees stay fixed.

The BDT is booked through the TMVA Factory via the command line:

```
  factory.BookMethod(dataloader, TMVA.Types.kBDT, "BDT",
"!H:!V:NTrees=1000:nEventsMin=400:MaxDepth=7:BoostType=AdaBoost:\
SeparationType=GiniIndex:nCuts=20:PruneMethod=NoPruning" );
```

The Boosted Decision Trees with gradient boosting is booked through the TMVA [3] Factory via the command line:

```
factory.BookMethod(dataloader, TMVA.Types.kBDT, "BDTG",
"!H:!V:NTrees=1000:BoostType=Grad:Shrinkage=0.30:UseBaggedGrad:\
GradBaggingFraction=0.6:SeparationType=GiniIndex:nCuts=20:\
PruneMethod=CostComplexity:PruneStrength=50:NNodesMax=5" );
```

3.4 k-Nearest Neighbour

KNN method compares an observed (test) event to reference events from a training data set. It is intrinsically adaptive. It searches for a fixed number of adjacent events, which then define a volume for the metric used. The k-NN classifier has best performance when the boundary that separates signal and background events has irregular features that cannot be easily approximated by parametric learning methods.

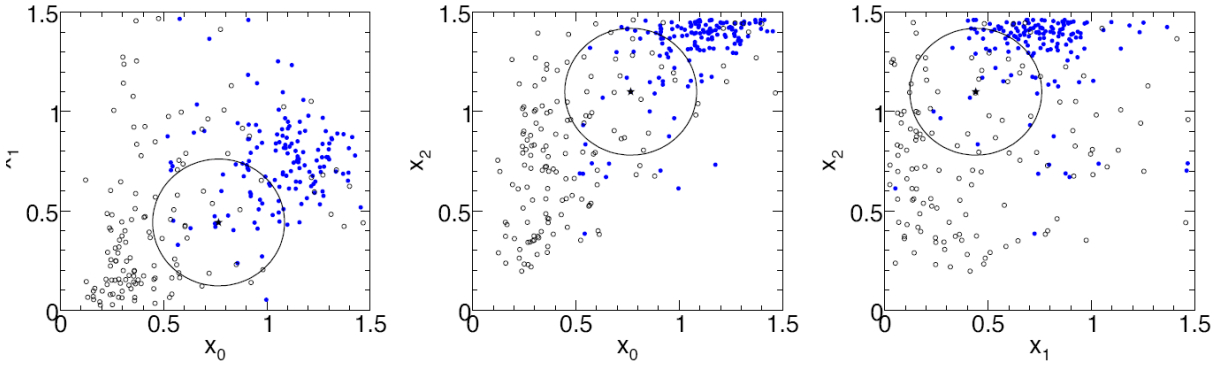


Fig. 3.3 Example for the k-nearest neighbour algorithm in a three-dimensional space (i.e., for three discriminating input variables) [4]. The three plots are projections upon the two-dimensional coordinate planes. The full (open) circles are the signal (background) events. The k-NN algorithm searches for 20 nearest points in the nearest neighborhood (circle) of the query event, shown as a star. The nearest neighborhood counts 13 signal and 7 background points so that query event may be classified as a signal candidate.

3.4.1 Learning

The k-NN algorithm searches for k events that are closest to the test event. Closeness is thereby measured using a metric function. The simplest metric choice is the Euclidean distance

$$R = \left(\sum_{i=1}^{n_{\text{var}}} |x_i - y_i|^2 \right)^{\frac{1}{2}}$$

where n_{var} is the number of input variables used for the classification, x_i are coordinates of an event from a training sample and y_i are variables of an observed test event. The k events with the smallest values of R are the k-nearest neighbours. The value of k determines the size of the neighbourhood for which a probability density function is evaluated. The classification algorithm finds k-nearest training events around a query point

$$k = k_S + k_B$$

where $k_{S(B)}$ is number of the signal (background) events in the training sample. The relative probability that the test event is of signal type is given by

$$P_S = \frac{k_S}{k_S + k_B} = \frac{k_S}{k}$$

The KNN is booked through the TMVA [3] Factory via the command line:

```
factory.BookMethod(dataloader, TMVA.Types.kKNN, "KNN",
"H:nkNN=50:ScaleFrac=0.8:SigmaFact=1.0:Kernel=Gaus:UseKernel=F:\
UseWeight=T:!Trim" );
```

3.5 Others

Various other classification algorithms present in TMVA are trained, tuned for different set of features, hyperparameters, and optimized to solve the classification problem with the feature set mentioned in Chapter 2.

The evaluation of all the tested classification methods with their accuracy are listed in the table in decreasing order.

Classification Method	Testing Accuracy
MulitlayerPerceptron	94.8%
Boosted Decision Trees	94.1%
K-Nearest Neighbour	92.4%
LikelihoodKDE	91.2%
HMatrix	90.0%
CutsSA	89.3%
Cuts	88.7%
BoostedFishers	87.0%
Fishers	86.6%
CutsGA	84.4%
FisherG	84.0%
Linear discriminant	81.6%

Chapter 4

Result

In this project, a benchmarking study was performed between various classification algorithms viz. Artificial Neural Network (MultilayerPerceptron), Boosted Decision Trees with Adaboosting and k-Nearest Neighbour and other such classification algorithms implementation of TMVA with tensorflow as backend.

The codes for the all factory Bookmethods by TMVA for this problem with their outputs are available at `https://git.io/JOEVM`.

