JOINT INSTITUTE FOR NUCLEAR RESEARCH
Veksler and Baldin laboratory of High Energy Physics

FINAL REPORT ON THE INTEREST PROGRAMME

Artificial Intelligence in Industry-4.0

Supervisor:
Dr. Mihai Dima
Prof. Gheorghe Adam

Student:
Julio Maldonado, México
Universidad Autónoma de Sinaloa (UAS)

Participation period:
May 24 – July 2, Wave 4

Dubna, 2021

Abstract

This report is based on the subjects reviewed during the course *Artificial Intelligence in Industry-4.0*. The course covers some concepts of Industry 4.0, especially the concept of artificial intelligence applied to the industry. The software is used in the ROOT framework (a data analysis software developed by CERN). We reviewed artificial intelligence, machine learning and deep learning concepts. The separate compilation model project setup in a C++ language is implemented as examples of separate compilation model projects. We reviewed the Multilayer Perceptron model (MLP) and the implementation of two different Artificial Neural Networks (ANN) classifiers for RF modulation. The report covers some basic theory of the above subjects and some results using the presented classifiers.

Content

Introduction

On this course we reviewed some background, theory and practical examples of data processing in the context of a practical application in science and industry.

We covered some concepts of Industry 4.0, especially the concept of separate compilation model project setup in a C++ framework. Then we reviewed some artificial intelligence, machine learning and deep learning concepts and the implementation of two different Artificial Neural Networks (ANN) classifiers for RF modulation.

Background

In this part of the course we analyzed concepts of Industry 4.0, C++ language, and data analysis framework ROOT developed by CERN for HEP subjects.

Industry 4.0

The term *Industry 4.0* originated in 2011 at the Hanover Fair in Germany. It came as a result of the Germany initiative to enhance competitiveness in the manufacturing industry. The term refers to the technological evolution from embedded systems to cyber-physical systems (CPS). It can also be referred to as a name for the current trend of automation and data exchange in manufacturing technologies, including cyber-physical systems, the Internet of things, cloud computing and cognitive computing, and creating the smart factory. Some key concepts are described below: [1]

Decentralization: the ability of CPS within Smart Factories to make decisions on their own.

Interoperability: the ability of CPS, humans and Smart Factories to connect and communicate with each other via the Internet of Things and the internet of Services.

Modularity: flexible adaptation of Smart Factories for changing requirements of individual modules.

Real-Time Capability: the capability to collect and analyze data and provide the insights immediately.

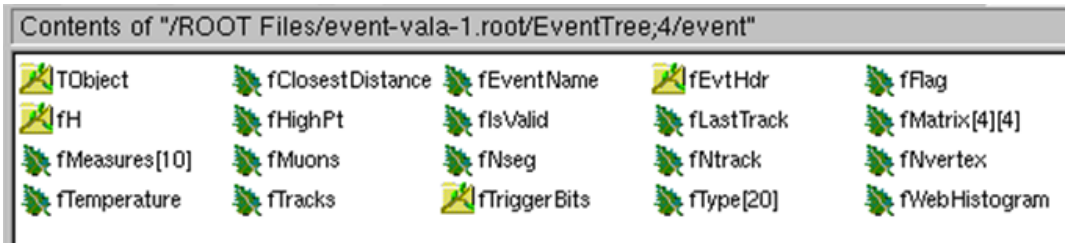
Service Orientation: offering of services (of CPS. humans and Smart Factories) via the Internet of Services.

Virtualization: a virtual copy of the Smart Factory which is created by linking sensor data with virtual plant models and simulation models.

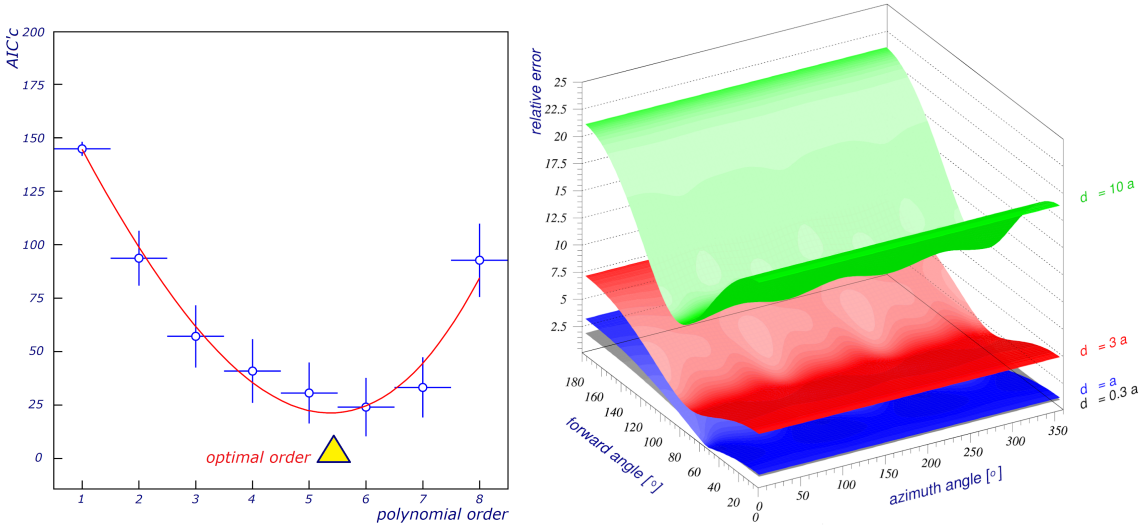
ROOT by CERN

Software framework developed by CERN for data analysis with C++ based language (Python and R also supported). It can do some activities like: [1]

- Store data (up to range of 100 TB, trees/tuples structure).



- Visualize data (2D, 3D plots).

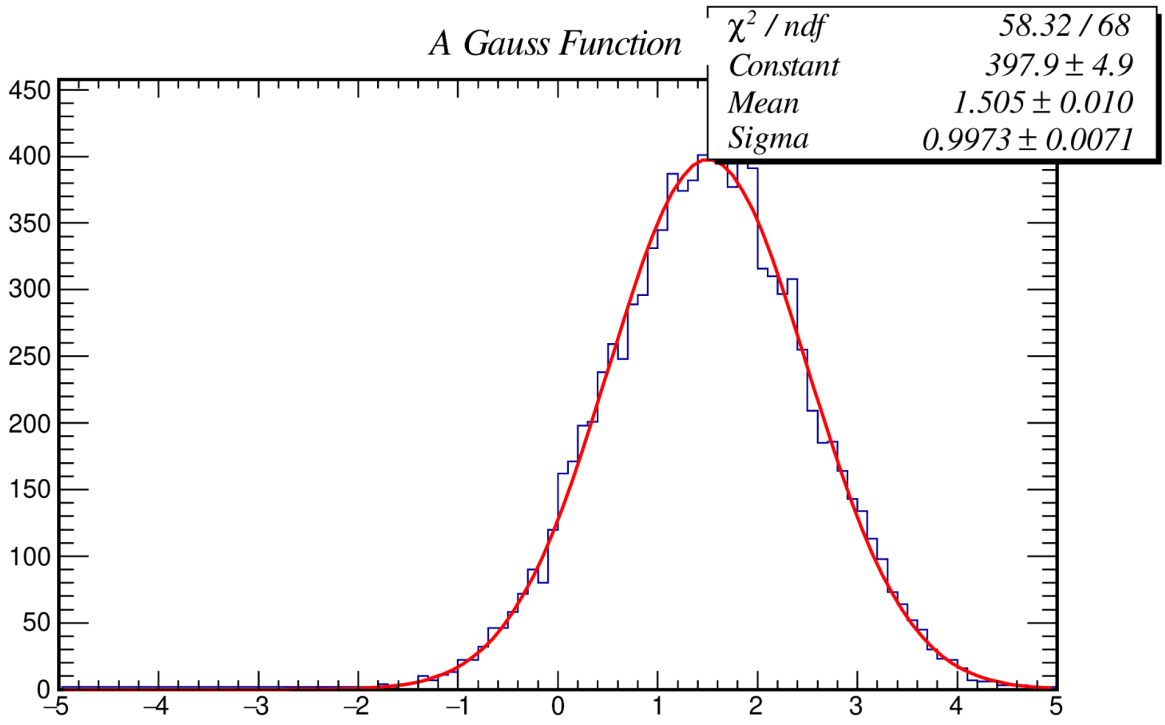


- Data analysis (TMultiLayerPerceptron, TFit Result, and other fit and classification modules).

