Machine Learning techniques used in LHCb analyses and online applications

Adam Dendek
AGH University of Science and Technology Krakow
On behalf of the LHCb collaboration

- How Machine Learning allows us to improve Particle Identification?
- Is the Monte Carlo the only method to simulate the data? How can we leverage Deep Learning?
- Machine Learning as a part of the Online Trigger

Part 2 (Tutorial): How to build and verify Machine Learning model?

- Define the problem and select a proper model
- List of available open source libraries, that make your life easier
- How to validate the model predictions
- Model interpretability problem. Trade-off between accuracy and outcome understanding?
Part 1: Application of the Machine Learning techniques in LHCb
Charged particle identification

- Particle identification (PID) plays a crucial role in LHCb analyses. [DOI: 10.1088/1748-0221/3/08/S08005]
- The LHCb PID system is composed of two ring-imaging Cherenkov detectors (RICH), a series of muon chambers and a calorimeter system (ECAL and HCAL). [see Stefania Vecchi’s talk]
- Combining information from these subdetectors allows one to distinguish between various species of long-lived charged particles. (multiclass classification problem)
- Classifier trained on MC samples (kinematic properties)

We tested the following Models
- Shallow NN (TMVA)
- Deep NN (keras)
- Boosted Decision Trees (cat boost)
Generative Adversarial Networks (GAN)

**Predictive modes** learn conditional distribution $P(Y|X)$, when **generative** models try to learn the joint probability distribution $P(X,Y)$.

**The idea of GAN (Goodfellow, et. all)** [Goodfellow, et. al. (2014). Generative Adversarial Networks]:

GAN consists of two NN: Generator $G$ and Discriminator.

This two networks play a minmax game against each other to minimize:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))].$$
Examples of GANs (outside of HEP)

[Semantic Image Synthesis
arXiv:1903.07291]
GAN for Fast Calorimeter simulation

- The traditional way to get a simulated data is to perform a Monte Carlo simulations.
- The MC simulations require billions of CPU hours and constitute more than half of LHCb computing resources.
- The simulation of particle shower in calorimeters is the most computationally demanding step.
- The idea is to replace MC by GAN

GAN for Fast Calorimeter simulation: Results

$E_0 = 63.7 \text{ GeV}$  
$E_0 = 6.5 \text{ GeV}$  
$E_0 = 15.6 \text{ GeV}$  
$E_0 = 15.9 \text{ GeV}$
GAN for Fast Calorimeter simulation: Results

Weighted real-data and generated distributions of RichDLLk for kaon and pion track candidates in bins of pseudorapidity (ETA) and momentum (P) over full phase-space.
Track reconstruction at LHCb

- LHCb moved to real time reconstruction, alignment and calibration setup in Run II
- This allows to achieve the offline quality of the events selections performed at trigger level. [CERN-LHCb-DP-2019-001]
- The track reconstruction had to be made identical online and offline.
- We need to make track reconstruction faster without performance loss.
- The two most important track types: [see Stefania Vecchi’s talk]
  - **Long tracks:**
    - Hits in VErtex LOcator, Inner Tracker and/or Outer Tracker and can have hits in TT
    - Used in majority of analyses
  - **Downstream Tracks:**
    - Hits in TT and IT/OT
    - Tracks from daughters of long lived particles
  - **T-Seeds (T tracks)**
    - Track segments reconstructed in T station
The downstream tracking algorithm contains following steps (listed only the most important ones from the ML point of view):

- The algorithm is seeded by tracks reconstructed in T stations
- Reject as T-Seeds that cannot be reconstructable,
- Find matching TT hits
- Accept downstream tracks candidates
T-Seed classifier performance

The current version of classifier was trained using $2 \times 10^6$ track seeds from simulated samples containing $B \rightarrow J/\psi K_s$ signal decay.

The classifier scores - 0.86 (ROC auc).

It corresponds to the rejection of about 30% of fake T-Seeds

Above results obtained for validation set that constituted 20% of the input sample

[CERN-LHCb-PUB-2017-001]
Part 2 (Tutorial): How to build and verify Machine Learning model
(recommended) Machine Learning Tools

- The open source tools, that I used during my studies on T-Seed classifier:
  - Python
  - Jupyter notebook (swan and Google Collab)
  - Numpy, scipy, matplotlib
  - Sklearn
  - Xgboost
  - Pytorch

- Cloud computing and GPU clusters:
Tuning hyperparameters

The hyperparameters are models parameters, that are not learnable and have to be set prior to training.

