Machine learning in ALICE

Tutorial & hands-on practice

Rüdiger Haake (CERN)
(23.03.2018)

Outline

- This tutorial consists of two parts

  1) Basic introduction to some concepts of machine learning. Focus: Neural networks
  2) Hands-on session: Examples for classification & regression

- Don’t miss the

IML

2nd IML Machine Learning Workshop

9-12 April 2018, CERN (Vidyo available)

https://indico.cern.ch/event/668017/

- Tutorials on methods & concepts (from actual experts!)
- Industry talks by Google, IBM, …
- Modern ML applications
- …

Concepts

What is Machine Learning?

What is Machine Learning?

We want to use Machine Learning techniques to

- solve a problem for us that we can teach,
- and to be robust against obstacles

What is Machine Learning?

- Wide field with increasing applications in research and industry
- First ideas already since 1950's
- Huge progress in last years, especially in deep learning

Basic ansatz of ML algorithms:

The algorithm improves its own performance on a task by gaining more experience

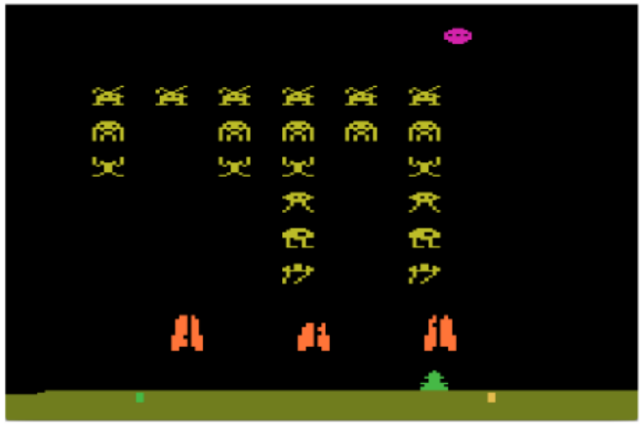
- In the simplest cases, these are more involved fitting algorithms
- More complex cases include several connected methods including huge neural networks

In a simplified view, one can divide two types of algorithms:

- **Supervised learning** algorithms get training data and learn the correct mapping of input data and desired output
- **Unsupervised learning** tries to find a structure in the data without a priori knowledge of desired outcome

What is Machine Learning?

- In HEP, many ‘classic’ ML methods are already in use.
  - Boosted decision trees (BDTs) in Higgs search
  - In ALICE, e.g. BDTs for signal extraction ($\Lambda_c \rightarrow K^0_s p$)
- ALICE uses (explicit) ML methods rarely, interest is rising



- Recently, most innovations done with deep learning methods
- Reached super human abilities in image recognition in 2015
  - see ImageNet challenge
- Much progress in self-driving cars, speech recognition, image tagging, playing ATARI games
- So far, deep learning methods are not very common in HEP

What is Machine Learning?

Important: Machine Learning is not the solution for all our problems

- Usually, one cannot just throw data and expect the algorithm to perform better than human even in a well-defined task
  - ML no replacement for domain knowledge
  - “Garbage in → Garbage out”
  - Still: algorithms, classifiers, and training data need to be selected carefully
- Deep learning is not necessarily better than classic ML methods

Development of Machine Learning techniques is incredibly fast

- A lot of work is going on, fruitful collaboration also with ML experts
  - Data science@LHC workshop
  - IML working group at CERN
- Several open implementations exist
  - SciKit, TMVA, Keras, TensorFlow…
  - Many solutions for “human problems” can also be applied to HEP problems (in general, pattern recognition)