We used an example of Gauss Function Fit code.

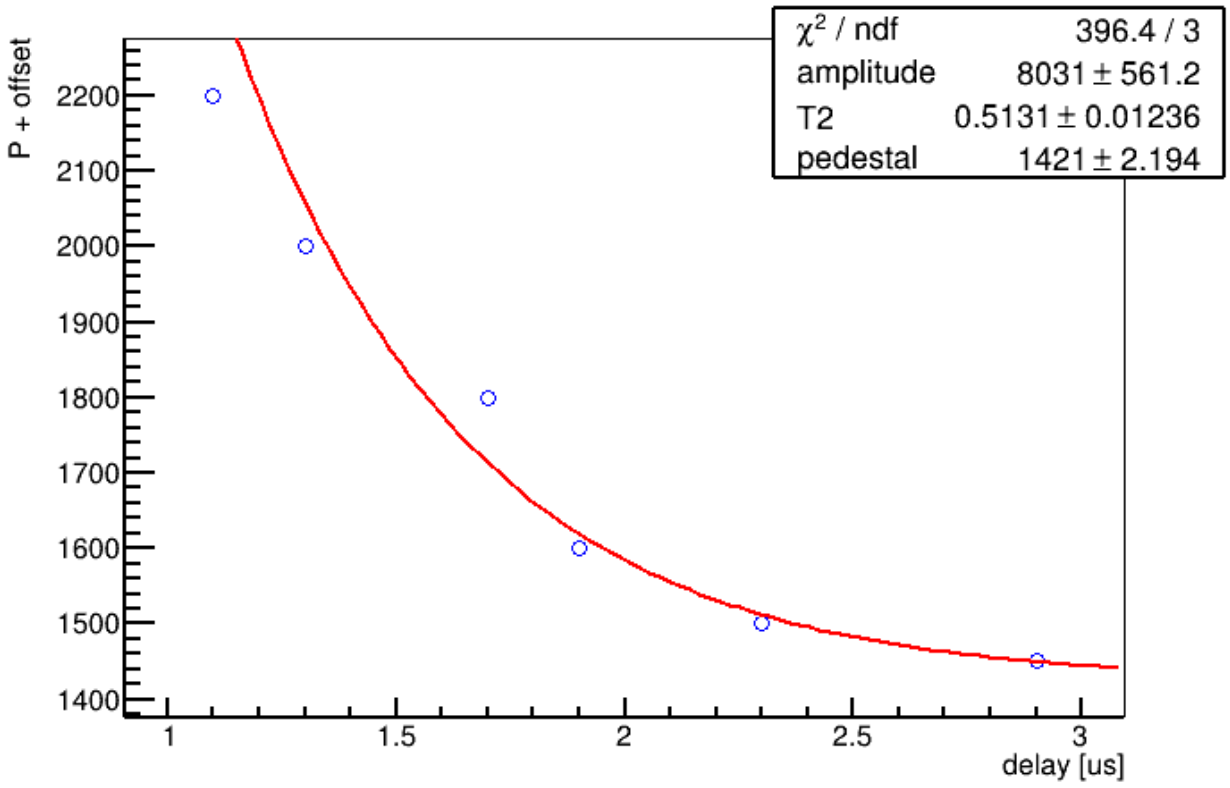
C++ code

```
#include <TH1.h>
#include <TFile.h>
#include <TRandom.h>

int main() {
  TH1F *histo = new TH1F("hgaus", "A Gauss Function", 100, -5.0, 5.0);
  TRandom rnd                                                        ;
  for (int i = 0; i < 10000; ++i) {double x = rnd.Gaus(1.5, 1.0)     ;
    histo->Fill (x)                                                  ;}
  TFile outfile ("gaus.root", "RECREATE")                            ;
  histo->Write()                                                     ;
  outfile.Close()                                                    ;
  return 0                                                           ;}
```



- Run your own C++ codes.

```
// ____________________ ROOT FITS ____________________________

void myfit() {

// TGraph gr  ("data.txt", "%lg %lg");
// TGraph grr ("test.txt", "%lg %*lg %lg")
// TGraph grrr("test.txt", "%lg %*lg %*lg %lg")

gStyle->SetOptFit   (1)                          ;
gStyle->SetLineWidth(2)                          ;

TGraphErrors* gr = new TGraphErrors("dx.txt")    ;

Int_t N = gr->GetN()                             ;
Double_t x,y                                     ;
  for (Int_t i=0; i<N; i++) {
    gr->GetPoint     (i,      x,      y)         ;
    gr->SetPointError(i,   0.01,   0.01)         ;}

TF1 fit("fit", "([0]*exp(-x/[1])+[2])", 12, 462) ;

    fit.SetParName  (0, "amplitude" )            ;
    fit.SetParName  (1, "T2"        )            ;
    fit.SetParName  (2, "pedestal"  )            ;

    fit.SetParameter(0,  800.0      )            ;
    fit.SetParameter(1,    1.6      )            ;
    fit.SetParameter(2, 1350.0      )            ;

    gr->Fit("fit")                               ;

    gr->SetMarkerColor(  4                  )    ;
    gr->SetMarkerSize (  1                  )    ;
    gr->SetMarkerStyle( 24                  )    ;

  //gr->SetTitle      ("Dynamic decoupling" )    ;
    gr->SetTitle                (""         )    ;

  //gr->GetYaxis()->CenterTitle(true        )    ;
    gr->GetYaxis()->SetTitle   ("P + offset")    ;
  //gr->GetXaxis()->CenterTitle(true        )    ;
    gr->GetXaxis()->SetTitle   ("delay [us]")    ;

    gr->Draw("AP")                               ;
  //gr->Draw("ALP")                              ;

  }
```



Project Goals

- Learn how to use a separate-compilation model framework that allows us to integrate various contributions into one project.
- Learn a few of the advanced C++ features.
- Learn the theory of Artificial Intelligence, Machine Learning, and the MLP perceptron.
- Learn how to train and use a neural network.
- Implement RF Modulation ANN classifiers

Scope of Work

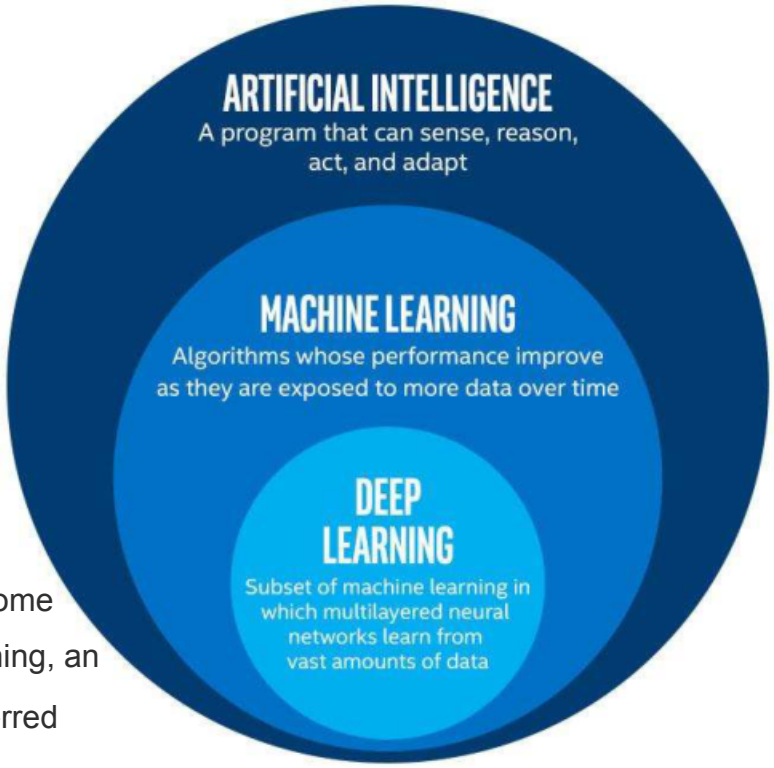
Artificial Intelligence

Artificial Intelligence is a field of study that attempts to understand and build intelligent entities. A rational approach is concerned with computational intelligence: the study of the design of intelligent agents. [2]

