TMVA

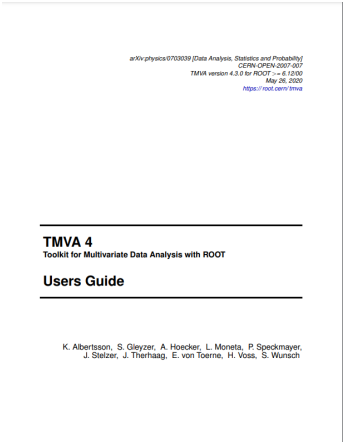
ROOT Seminar

Yukai Zhao

January 10, 2024

Content

- Introduction
- Data preprocessing
- TMVA structure
  - Decision Trees
  - Neural Networtk
  - Convolutional Neural Network



TMVA User Guide

Introduction

Definition:
Multivariate Data Analysis (MVA) is a statistical and computational approach that involves the simultaneous analysis of multiple variables to understand complex relationships and patterns within a dataset.

TMVA = Toolkit for Multivariate Analysis
→ Important part of the ROOT data analysis framework.

Example:
Goal: Differentiate $B_s^0 \to \phi\gamma(\to e^+e^-)$ from $B_s^0 \to \phi J/\psi(\to e^+e^-)$
What we obtain from experiment are parameters such as ($p_T, E, m,$ flight distance, ...)

Data preprocessing

Decorrelation

Correlation makes it harder to learn the underlying structure

$\rightarrow$ Utilizing covariance matrix $x' \mapsto (C)^{-1} x$

Data preprocessing

Primarily used for machine learning applications

Preprocessing to $O(1)$ numbers, e.g. $\frac{x-\mu}{x_{\max}-x_{\min}}$

$\rightarrow$ Improved Convergence and Better Interpretability

Data seperation

Dataset split into $\sim$ 80% training data, $\sim$ 20% validation data

Using validation data to test for overfitting

(Boosted) Decision Tree

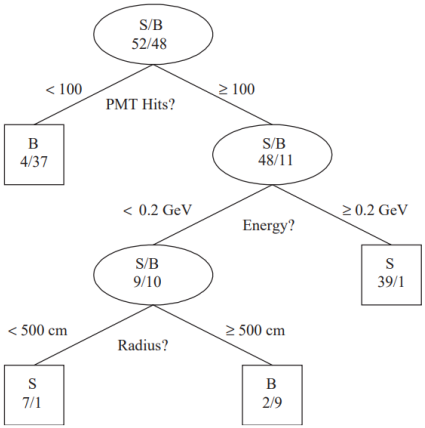
Supervised learning: Have events and know if it's signal and background

Example: Selection of ν_e from a beam of ν_μ using the MiniBooNE Cerenkov detector

signal efficiency, or true positive rate
$\epsilon_S \equiv s^{\text{S-tagged}}/s^{\text{truth}}$
background mis-id rate, or false positive rate
$\epsilon_B \equiv b^{\text{S-tagged}}/b^{\text{truth}}$

$$\text{purity} = \text{precision} = \frac{s^{\text{S-tagged}}}{s^{\text{S-tagged}} + b^{\text{S-tagged}}} \quad (1)$$

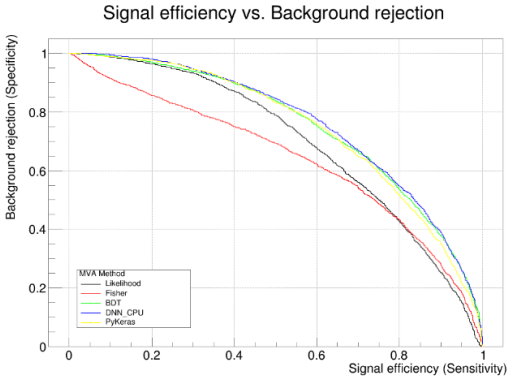
$$\text{accuracy} = \frac{s^{\text{S-tagged}} + b^{\text{B-tagged}}}{s^{\text{truth}} + b^{\text{truth}}} \quad (2)$$