4.1 Discussion

The classification methods embedded in TMVA provides a sophisticated tool for the analysis of highly correlated data, with greater flexibility and modularity new features such as deep learning neural networks, cross-validation, hyper parameter tuning and interfaces to python.

Highest testing accuracy of 94.8% was obtained using Multilayer Perceptron, the results are promising and it is envisaged that machine learning can be successfully applied to high energy physics problems which can enhance the capabilities of the Large Hadron Collider at CERN. Machine learning algorithms are already state of the art in many areas of particle physics and will likely be called on to take on a greater role in solving upcoming data analysis and event reconstruction challenges.

As particle physics moves into the post-Higgs boson discovery era, future experiments will require increasingly more powerful identification, reconstruction algorithms, order-of-magnitude increase in compute capacity and incorporating new parallelism constructs from ROOT, to extract rare signals from copious and challenging backgrounds.

4.2 Code

```
1
2 from ROOT import TMVA, TFile, TTree, TCut
3 from subprocess import call
4 from os.path import isfile
5
6
7 # Setup TMVA
8 TMVA.Tools.Instance()
9 TMVA.PyMethodBase.PyInitialize()
10
11 output = TFile.Open('output.root', 'RECREATE')
12 factory = TMVA.Factory('TMVAClassification', output,
13                        '!Silent:Color:DrawProgressBar:Transformations=D,G
    :AnalysisType=Classification')
14
15 # Load data
16 dataS = TFile.Open('beta/data/signal.root')
17 dataB = TFile.Open('beta/data/background.root')
18
19 signal     = dataS.Get('h1')
20 background = dataB.Get('h1')
21
22 dataloader = TMVA.DataLoader('dataset')
23
24
25 dataloader.AddVariable('fit_de')
26 dataloader.AddVariable('fit_mbc')
27 dataloader.AddVariable('thrust')
28 dataloader.AddVariable('bs0mass')
29 dataloader.AddVariable('bl_mass')
30 dataloader.AddVariable('bc_mass')
31 dataloader.AddVariable('blfmass')
32 dataloader.AddVariable('p3bfcm')
33 dataloader.AddVariable('ebcm')
34 dataloader.AddVariable('p1p2dec')
35
36
37 dataloader.AddSignalTree(signal, 1.0)
38 dataloader.AddBackgroundTree(background, 1.0)
39 dataloader.PrepareTrainingAndTestTree(TCut(''),'nTrain_Signal=144000:
    nTrain_Background=900000:SplitMode=Random:NormMode=NumEvents')
40
41 #Model
42
43 # ANN (Multilayer perceptron)
44 0.945
45 factory.BookMethod(dataloader, TMVA.Types.kMLP, "MLP",
46 "H:!V:NeuronType=tanh:VarTransform=N:NCycles=1000:HiddenLayers=N+5:\
47 TestRate=5" );
48
49 0.941
```

```
#Boosted Decision Trees with gradient boosting
factory.BookMethod(dataloader, TMVA.Types.kBDT, "BDTG",
"!H:!V:NTrees=1000:BoostType=Grad:Shrinkage=0.30:UseBaggedGrad:\
GradBaggingFraction=0.6:SeparationType=GiniIndex:nCuts=20:\
PruneMethod=CostComplexity:PruneStrength=50:NNodesMax=5" );

0.924
# k-Nearest Neighbour method (similar to PDE-RS)
factory.BookMethod(dataloader, TMVA.Types.kKNN, "KNN",
"H:nkNN=50:ScaleFrac=0.8:SigmaFact=1.0:Kernel=Gaus:UseKernel=F:\
UseWeight=T:!Trim" );


# Run training, test and evaluation
factory.TrainAllMethods()
factory.TestAllMethods()
factory.EvaluateAllMethods()
```

References

[1] Jelena Filipović, Dimitrije Maletić, Vladimir Udovičić, Radomir Banjanac, Dejan Joković, Mihailo Savić, and Nikola Veselinović. The use of multivariate analysis of the radon variability in the underground laboratory and indoor environment. *Nukleonika*, 61, 01 2016.

[2] Francesco Giuseppe Gravili. Multivariate analysis tutorial, atlas experiment.

[3] A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss, M. Backes, T. Carli, O. Cohen, A. Christov, D. Dannheim, K. Danielowski, S. Henrot-Versille, M. Jachowski, K. Kraszewski, A. Krasznahorkay Jr., M. Kruk, Y. Mahalalel, R. Ospanov, X. Prudent, A. Robert, D. Schouten, F. Tegenfeldt, A. Voigt, K. Voss, M. Wolter, and A. Zemla. Tmva - toolkit for multivariate data analysis, 2007.

[4] A. Hoecker L. Moneta P. Speckmayer J. Stelzer J. Therhaag E. von Toerne H. Voss S. Wunsch K. Albertsson, S. Gleyzer. Toolkit for multivariate data analysis with root. In *TMVA 4*, 2018.

[5] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[6] M Mythili, R Thangarajan, and N Krishnamoorthy. Classification of signal versus background in high-energy physics using deep neural networks. In *International Conference on Emerging Current Trends in Computing and Expert Technology*, pages 1096–1106. Springer, 2019.

[7] ROOT CERN. Analysis of boosted decision trees, 2011.

[8] ROOT CERN. Tmultilayerperceptron class reference, 2011.

[9] HSE University. Addressing large hadron collider challenges by machine learning.

[10] Wikipedia. Multilayer perceptron, 2008.

[11] Hai-Jun Yang, Byron P. Roe, and Ji Zhu. Studies of boosted decision trees for miniboone particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 555(1-2):370–385, Dec 2005.