An agent is just something that acts. Computer agents are expected to operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals. A rational agent is one that acts so as to achieve the best expected outcome. [2]

Figure 1 shows the relation between artificial intelligence, machine learning and deep learning. [1]

Machine Learning



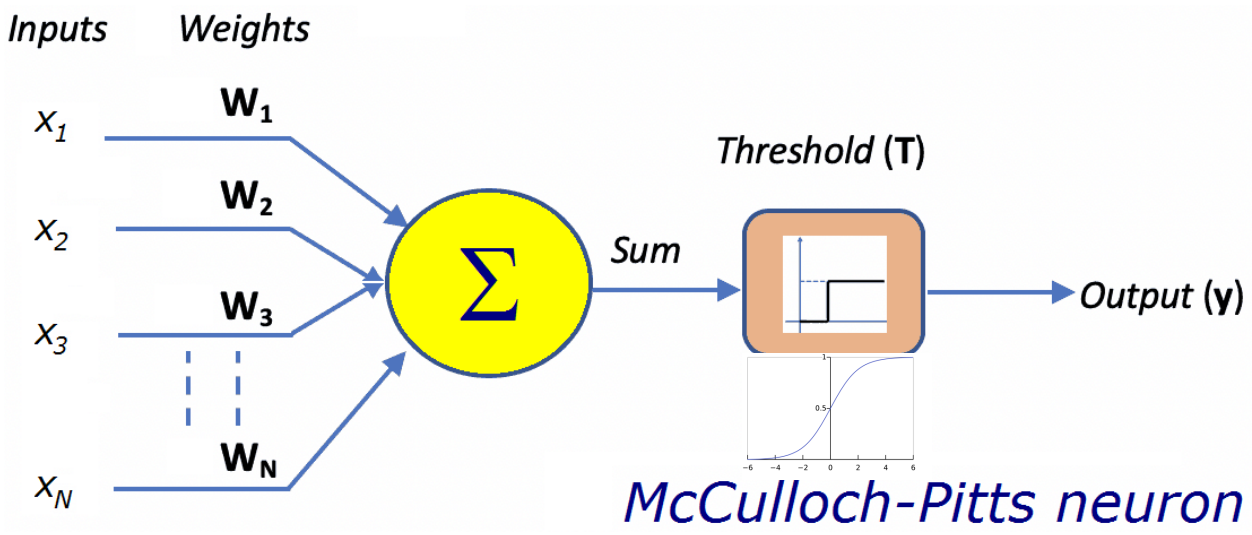
Machine learning is the paradigm for automated learning from data, using computer algorithms.
The goal of learning is to respond correctly to future data [3].

Statistical techniques use mathematical model $f(x)$ and find parameters of the model either analytically or numerically by using some optimization criteria. In machine learning, an approximating function $f(x, \omega)$ is inferred

automatically from the given data without requiring a priori information about the function. [3]

Supervised learning is the most powerful approach to obtain the approximation function using a training data set with feature vectors (inputs) and the corresponding targets (desired outputs). [3]

The training data set $\{y, x\}$, where y are the targets from the true function $f(x)$, encodes information about the input-output relationship to be learned.



The Figure [1] above shows a neuron model (node). This node receives the information as feature variables and each one of those has an interconnection characterized by a weight whose values are learned during the training phase. Those values the parameters of the model.

This node processes the information it receives with an activation or transformation function and may have a bias or a threshold. The activation function is generally a nonlinear function that allows for flexible modeling.

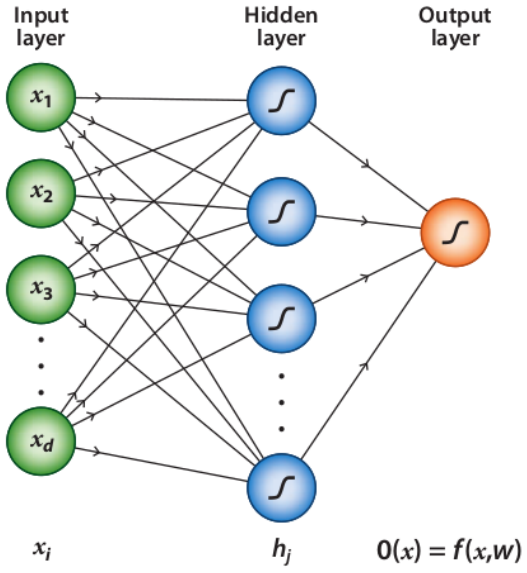
Neural Networks (NNs) are composed of several neurons arrayed on multiple layers. [3]

Deep Learning

Subset of machine learning in which multilayered Neural Networks learn from vast amounts of data. [1]

Multilayer Neural Networks consist of an interconnected group of neurons arranged in layers; each neuron processes the information it receives with an activation function, then passes the result to the next layer of nodes. [3]

The first layer, known as the input layer, receives the feature variables, then it is followed by one or more hidden layers of neurons. The last layer outputs the final response of the network. NNs with one hidden layer are sufficient to model the posterior probability to arbitrary accuracy. [3]



Defining a data set with d feature variables, $x = \{x_1, x_2, ...x_d\}$, the output of the network is,

$$O(\mathbf{x}) = f(\mathbf{x}.\omega) = g\left(\theta + \sum_j \omega_j b_j\right) = p(s|\mathbf{x})$$

where b_j is the output from the hidden nodes:

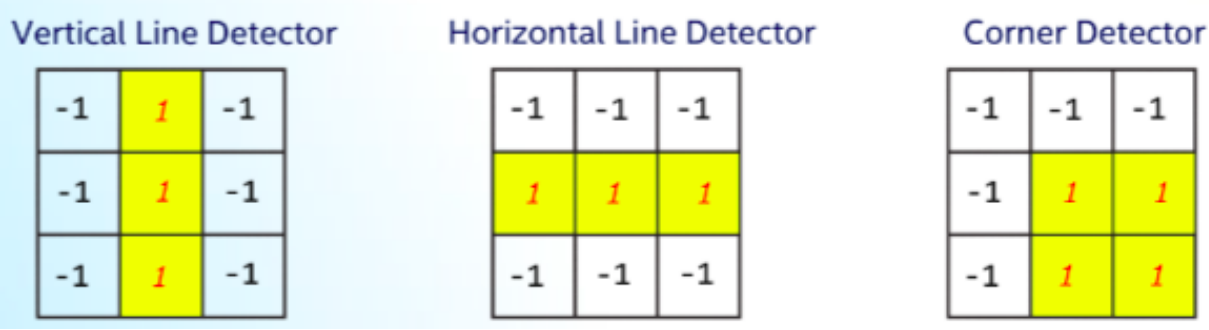
$$b_j = g\left(\theta_j + \sum_i \omega_{ij} x_i\right)$$

