O2AT: Machine Learning

Francesco Mazzaschi

Machine learning: introduction

ALICE

- Machine learning (ML) models are able to perform a task without being explicitly programmed to do so
  - models are built from an input sample data and try to extract patterns from it
  - no need to specify a sequence of instructions to solve the task

- ML is nowadays ubiquitous in real-world applications
  - speech recognition and translation
  - autonomous driving
  - text-to-image generation
  - HEP!
  - ….

ChatGPT: chatbot based on deep learning

F write a python function for fitting a TH1D with a Gaussian + a pol2. Stay in 20 lines

Here's a Python function for fitting a TH1D histogram with a Gaussian plus a pol2 function in 20 lines:

python Copy code

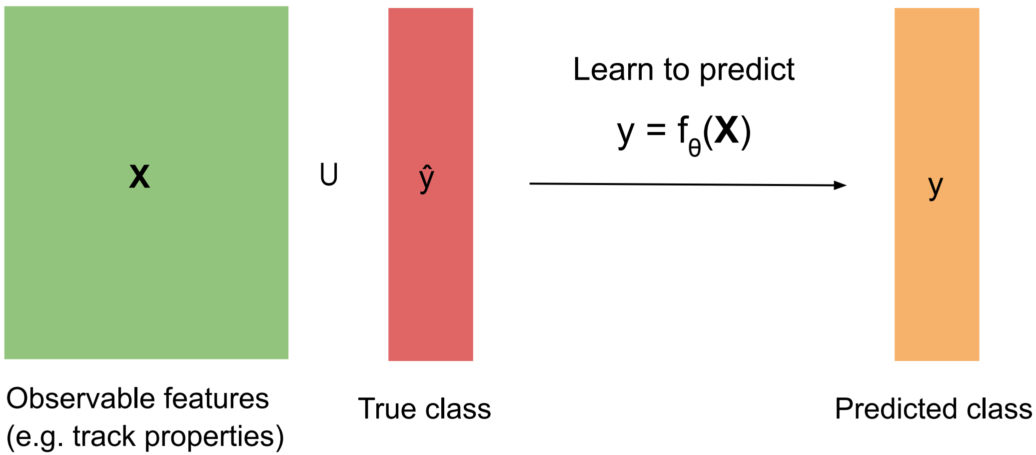
```python
import ROOT

def fit_histogram_with_gaussian_pol2(histogram, title="Invariant Mass Fit"):
    fitFunc = ROOT.TF1("fitFunc", "gaus(0) + [3]*x*x + [4]*x + [5]", 0, 5)
    histogram.Fit("fitFunc")
    canvas = ROOT.TCanvas("canvas", title, 800, 600)
    histogram.Draw()
    fitFunc.Draw("same")
    return fitFunc

# Example usage:
# inputFile = ROOT.TFile("your_input_file.root")
# histogram = inputFile.Get("your_histogram_name")
# fitted_function = fit_histogram_with_gaussian_pol2(histogram)
# inputFile.Close()
# ROOT.gApplication.Run()
```

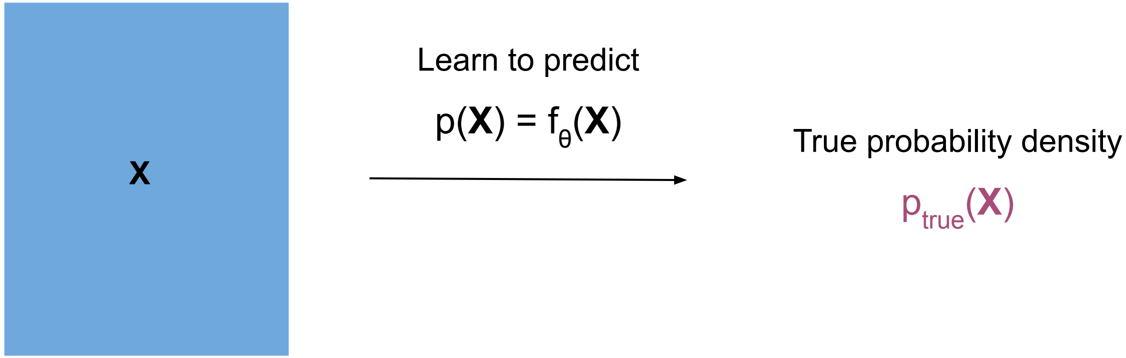
Supervised Learning

- The desired output of the task to be performed is known and a set of examples is available
- Typical tasks
  - classification: distinguish between a pair of or several classes (e.g. signal vs bkg.)
  - regression: predict a continuous value (e.g. particle energy)
- Model "trained" to infer some target starting from the input data
  - ideally the model output matches the target for "unseen" data