Supervised learning

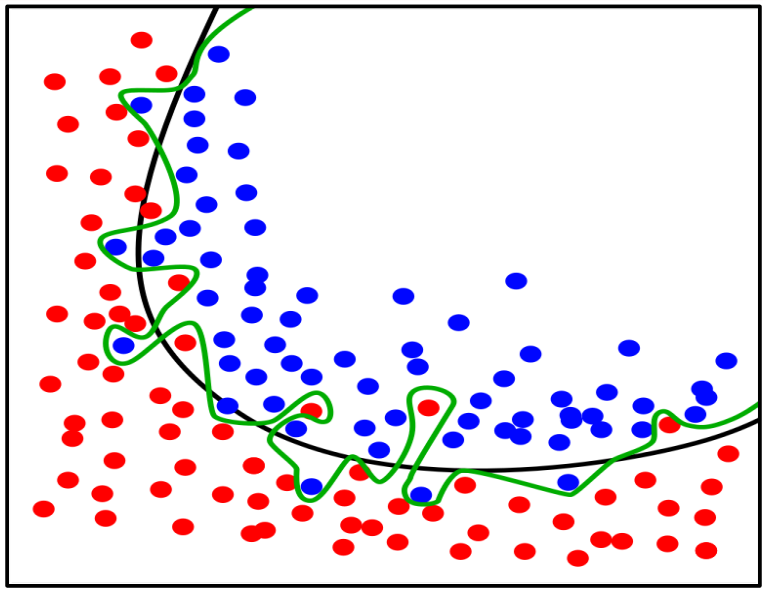
- Supervised learning helps with either
  classification or **regression** tasks

- **Classification:** Group samples into predefined classes
  Example: Tag b-jets
- **Regression:** Assign value to each sample
  Example: Jet-underlying background in Pb-Pb

- General approach:
  - Give training dataset to model where truth is known (labeled data)
  - Model is optimized according to a loss function to produce the correct output. In a neural network: Weights are adapted
  - Idea: Model learns general features and gives correct output for unknown samples
  - Output can be a class (e.g. b-jet or not) or a value (e.g. background)

Supervised learning

- Important: Performance of the model must not be evaluated on the training dataset
- Model “has seen” training data and could have learned dataset-specific features
- This won’t help on general data
- This is called overfitting and can happen e.g. for too complex models
  Here the model works amazingly well on the training data, but fails on other data



Green line = overfitting

Solution: Strictly split training, validation, and testing data

- Training data: Used only for training, exclusively
- Validation data: Used for performance evaluation during training
- Testing data: Dataset used for “real physics”

Supervised learning

Crucial: Labeled training samples need to be as accurate as possible

- Classes must be well-defined
- Regression param. must be precisely known
- Train with garbage = Find garbage
- Usually this means we need a good Monte Carlo description
- There are techniques to improve the MC to better fit the data

- Good idea might always be:
  - Define a model that is as general as possible, not taking into account details which are in fact no general features
  - Adjust number of training samples to model complexity

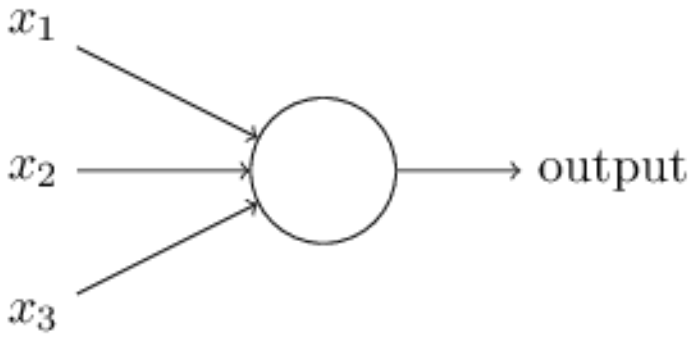
    O(samples) = O(free parameters)

Neural networks

- Several techniques on market
- Here focus on neural networks → deep learning

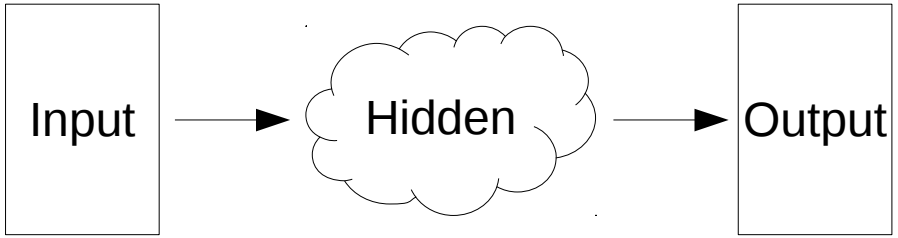
- Neural networks = Loosely inspired by biological neural networks
- Connected systems of nodes