The nonlinear activation function g is commonly taken as a sigmoid,

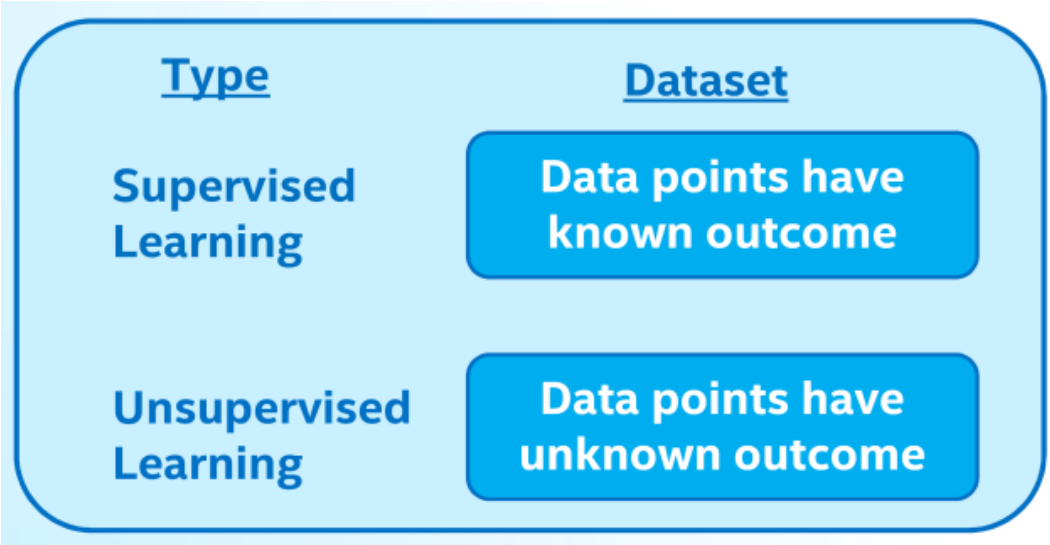
$$g(a) = \frac{1}{1 + e^{-a}}$$

Key concepts

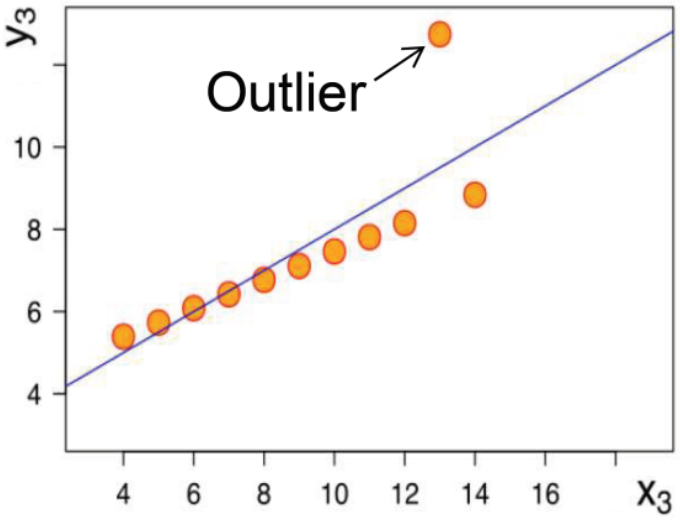
Convolutions: can be thought of as “local feature detectors”. [1]

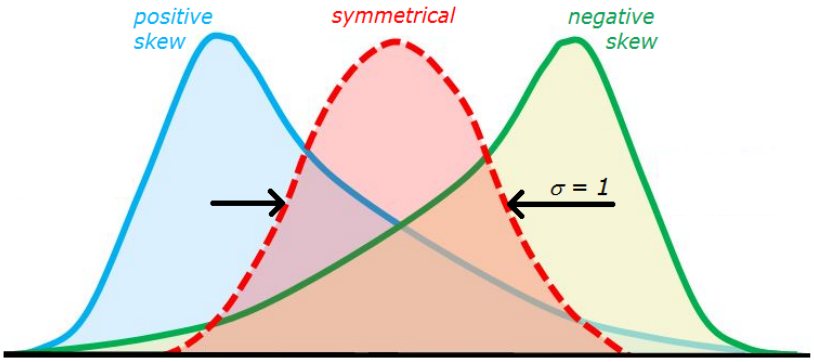


Training methods: training is the step of data analysis when we define the parameters of our models and learn from our data. The learning process can be supervised or unsupervised.



Data conditioning: it is important to remove outliers, uniformise data and norm data to $\sigma = 1$





Data splitting: split data into two sets. Training set data used during the training process. Test set data used to measure performance, simulating unseen data.

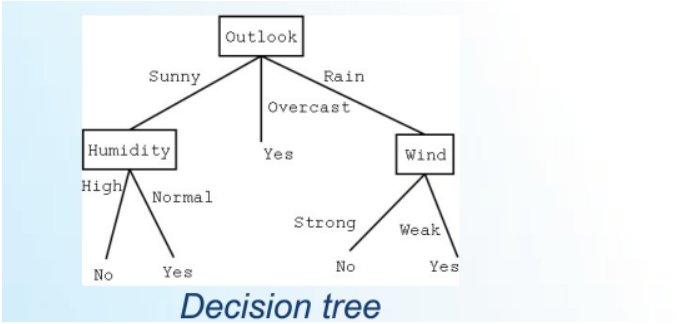
sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
		5.6	2.1	virginica
		1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4		
5.4	3.4	1.7		

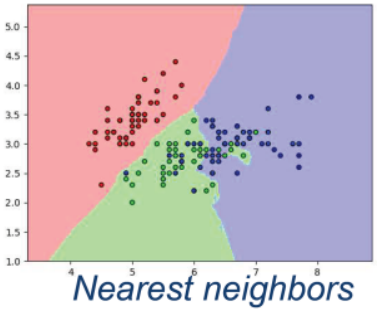
Training Set

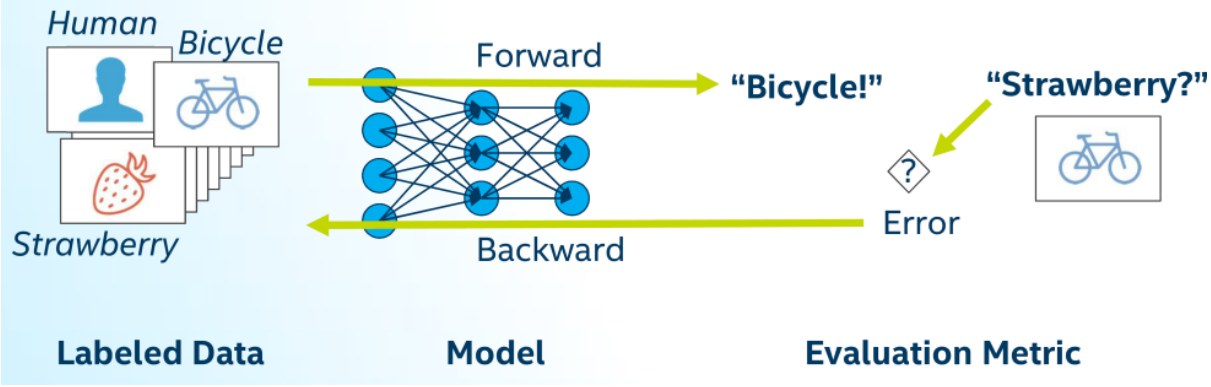
Testing Set

Training step (supervised learning problem): [1]

- Collect a labeled dataset (features vectors and target output labels).
- Choose the model.