arXiv:physics/0408124

Receiver Operating Characteristic (ROC) curve

ROC curve graphical representation of the trade-off between signal efficiency and background rejection



```
void TMVA_Higgs_Classification() {

}
    auto outputFile = TFile::Open("Higgs_ClassificationOutput.root", "RECREATE");
// Declare Factory. Main object for TMVA. All functions called later are from this class
    TMVA::Factory factory("TMVA_Higgs_Classification", outputFile,
                          "!V:ROC:!Silent:Color:AnalysisType=Classification" );

/**
Set up of data
**/

    TString inputFileName = "Higgs_data.root";
    inputFile = TFile::Open( inputFileName );

    TTree *signalTree     = (TTree*)inputFile->Get("sig_tree");
    TTree *backgroundTree = (TTree*)inputFile->Get("bkg_tree");

    signalTree->Print();

/***
## Declare DataLoader(s)

Define the input variables that shall be used for the MVA training
note that you may also use variable expressions, which can be parsed by TTree::Draw( "expression" )]
***/

    TMVA::DataLoader * loader = new TMVA::DataLoader("dataset");

    loader->AddVariable("m_jj");
    loader->AddVariable("m_jjj");
    loader->AddVariable("m_lv");
    loader->AddVariable("m_jlv");
    loader->AddVariable("m_bb");
```

```
/// We set now the input data trees in the TMVA DataLoader class
// You can add an arbitrary number of signal or background trees
    loader->AddSignalTree    ( signalTree,     signalWeight     );
    loader->AddBackgroundTree( backgroundTree, backgroundWeight );

///Spliting and preparing data
    loader->PrepareTrainingAndTestTree( mycuts, mycutb,
        "nTrain_Signal=7000:nTrain_Background=7000:SplitMode=Random:NormMode=NumEvents:!V" );
    //7000 training events for signal and background, random slected, normalization by weights

//Boosted Decision Trees
if (useBDT) {
    factory.BookMethod(loader,TMVA::Types::kBDT, "BDT",
                       "!V:NTrees=200:MinNodeSize=2.5%:MaxDepth=2:BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction=0.5:SeparationType=GiniIndex:nCuts=20" );
}

    factory.TrainAllMethods();


    factory.TestAllMethods();

    factory.EvaluateAllMethods();


    auto c1 = factory.GetROCCurve(loader);
    c1->Draw();

    outputFile->Close();
}
```

$\Rightarrow$ obtain weights file

```
TMVA::Tools::Instance();

TMVA::Reader *reader = new TMVA::Reader("!Color:!Silent");

float m_jj, m_jjj;
reader->AddVariable("m_jj", &m_jj);
reader->AddVariable("m_jjj", &m_jjj);

reader->BookMVA("cut", "dataset/weights/TMVAClassification_CutsSA.weights.xml");


TFile *input = TFile::Open("root.root");
TTree *theTree = (TTree *) input->Get("tree");
double_t userVar1, userVar2;
theTree->SetBranchAddress("m_jj", &userVar1);
theTree->SetBranchAddress("m_jjj", &userVar2);

for (Long64_t ievt = 0; ievt < theTree->GetEntries(); ievt++) {
    theTree->GetEntry(ievt);
    m_jj = userVar1;
    m_jjj = userVar2;
    Bool_t passed = reader->EvaluateMVA("cut", .99);
    if (passed) { // do something
    }
}
```

Neural Networks

Neural Networks can be seen as a fit function with huge number of model parameters θ

$$f_\theta(x) \sim f(x) \tag{3}$$

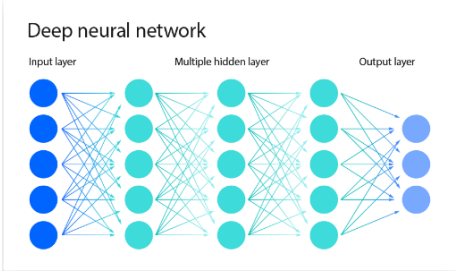
Using many **layers** to perform the "fit":

$$x \to x^{(1)} \to x^{(2)} \to ... \to x^{(n)} \equiv f_\theta(x) \tag{4}$$

where the layers are defined as:

$$x^{(n-1)} \to x^{(n)} := W^{(n)} x^{(n-1)} + b^n \tag{5}$$

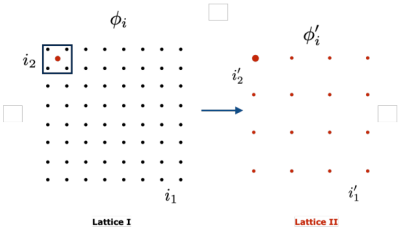
Neural network learns network weights W and bias b through **Loss function** $\mathcal{L} = |x_{\text{pred}} - x_{\text{true}}|$



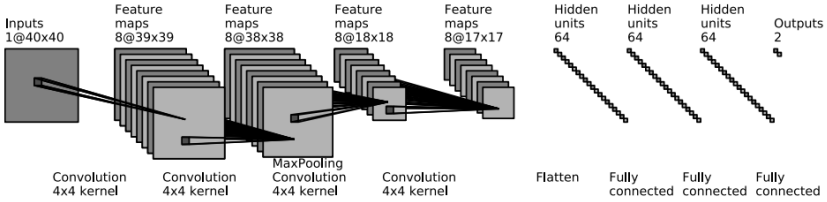
Deep neural network

CNN

CNN primarily used for image classification:
Zero padding, Convolution, Feature maps, Pooling



machine learning and physics



arXiv:1701.08784v2