Neuron:



- Several inputs
- Activation function that weights inputs
- Triggers output according to weights

Neural network:



- Chain of tensor operations, multiple neuron layers
- Multidim. input → multidim. output
- Massively parallelized → use GPUs

Neural networks

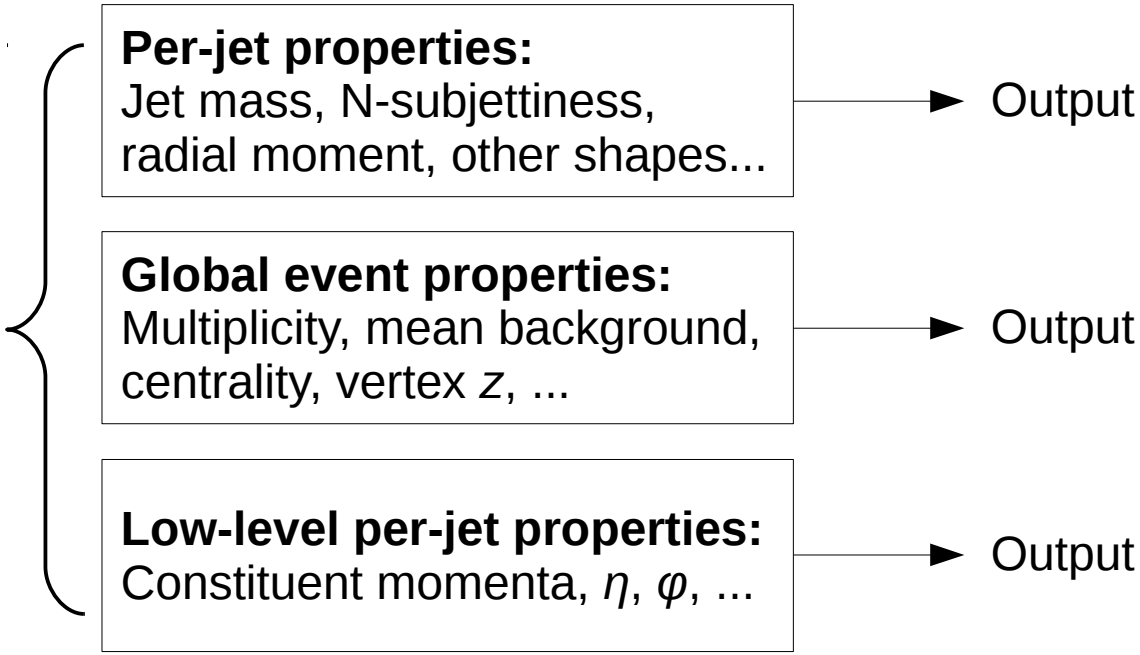
- Several techniques on market
- Here focus on neural networks → deep learning

- Neural networks = Loosely inspired by biological neural networks
- Connected systems of nodes

Network of networks:

Neural networks allow to combine useful networks
→ Powerful ansatz!

- Train several networks
- Inputs can be very different!