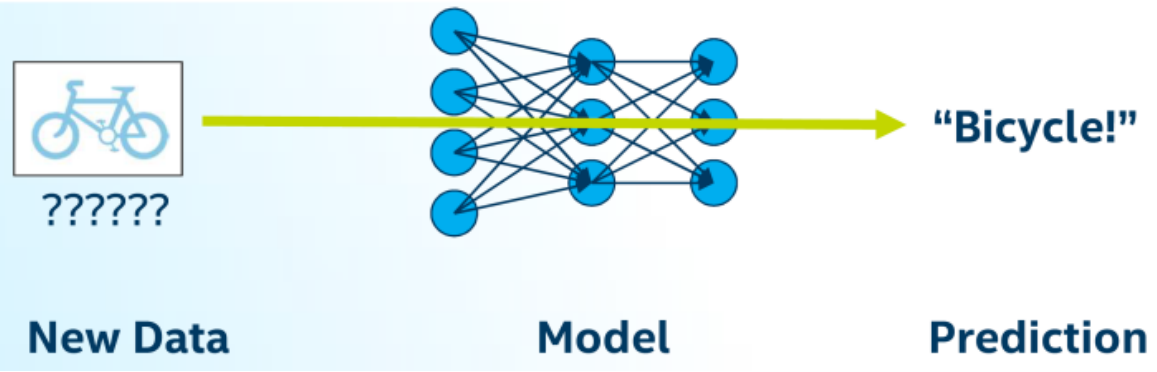




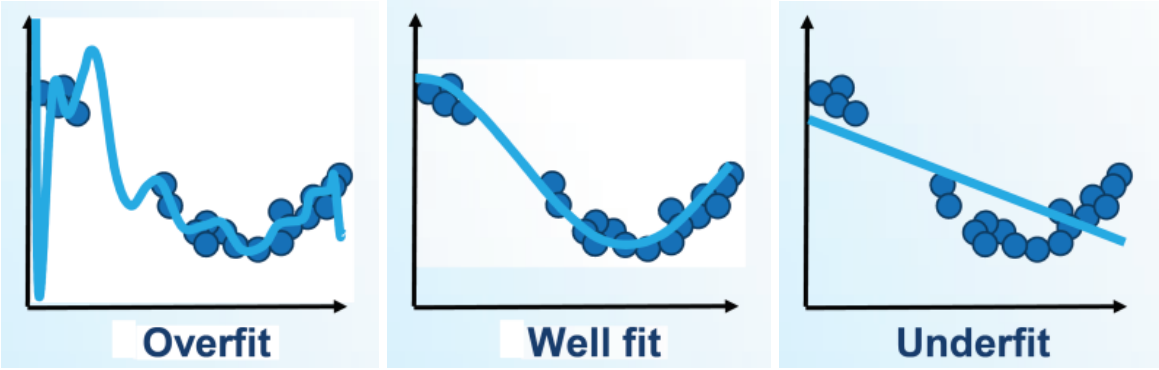
Test step (supervised learning problem): [1]

- Choose an evaluation metric based on accuracy (how well predictions match true values), and Mean Squared Error (average square distance between prediction and true value).

- Choose an optimization method. Model configuration that gives the best performance.
- Inference. Once the model is trained, we can provide new examples for predictions.



- Pruning. How well the model fits the data.

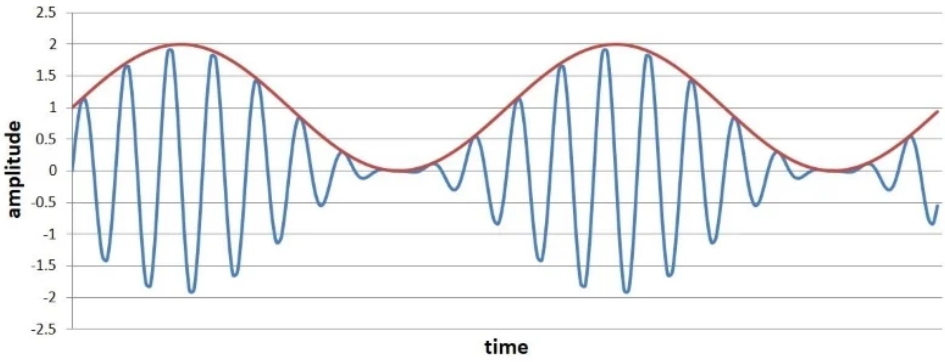


Methods

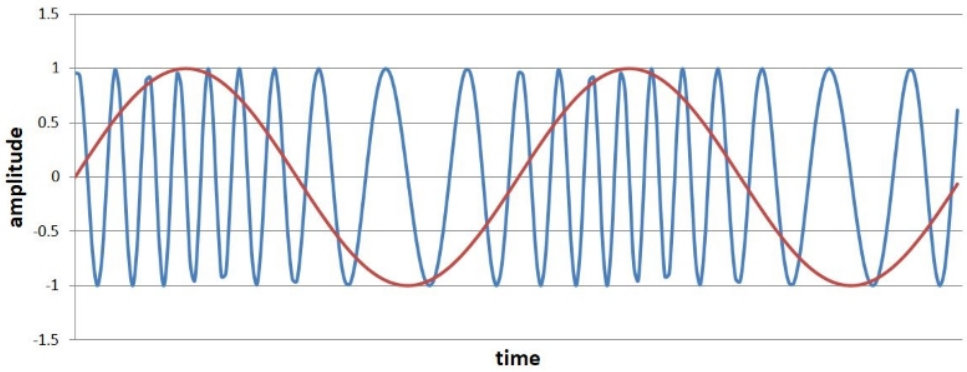
We implement a separate-compilation model framework for classification of Radio Frequency (RF) modulation. Then we learn the theory and implementation of a MLP perceptron model (training and using a neural network) for RF Modulation.

RF modulation types [1] [4] [5]

Amplitude modulation (AM) is a modulation technique used in electronic communication, most commonly for transmitting messages with a radio wave. In amplitude modulation, the amplitude (signal strength) of the carrier wave is varied in proportion to that of the message signal, such as an audio signal.



Frequency modulation (FM) is the encoding of information in a carrier wave by varying the instantaneous frequency of the wave. The technology is used in telecommunications, radio broadcasting, signal processing, and computing.



Phase modulation (PM) is a modulation pattern for conditioning communication signals for transmission. It encodes a message signal as variations in the instantaneous phase of a carrier wave.

Angle modulation is a class of carrier modulation that is used in telecommunications transmission systems. The class comprises frequency modulation (FM) and phase modulation (PM), and is based on altering the frequency or the phase, respectively, of a carrier signal to encode the message signal.

Those are examples of analog modulation. In digital modulation, an analog carrier signal is modulated by a discrete signal. Digital modulation methods can be considered as digital-to-analog conversion and the corresponding demodulation or detection as analog-to-digital conversion. The changes in the carrier signal are chosen from a finite number of M alternative symbols (the modulation alphabet).

The most fundamental digital modulation techniques are based on keying:

PSK (phase-shift keying): a finite number of phases are used.

FSK (frequency-shift keying): a finite number of frequencies are used. The carrier signal is periodically shifted between two frequencies that represent the two binary digits.

ASK (amplitude-shift keying): a finite number of amplitudes are used.

QAM (quadrature amplitude modulation): a finite number of at least two phases and at least two amplitudes are used.