Unsupervised Learning

- No examples with known labels are available
  - model “learns” the probability distribution from the input data
- Typical tasks
  - clustering: find structures in the data
  - anomaly detection: identify outliers w.r.t. the input data
  - sampling: generate data from the underlying probability distribution



Observable features
(e.g. track properties)

ML in ALICE: few examples

Signal-vs-background classification

- Boosted Decision Trees (BDTs) and Neural Networks (NN) replacing “traditional” linear selections

Jet p_T reconstruction

- correction for the background from the underlying event
- regression task using shallow NN

Heavy flavor jet tagging

- BDTs and Deep Neural Networks (DNN) to tag heavy-flavour jet topologies

HF-hadron trigger

- BDTs to trigger on displaced decay-vertex topologies

Particle identification (PID)

- exploit complex relationship between track properties and PID
  - NNs to combine info from different detectors
  - PID with ITS2 using BDT regression

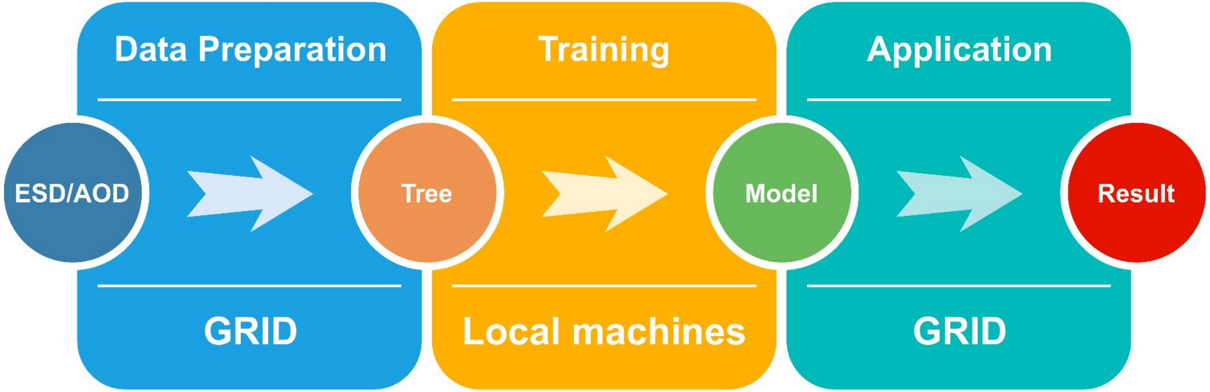
TPC response calibration

- ML to compute corrections of spatial charge distortions
- NN for energy-loss (dE/dx) calibration

Clustering

- TPC clustering with DNNs

ML workflow



Data preparation

- ➢ Information written from AO2D to ROOT TTree
- ➢ Only data needed for training downloaded locally
  - ○ a few GBs independently from the collision system

Training and optimisation

- ➢ Small fraction of real data and all MC simulations used to train/optimise the model
  - ○ a few minutes/hours on laptops or desktops

Inference on full data sample

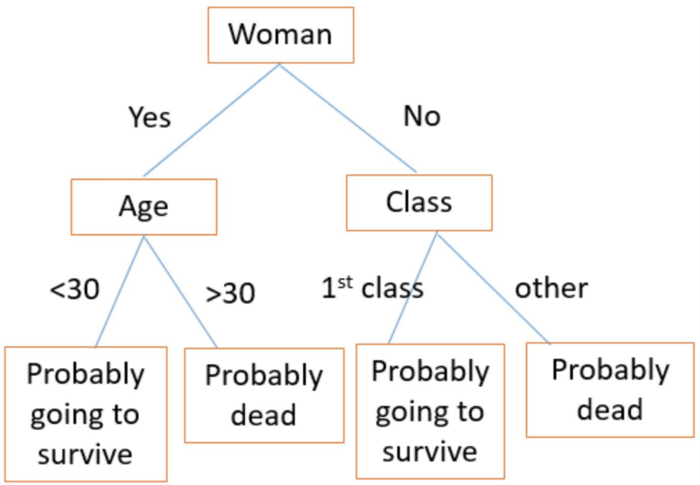
- ➢ From about 1 to 3 days on the GRID
  - ○ usual time for a train run from the user point of view
  - ○ the ML inference can be added to standard analysis tasks

Hands-on session

- Focused on Boosted Decision Trees (BDTs) training and inference for physics analyses
- Divided in two parts
  1. Training and testing a BDT model (S. Politanò)
     - Classification of the $D_s^+ \rightarrow K^+K^-\pi^+$ signal
     - Python software strack, hipe4ml package developed in ALICE for ML based physics analyses
  2. Apply the BDT to data with a O2Physics task (F. Catalano)
     - Inference using the O2Physics interface for ONNXRuntime

Boosted Decision Trees

- Hands-on session focused on BDTs
- Simple (apparently) supervised learning model well suited for classification and regression problems
- Building block → Decision Tree (DT)
  - A sequence of simple tests on the variables of the candidate
  - Combining all the tests one gets an output as a function of the variables of the single candidate
- Training a DT:
  - each test is built to maximise the separation between the signal and the background classes



DT applied to the Titanic dataset: was the passenger survived?