Neural networks

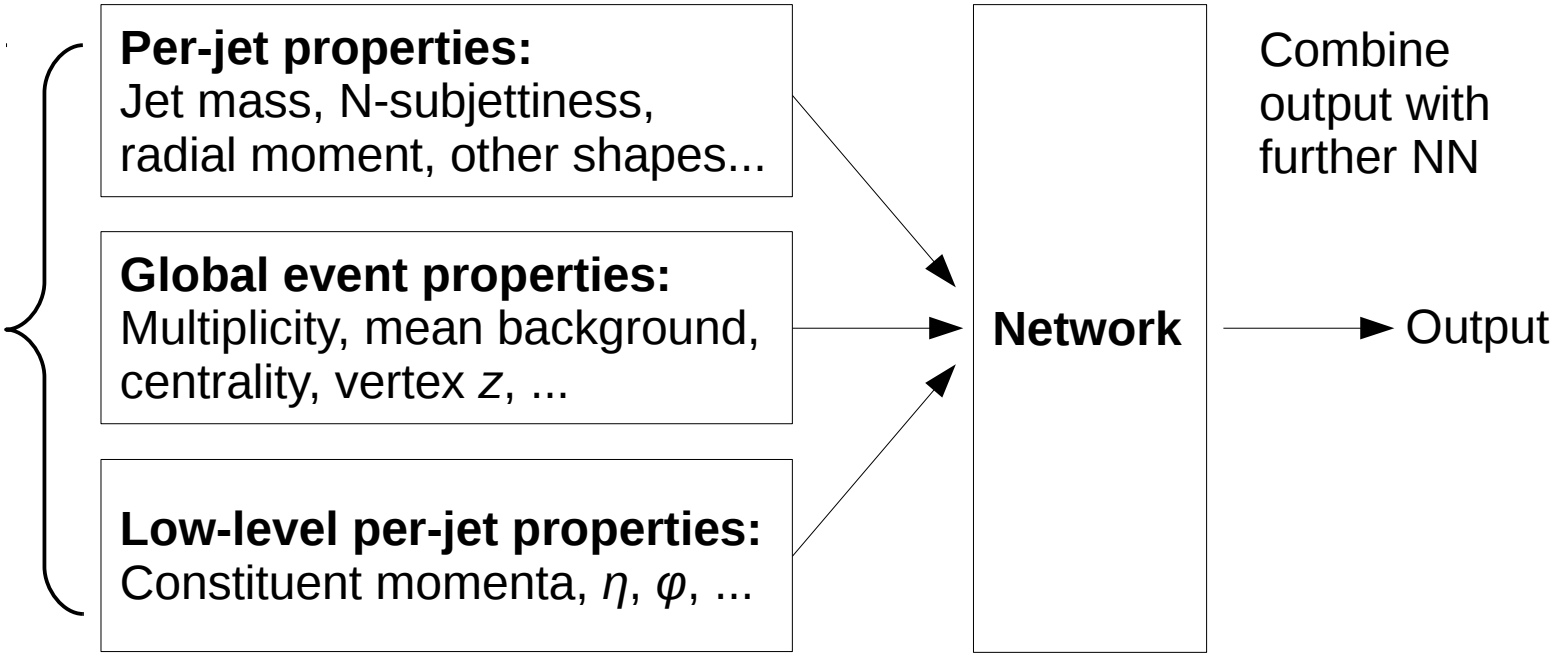
- Several techniques on market
- Here focus on neural networks → deep learning

- Neural networks = Loosely inspired by biological neural networks
- Connected systems of nodes

Network of networks:

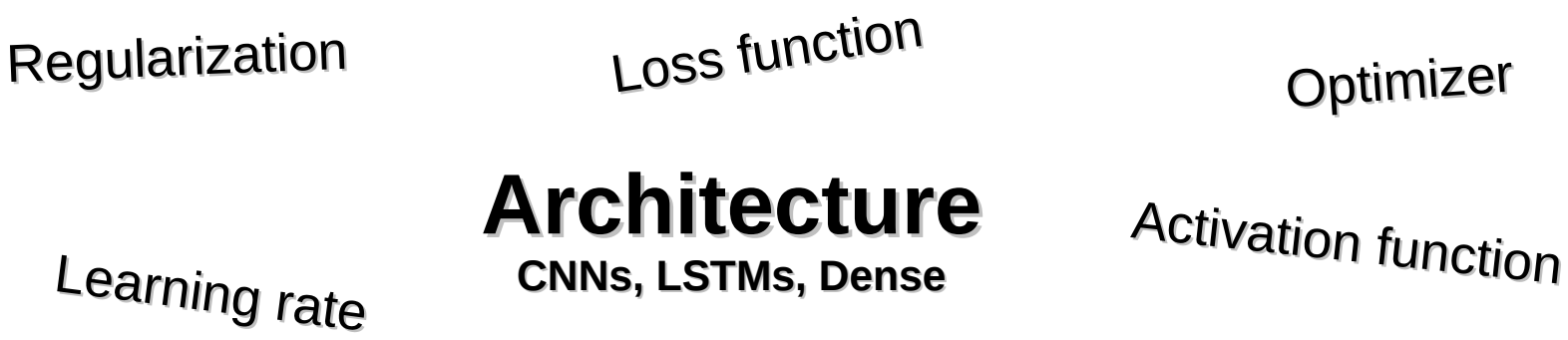
Neural networks allow to combine useful networks
→ Powerful ansatz!

- Train several networks
- Inputs can be very different!



Neural networks

Neural networks can be configured very differently



- Some settings can be chosen according to experience, e.g.:
  - Adam optimizer for deep networks
  - ReLU activation function

- There is a lot of knowledge available online for problems similar to those we face in physics
- Good settings are adapted to the problem

How to design a good model

Define your problem

- Is it a regression or classification task?
  Optimizer, loss function, activation function, etc. depend on this choice
- In a classification task, do you need multiple classes or does binary classification suffice?
  Binary classification might be easier to learn for the network than multi-class classification
- Will the problem only rely on high-level parameters?
  If yes, also different technique (like BDTs) can be considered
  High-level parameters are e.g. jet mass, jet shapes. Low-level parameters e.g. constituents

Define your dataset

- In case of classification, clearly define signal and background
- In case of regression, be sure your regression parameter is well defined and represents what you want
  Crucial step, better put more effort here than less
- Which input features could potentially have discrimination power for your problem? Implement them

How to design a good model

Define your model(s)

- Get inspired by similar problems, experiment with different designs
- Once you found a suitable design → Perform grid search
  Clever “brute force” trial of possible hyper parameters
  - Number of layers, neurons per layers …
  - Activation function, loss function, …

If suitable, combine several models on features

- Useful, if the models work on distinct input features
- Example: Combination of PID classifier models on TPC and TOF might be useful

Control parameters

- To monitor the training progress, several control parameters exist
  - Loss: Is directly minimized in the training
  - Only for classification:
    - Scores: Score distribution hint to model performance
    - ROC-Curve: Plots the true vs. false rate
    - AUC (Area Under (ROC) Curve): Performance

Keras output during training looks like this:

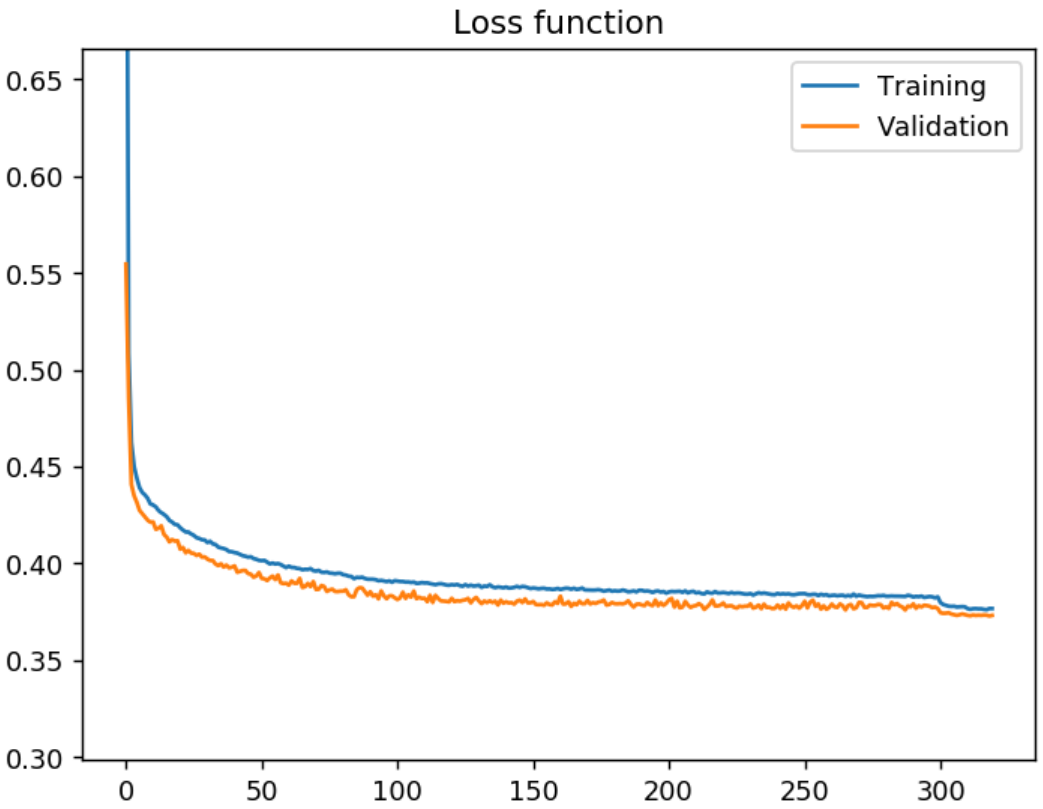
```
Train on 38000 samples, validate on 38000 samples
Epoch 1/1
38000/38000 [==============================] - 107s 3ms/step -
loss: 1.0340 - acc: 0.6808 - val_loss: 0.7285 - val_acc: 0.7342
```

Loss on training data: Good check that something is learned. Should get lower

Loss on valid. data: Check that learning is generalizable on unseen data.