RF parameter determination [1]

Defining the RF wave

$$y = p + Asin(2\pi ft + \phi)$$

Pedestal: find for average

$$\langle\, y \,\rangle = p + A_e sin\left(2\pi ft\frac{t_i + t_f}{2} + \phi\right) sinc\left(\frac{2\pi f \Delta t}{2}\right)$$

$$A_e = \frac{A}{sinc(\pi f \Delta)}$$

Amplitude: find for second derivation

$$\langle\, \delta^2 y \,\rangle = A_e \langle\, \delta^2 (sin) \,\rangle$$

Frequency: find for deviation

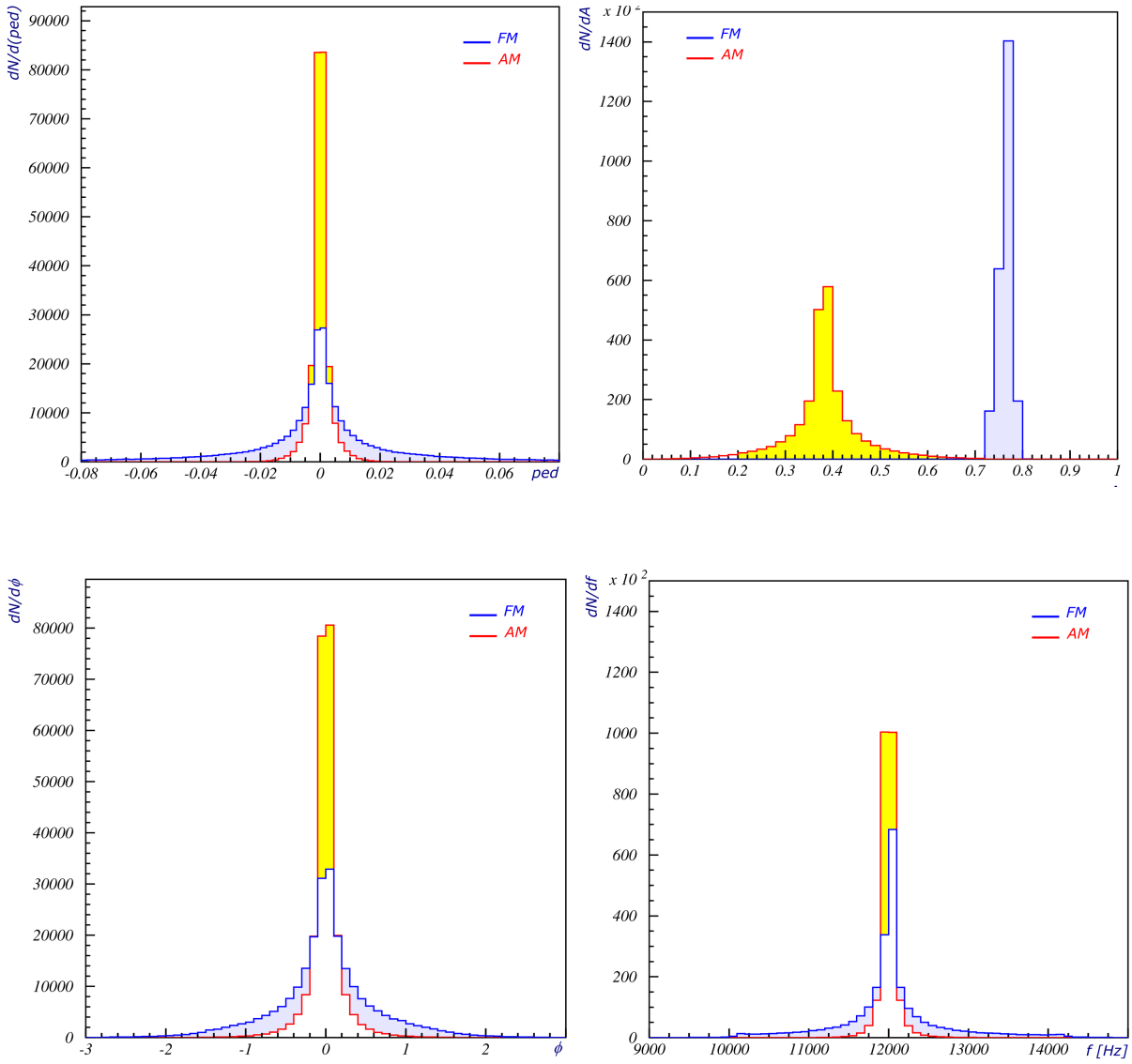
$$\begin{aligned}\langle\, y(y - y_{k\Delta}) \,\rangle = pA_e(\langle\, sin \,\rangle - \langle\, sin_{k\Delta} \,\rangle) \\ + \\ A_e^2(\langle\, sin^2 \,\rangle - \langle\, sin \cdot sin_{k\Delta} \,\rangle)\end{aligned}$$

$$\simeq \pi k \Delta A_e^2 sin(2\pi f k \Delta) \cdot \delta f$$

Phase: find for angle derivation

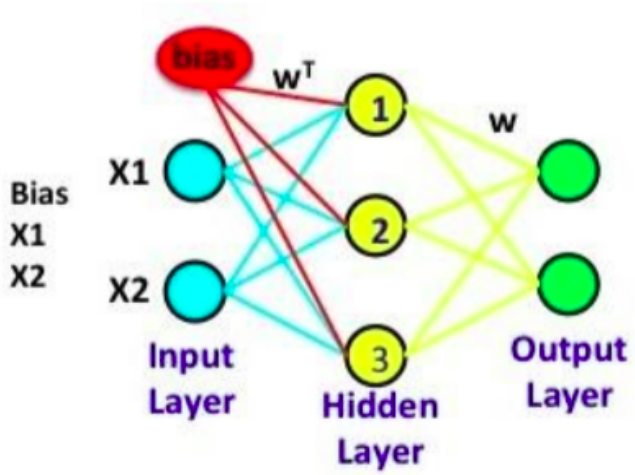
$$\langle\, y \cdot cos(\pi ft) \,\rangle \simeq \frac{A_e}{2} sin\phi$$

The Figure below shows the distribution for different modulations obtaining best results for frequency modulation distribution. [1]

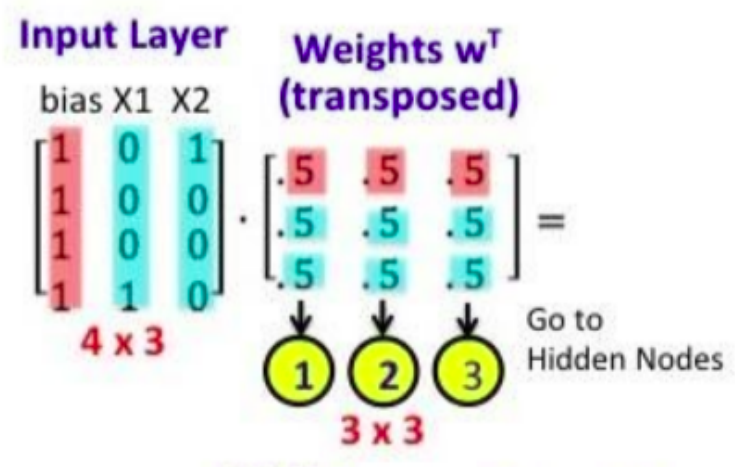


MLP Perceptron

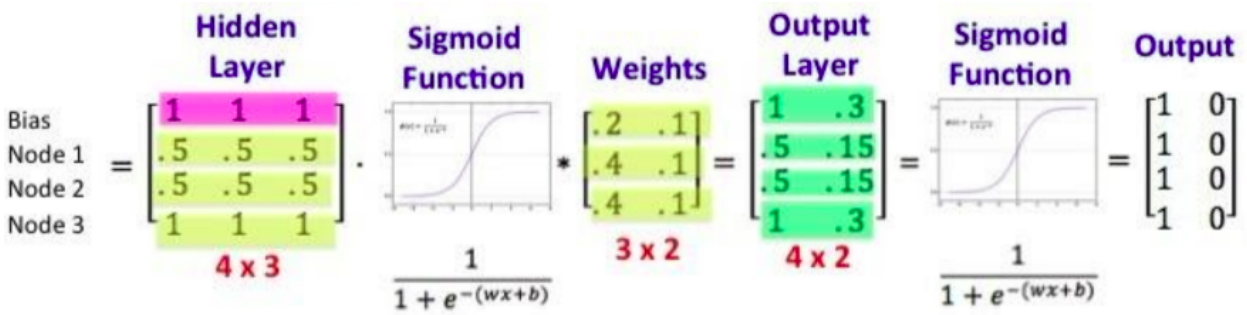
We implement the MLP perceptron model represent in the Figure below, [1]



The model is determine for an input layer (x_1, x_2 and bias values), and default weights w^T for every connection and every element (data).