Boosted Decision Trees

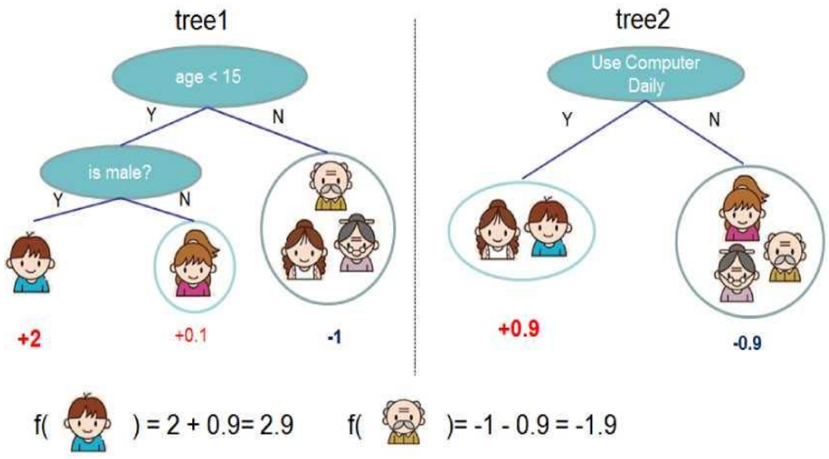
- DT: poor performances on independent samples → overfitting

Boosting

- Many simple (shallow) trees built sequentially
- Each tree is built to compensate the errors of the previous one

Ensemble model

- predictions are made combining the output of all the trees
- Very resilient to overfitting



Do they like computer games?
Score based approach to evaluate it

Hands-on session

- Focused on Boosted Decision Trees (BDTs) training and inference for physics analyses
- Divided in two parts
  1. Training and testing a BDT model (S. Politanò)
     - Classification of the $D_s^+ \rightarrow K^+K^-\pi^+$ signal
     - Python software strack, hipe4ml package developed in ALICE for ML based physics analyses
  2. Apply the BDT to data with a O2Physics task (F. Catalano)
     - Inference using the O2Physics interface for ONNXRuntime

ML inference

- ML applications in ALICE usually based on
  - python software stack (scikit-learn, XGBoost, TensorFlow, PyTorch, …)
- Application of models on the GRID (in your analyses!)
  - how to use a model trained in python in the ALICE C++ software?
- Run 3: ONNXRuntime
  - integrated in O2/O2Physics, available on GRID/EPN/FLP
  - supports almost any ML model (BDT, NN, …) and library (XGBoost, PyTorch, TensorFlow, ...)

MLResponse in O2Physics

- MLResponse class implemented in O2Physics
  - link
  - Interface for smooth ML inference

```
// TypeOutputScore is the type of the output score from o2::ml::OnnxModel (float by default)
template <typename TypeOutputScore = float>
class MlResponse
{
 public:
  /// Default constructor
  MlResponse() = default;
  /// Default destructor
  virtual ~MlResponse() = default;
```

```
protected:
 std::vector<o2::ml::OnnxModel> mModels;                     // OnnxModel objects, one for each bin
 uint8_t mNModels = 1;                                        // number of bins
 uint8_t mNClasses = 3;                                       // number of model classes
 std::vector<double> mBinsLimits = {};                        // bin limits of the variable (e.g. pT) used to select which model to use
 std::vector<std::string> mPaths = {""};                      // paths to the models, one for each bin
 std::vector<int> mCutDir = {};                               // direction of the cuts on the model scores (no cut is also supported)
 o2::framework::LabeledArray<double> mCuts = {};              // array of cut values to apply on the model scores
 std::map<std::string, uint8_t> mAvailableInputFeatures;     // map of available input features
 std::vector<uint8_t> mCachedIndices;                         // vector of indices correspondance between configurable and available input features
```

MLResponse in O2Physics

- MLResponse class implemented in O2Physics
  - link
  - Interface for smooth ML inference
  - Wrappers based on it for the different PWGs