Control parameters

Example: b-jet tagging

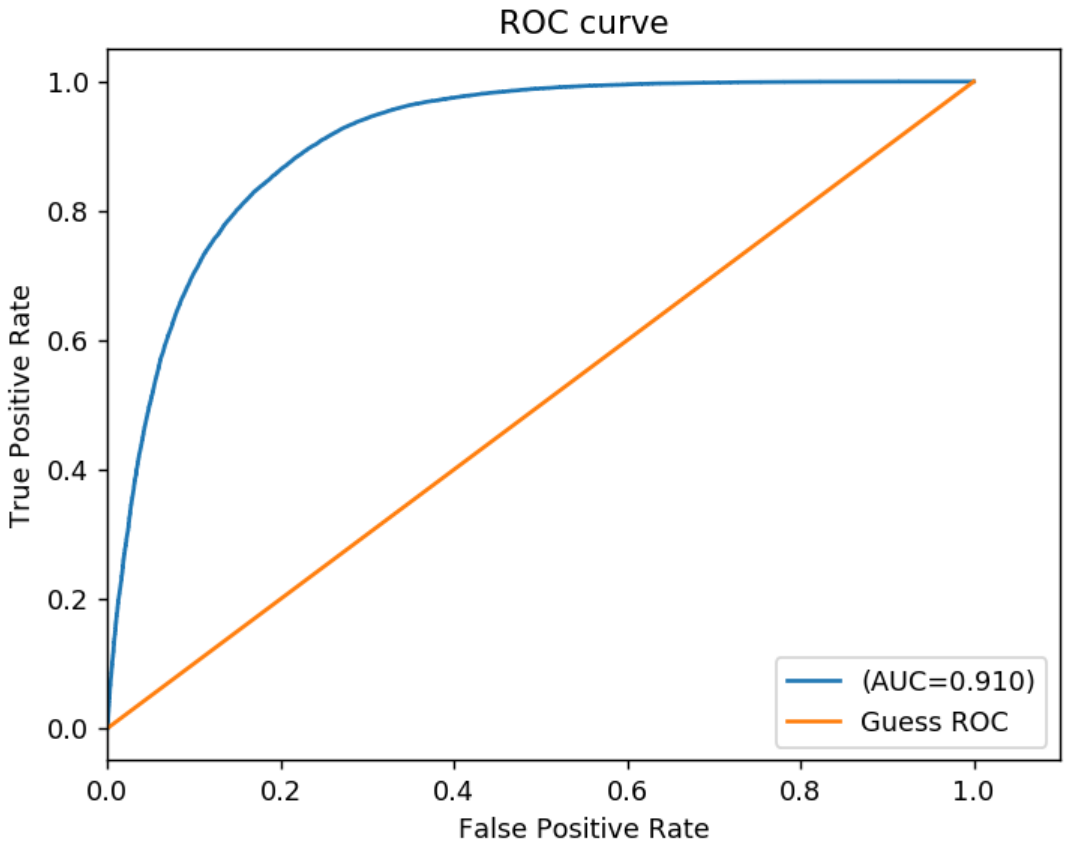


Typical control plot:

- Loss vs. training epoch
- Gets lower during whole training
- Seems to reach plateau
- This means:
  More training does not help anymore

Control parameters

Example: b-jet tagging

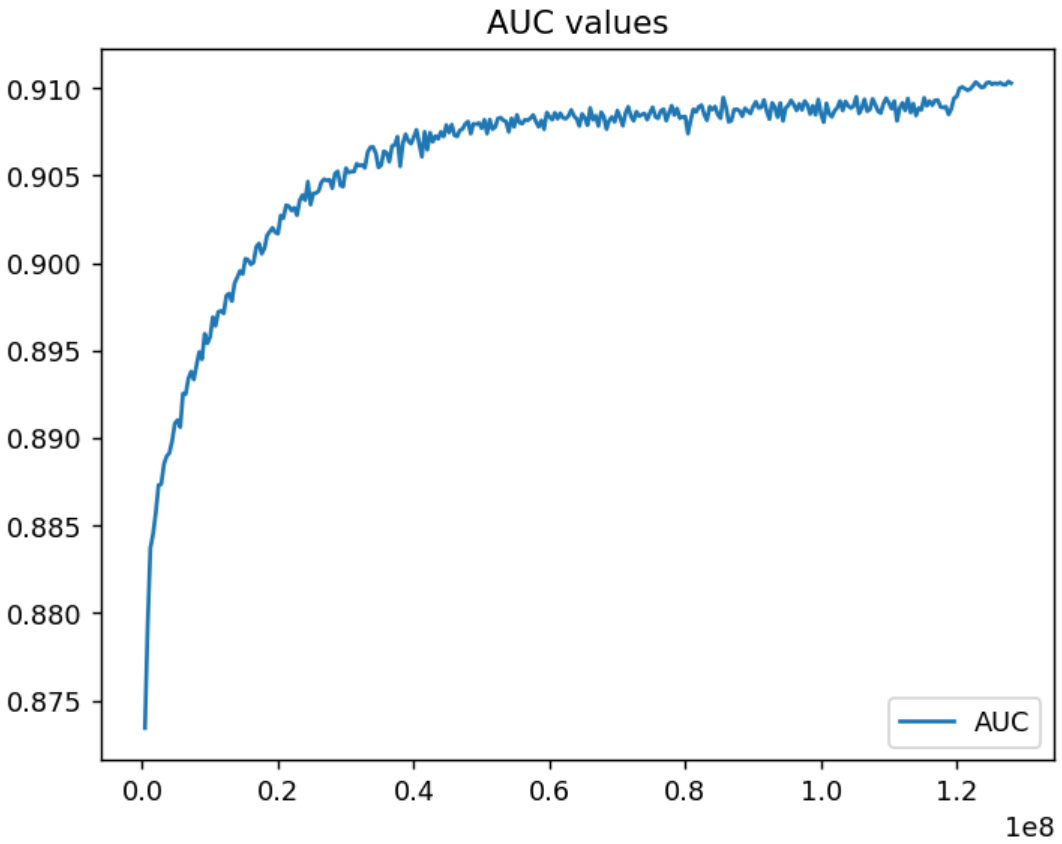


Typical control plot:

- ROC curve
- Optimally, the curve were always at 1
  → Full efficiency with lowest misclassification rates
- Orange line indicates how good guessing performed

Control parameters

Example: b-jet tagging

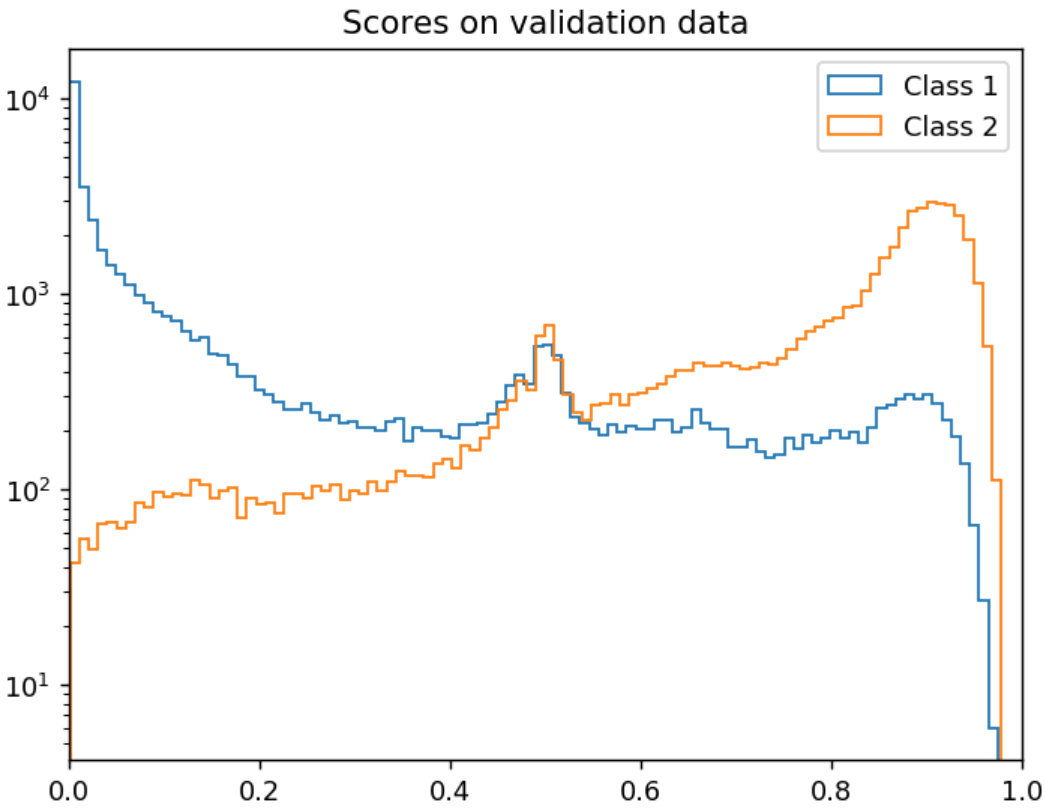


Typical control plot:

- AUC= Integrated area under the ROC curve
- Models shows good behavior → reaches plateau

Control parameters

Example: b-jet tagging



Typical control plot:

- Score distribution
- Good indicator how clear the two classes can be distinguished

Hands-on

Keras examples

Hands-on sessions

- The tutorial will be done on SWAN in your browser
- Prerequisites
  - You need a web browser
  - You need a CERN account
  - You need a CERNBox space
    (if never done before, login at https://cernbox.cern.ch/)

- When you click on the link below, notebooks are automatically copied to your CERNBox

https://swan004.cern.ch/hub/spawn?projurl=https://gitlab.cern.ch/rhaake/MLTutorial.git