This model has a single hidden layer with three nodes. There is a connection of every element (four in total) with each node (Node 1, 2, and 3, and Bias). The output of every node, determine for the product of input values and weights is transformed with the Sigmoid Function and then multiplied with the second weights.



The output layer has two values for each element, one for every input variable. Using the Sigmoid Function one more time, we transform the output in a way related with original data values suitable for classification.

TMultiLayerPerceptron

We used the TMultiLayerPerceptron class of ROOT for implementing.

Public Member Functions

TMultiLayerPerceptron ()
Default constructor. More...

TMultiLayerPerceptron (const char *layout, const char *weight, **TTree** *data, **TEventList** *training, **TEventList** *test, **TNeuron::ENeuronType type=TNeuron::kSigmoid**, const char *extF="", const char *extD="")
The network is described by a simple string: The input/output layers are defined by giving the branch names separated by comas. More...

Example: radial field of a magnet

This the the code using C++ creating a macro run with ROOT:

```
// read data ____________________________________________________

TTree* t = new TTree("treename", "description")              ;

    // (r,z) = cylindrical coordinates
    // Br    = radial component of magnetic field

Int_t nlines = t->ReadFile("Br.dat","r:z:Br")                ;
```

First, we read data from a tree structure. The data contains cylindrical coordinates values for magnetic fields (r, z).

```
// MLP setup ___________________________________________________

TMultiLayerPerceptron *mlp =
  new TMultiLayerPerceptron("@r,@z:10:10:10:@Br",
                            t                   ,
                           "Entry$%2"           ,
                           "(Entry$+1)%2"       )         ;

    // i/p           = r, z (both normed: @)
    // mid-layers    = 10+10+10 neurons
    // o/p           = Br (normed: @)
    //
    // training set = even,  Entry$%2     = true
    // testing  set = odd , (Entry$+1)%2 = true
```

Then we set up MLP using three hidden layers, every one of them with ten different neurons. The output values are the magnetic strength component, B_r. We also divide the set between training set and testing set.

```
// set learn method ____________________________________________

mlp->SetLearningMethod(TMultiLayerPerceptron::kBFGS )      ;

    // kStochastic = default
    // kBatch
    // kSteepestDescent
    // kRibierePolak
    // kFletcherReeves
    // kBFGS
```

Then the learning method is set, the Broyden, Fletcher, Goldfarb, and Shanno local search optimization algorithm (BFGS).

The training set is filled with tree information.

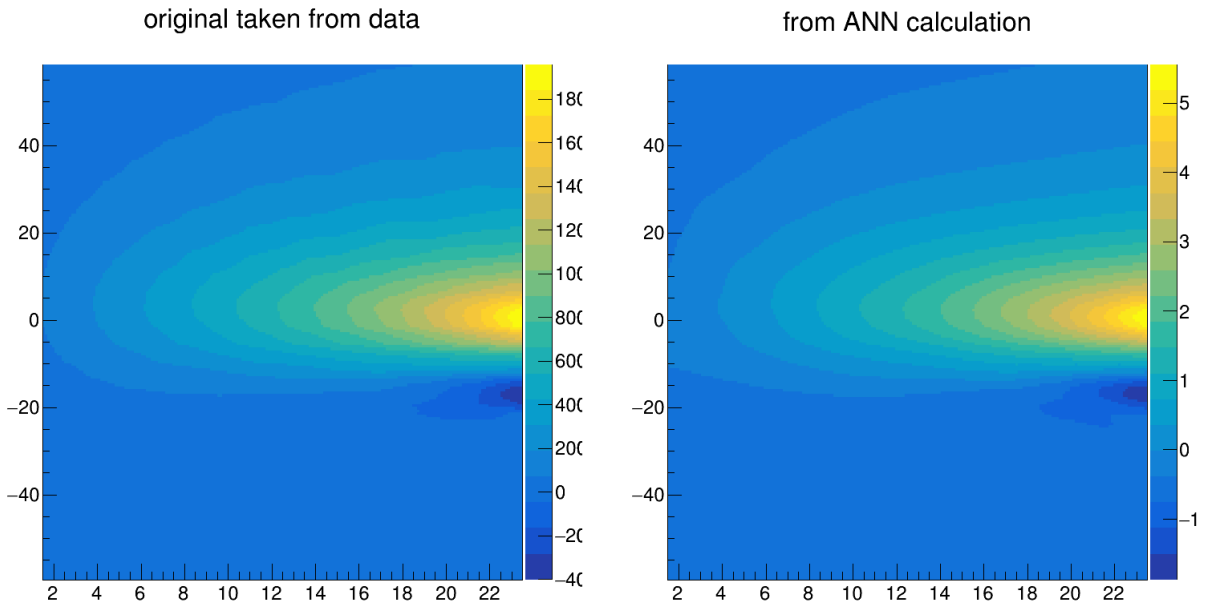
```
// training ____________________________________________________________

mlp->Train( 1000
           "text,update=100" )                                          ;

    //  1000 events
    //  write text to console
    //  updates every 100 epochs


// fill Br(r, z) ______________________________________________________

Int_t nEvent = t->GetEntries()                                          ;
Double_t* r       = new Double_t [ nEvent ]                             ;
Double_t* z       = new Double_t [ nEvent ]                             ;
Double_t* Br      = new Double_t [ nEvent ]                             ;
Double_t* Br_cal  = new Double_t [ nEvent ]                             ;

Float_t rr ,
        zz ,
        Brr                                                             ;
t->SetBranchAddress("r" , &rr)                                          ;
t->SetBranchAddress("z" , &zz)                                          ;
t->SetBranchAddress("Br", &Brr)                                         ;

for (Int_t i = 0; i <  t->GetEntries(); i++) {
  t->GetEntry(i)                                                        ;
  r [i] = rr                                                            ;
  z [i] = zz                                                            ;
  Br[i] = Brr                                                           ;}
```

Input 0, 1, and output are set, and then mlp function is evaluated for each event (data).

```
// using the MLP  _____________________________________________________

Double_t inputs[2];
for (Int_t i=0; i<nEvent; i++ ){
  inputs[0] = r[i]                                                      ;
  inputs[1] = z[i]                                                      ;
  Br_cal[i] = mlp->Evaluate(0, inputs )                                 ;}
```