Tuning hyperparameters is very similar to configuring old tv set.

Each of the hyperparameters represents one knob.

The interaction between each of the knobs are usually unknown.

What strategies can be implemented?
Hyperparameters (HP) optimization

The two most common methods to tune HP are **Grid** and **Random Search**.

**Grid Search**: is simply an exhaustive search through a manually specified subset of the hyperparameter space of a learning algorithm.

**Random Search**: technique where random combinations of the hyperparameters are used to find the best solution for the built model.

Grid and Random Search does not keep track of past evaluation results.

We can create **surrogate model** of the objective function:

\[ P(\text{score} \mid \text{hyperparameters}) \]

Surrogate Model = **Gaussian Process** (posterior distribution over functions)

As the number of observations grows, the algorithm becomes more certain of which region to explore.

**Acquisition function** - balances exploration and exploitation
Why should I trust you?

- Despite widespread adoption, machine learning models remain mostly black boxes.
- In some applications, models prediction trigger critical actions like cancer treatment, fire an employee or attack on terrorist base.
- Is the global accuracy sufficient?

[doi:10.1145/2939672.2939778]
LIME - Local Interpretable Model-Agnostic Explanations

- The idea is to simplify complex model prediction by its interpretable representation.
- The interpretation is calculated for a single instance.
- In order to ensure both local fidelity and interpretability, we need to minimize the following formula:

\[ \xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \]

[doih:10.1145/2939672.2939778]
LIME explanation example

Local explanation

- LSTAT <= 7.31
- RM > 6.59
- PTRATIO <= 17.40
- AGE <= 46.95
- 0.45 < NOX <= 0.54
- CHAS=0
- 375.02 < B <= 391.43
- ZN <= 0.00
- RAD=8
- 3.20 < DIS <= 4.84

Local explanation

- LSTAT > 16.55
- 6.16 < RM <= 6.59
- NOX > 0.62
- DIS <= 2.10
- 375.02 < B <= 391.43
- CHAS=0
- 19.10 < PTRATIO <= 20.20
- AGE > 94.10
- 330.00 < TAX <= 666.00
- 9.90 < INDUS <= 18.10
SHAP (SHapley Additive exPlanations)

- SHAP connects game theory with local explanations.
- In a game there is N players, each of it may cooperate and obtains certain gain from cooperation.
- Taking into consideration that some players may contribute more to the coalition the question is how to fairly share the payoffs?
- The answer for this question is known as Shapley value:

\[
\varphi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! (N - |S| - 1)!}{N!} (v(S \cup \{i\}) - v(S))
\]

- Same idea can be used to measure the features importance.

T-Seed filter SHAP explanation

Overall features importance based on SHAP value:

[Graph showing Global feature importance and Local explanation summary]

[Graph showing Shap dependence plots]

[CoRR abs/1705.07874]
How to speed-up the classifier?

- Using continuous input features to calculate classifier response takes too much CPU time.
- The idea is: Discretise input features space and for each of bin calculate classifier response.
- Instead of calculating response for each tree we just need to take one number from the lookup table.
- The bBDT evolution time complexity is $O(1)$!
- The table has very complex structure, no way to fit approximation function.
- This idea comes from previous study on LHCb HLT [doi:10.1088/1748-0221/8/02/P02013]
Bonsai Boosted Decision Trees

- In order to reduce the computational complexity of the model, the generation of very long and complex if-else based function should be avoided.
- Instead, the input feature space is divided into bins.
- For each of these bins, classifier response is calculated.
- To evaluate classifier we need to find bin indices and take corresponding number from the lookup table.
Conclusion

- The LHCb is pioneering in application of Machine Learning
- The ML techniques are leveraged in various areas such as LHC data analysis, simulation, reconstruction
- Do not limit yourself to ROOT, try different open source tools
- Do not treat ML model as a black box, try to understand its outcome
4. PatLongLivedTracking: a tracking algorithm for the reconstruction of the daughters of long-lived particles in LHCb, LHCb-PUB-2017-001
5. Design and performance of the LHCb trigger and full real-time reconstruction in Run 2 of the LHC, CERN-LHCb-DP-2019-001
7. Random Search for Hyper-Parameter Optimization
Thank You
For Your Attention