14. Signal-background classification with Parametric Neural Networks

Author(s)

Name	Institution	Mail Address	Social Contacts
Luca Anzalone	University of Bologna, INFN Sezione di Bologna	luca.anzalone2@unibo.it	N/A
Tommaso Diotalevi	University of Bologna, INFN Sezione di Bologna	tommaso.diotalevi@unibo.it	N/A

How to Obtain Support

Mail	luca.anzalone2@unibo.it
Social	N/A
Jira	N/A

General Information

ML/DL Technologies	pNN classifier
Science Fields	High Energy Physics
Difficulty	low
Language	English
Type	runnable; fully annotated

Software and Tools

Programming Language	Python
ML Toolset	Keras, Tensorflow 2
Additional libraries	scikit-learn, numpy, pandas, matplotlib, mlhep
Suggested Environments	Google CoLab, Docker, own PC

Needed datasets

Data Creator	

Data Type	Simulation
Data Size	2.4 + 1.2 Gb (840 + 440 Mb compressed)
Data Source	HEPMASS (UCI ML repository); HEPMASS-IMB (Zenodo)

Short Description of the Use Case

The problem of **signal-background classification** is an important part of physics analyses, since an improved classifier helps to achieve more statistically relevant results. The task is usually framed as *binary classification*, in which the positive class is represented by the signal, and the negative one by the background since we want to reject it as much as possible while preserving (i.e. correctly classifying) the most signal we can.

Such problem can be solved either manually (with a *cut-based approach* by determining selection thresholds on multiple variables) or by means of machine learning. The machine learning approach we are going to discuss is about **parametric neural networks** (pNNs), which are a specialized kind of neural network classifier able to leverage a *physics parameter,* like the particle's mass. Such design enables the pNN to replace a set of classifiers each trained at a particular value of the physics parameters: e.g. each model trained on a specific signal mass hypothesis.

Another benefit of pNNs is their ability to *smoothly interpolate* intermediate values of the physics parameter, in a natural and consistent way: neural networks are notably smooth functions, but overfitting may prevent to achieve interpolation on some or all intermediate values of the parameter at all. In this regard is usually useful to ensure enough regularization of the network: for example, we use a combination of dropout and weight decay.

Finally, the main design decisions to consider when defining a pNN, are:

- Which **conditioning mechanism** to use, responsible of combining the input features (or intermediate output) with the given physics parameter.
- How to **assign the physics parameter** to the background data samples.
- How to **leverage the domain knowledge** about the data to improve the training of the model.

How to execute it

The full code that supports this tutorial is available at:

GitHub - Luca96/affine-parametric-networks/tutorial

The provided tutorial notebook can be run, either: manually (requires installing the dependencies, and downloading the datasets), or through our docker image (configured with libraries and data).

Manual Setup

Assuming a working Python setup, also having Jupyter notebook or lab (see here) already installed:

1. Clone the repository `tutorial` branch, and move within the folder:

```
git clone https://github.com/Luca96/affine-parametric-networks.git --branch tutorial
# if on Google Colab, use %cd instead
cd affine-parametric-networks
```

2. Run the notebook `tutorial.ipynb` either on your local machine, or via Google Colab:

```
# on a terminal:
jupyter notebook tutorial.ipynb
```

3. Copy the URL given in the terminal output on your browser.

Docker Setup

Assuming a working Docker installation, on a terminal:

1. Clone the `tutorial` branch and change directory, as described above at step 1.
2. Run the container (this also downloads the image which is about 4.3 Gb):

```
docker run -it -d -v ${PWD}:/affine-parametric-networks -w /affine-parametric-networks -p 8888:8888 --
name tutorial tommaso93/affine-parametric-networks
```

3. Execute the image named `tutorial`:

```
docker exec -it tutorial jupyter notebook --ip 0.0.0.0 --no-browser
```

4. Copy the URL starting with "http://127.0.0.1" given in the terminal output, and paste it on your browser. In alternative, type "http://localhost:8888 /tree" on your browser, and then insert the token provided on the terminal output (the one starting with "?token="). Then run the `tutorial.ipynb` notebook (skipping the installation of the dependencies - "Set-up" section).

References

Improving parametric neural networks for high-energy physics (and beyond) - MLST

Parameterized neural networks for high-energy physics - EPJ

Presentation made on 13 Feb 2023 : https://agenda.infn.it/event/34607/contributions/190747/attachments/101798/141949/pNNs_ML_INFN.pdf