HfMlResponse.h

HfMlResponseDplusToPiKPi.h

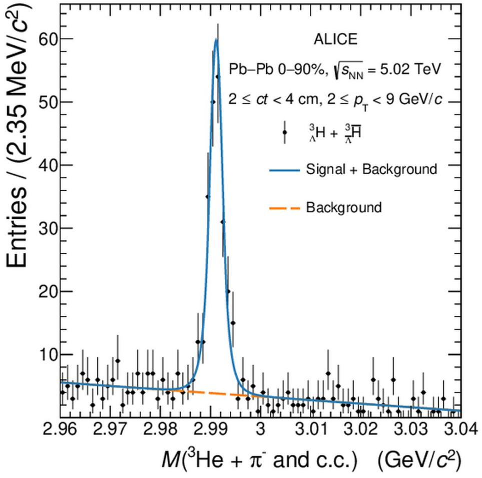
```
/// ML selections
/// \param input is the input features
/// \param pt is the candidate transverse momentum
/// \return boolean telling if model predictions pass the cuts
template <typename T1, typename T2>
bool isSelectedMl(T1& input, const T2& pt)
{
  auto nModel = findBin(&mBinsLimits, pt);
  auto output = getModelOutput(input, nModel);
  uint8_t iClass{0};
  for (const auto& outputValue : output) {
    uint8_t dir = mCutDir.at(iClass);
    if (dir != o2::cuts_ml::CutDirection::CutNot) {
      if (dir == o2::cuts_ml::CutDirection::CutGreater && outputValue > mCuts.get(nModel, iClass)) {
        return false;
      }
      if (dir == o2::cuts_ml::CutDirection::CutSmaller && outputValue < mCuts.get(nModel, iClass)) {
        return false;
      }
    }
    ++iClass;
  }
  return true;
}
```

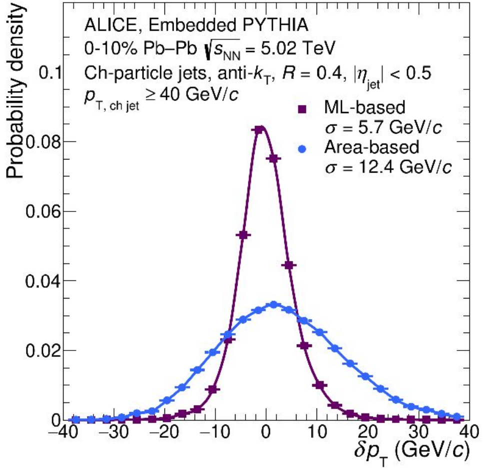
Useful resources - ML

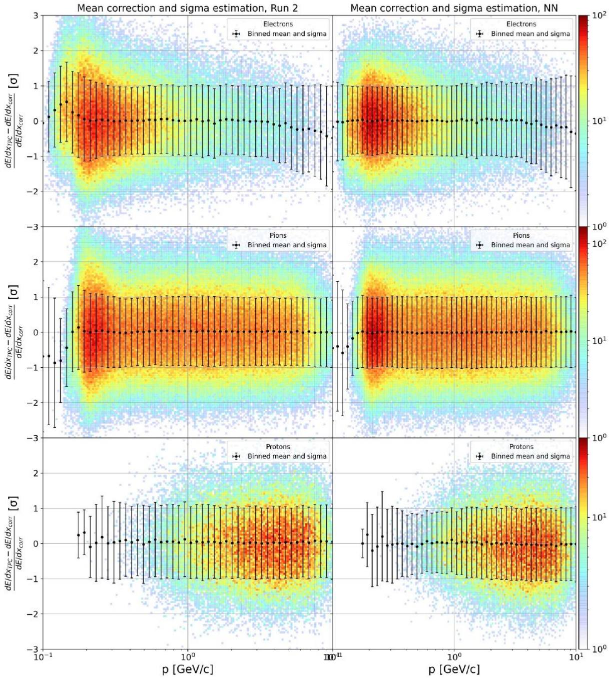
- ML theory from scratch
  - https://work.caltech.edu/telecourse , Learning from Data, Yaser Abu-Mostafa
- Hands-on ML book
  - Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow
- XGBoost BDTs
  - https://xgboost.readthedocs.io/en/stable/tutorials/model.html
- Common ML python libraries
  - https://scikit-learn.org/stable/
  - https://www.tensorflow.org/
  - https://keras.io/
  - https://pytorch.org/
  - https://onnx.ai/

Useful resources - ALICE

- ALICE-ML-Group
  - alice-machine-learning@cern.ch
  - Meetings ~ every two weeks (Tuesday or Wedenesday at 16:00 CET)
    - https://indico.cern.ch/event/1339879/
  - Contact:
    - Me: francesco.mazzaschi@cern.ch
    - Fabio Catalano: fabio.catalano@cern.ch







Conclusion

- ML is a powerful tool employed at different levels in ALICE
  - In Run 2 we improved many physics analyses by using ML
  - In Run 3 we are using/developing ML for reconstruction, clustering, PID
- What hands-on session does not cover
  - Neural Networks, ML models optimization, proper choice of the model thresholds, how to deal with systematics
    - For this, please subscribe and follow ALICE ML group meetings!

Reminder

- Tutorial on ML in O2 on Friday, covering BDT training and inference
- Can be run locally or in dedicated SWAN notebooks!
  - https://github.com/AliceO2Group/analysis-tutorials/tree/master/o2at-3