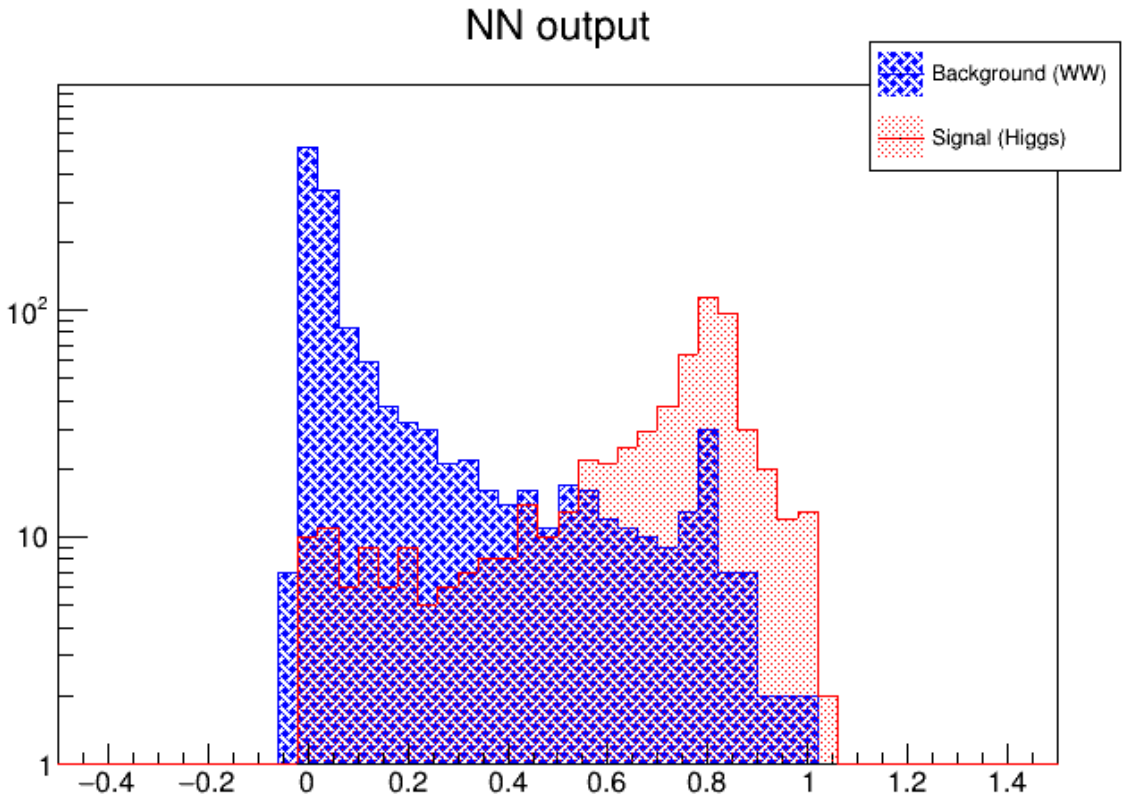
Finally, the canvas and plots are defined to obtain images from original data and MLP perceptron model.

```
TCanvas* c1 = new TCanvas("c1", "c1", 500, 400)              ;
c1->Divide(2,1)                                              ;
c1->cd(1)                                                    ;

TGraph2D* gr1 = new TGraph2D(nEvent, r, z, Br)               ;
gr1->SetTitle("original taken from data")                    ;
gr1->SetNpx(200)                                             ;
gr1->SetNpy(200)                                             ;
gr1->Draw("colz")                                            ;

c1->cd(2)                                                    ;
TGraph2D* gr2 = new TGraph2D("graphname"   ,
                             "description" ,
                              nEvent       ,
                              r            ,
                              z            ,
                              Br_cal       )                 ;
gr2->SetTitle("from ANN calculation")                        ;
gr2->SetNpx(200)                                             ;
gr2->SetNpy(200)                                             ;
gr2->Draw("colz")                                            ;
```



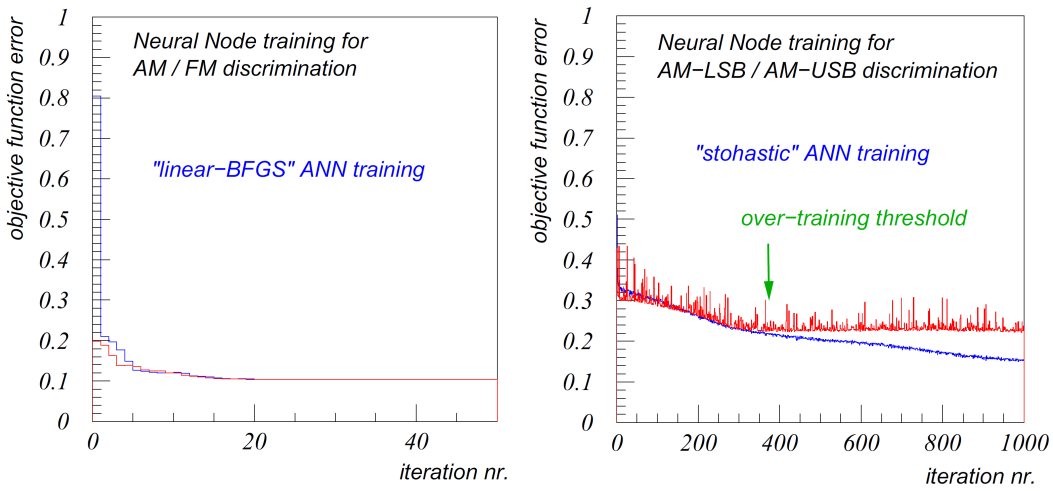
RF Modulation ANN classifier [1]

Using parameters determined with RF modulation types analysis, and using the MLP model for classification, we can classify AM vs. FM modulation, and AM-LSB vs. AM-USB modulation. The classification template was used for Higgs signal vs background classification as is shown in the Figure below,

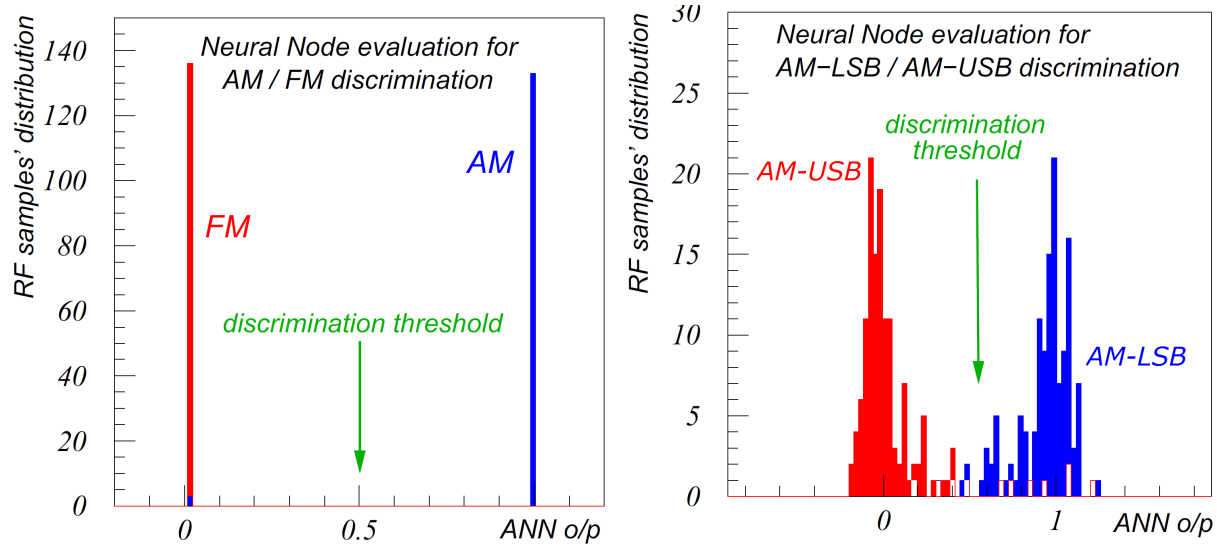


Results

Making several runs until 20% short of overtraining,



The result could be obtained for both cases using the different sample data,



Conclusion

We obtained good results for pattern recognition in the case of the magnetic fields data, and we also obtained good results for the Higgs signal vs background classification as well as for AM vs FM modulation, and for AM-LSB vs AM-USB modulation classification.

I learned some advanced aspects of C++ programming language. We also installed and used the ROOT framework for data analysis working locally and using a cluster connection.

I learned the theory and application of the Mult-Layer Perceptron model, which demonstrated to be very powerful for pattern recognition and classification. Their application in science and in a vast amount of industry sectors is undoubtedly important nowadays. Those machine and deep learning methods and other aspects of the Artificial Intelligence framework are the core of the “revolution” Industry 4.0.

References

[1] M. Dima and Gh. Adam. 2021. Artificial Intelligence in Industry-4.0 Course. Laboratory of Information Technologies, JINR. Russia.

[2] S. Russell and P. Norvig. 2009. Artificial Intelligence: A Modern Approach (3rd. ed.). Prentice Hall Press, USA.

[3] P. C. Bhat. 2011. Multivariate Analysis Methods in Particle Physics. Annual Review of Nuclear and Particle Science, Volume 61, Pages 281-309.

[4] D. Brandon. 2010. Multichannel DDS Enables Phase-Coherent FSK Modulation. Analog Dialogue, Volume 44, Number 4.

[5] B. Gilbert. 2008. Considering Multipliers (Part 1). Analog Dialogue, Volume 42, Number 4.