LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS
partículas e tecnologia

Machine Learning

A blitz hands-on tutorial

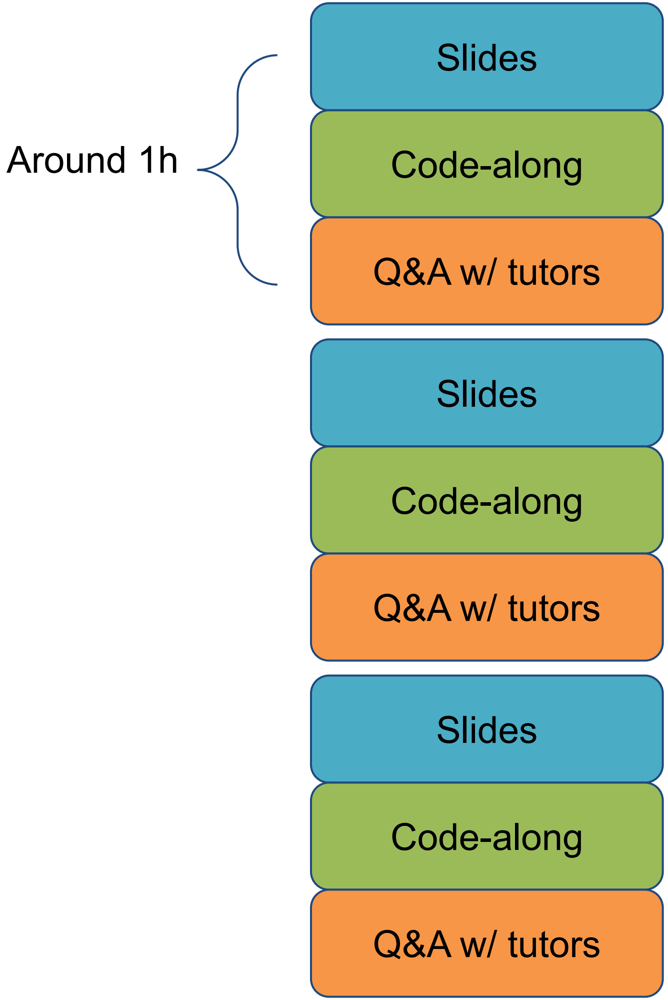
LIP Internship Program
Summer 2020

Miguel Crispim Romão
mcromao@lip.pt

How this tutorial will proceed
General idea

- I will guide you through some concepts using these slides
- We will then move on to Google Colab where I will guide you through a hands-on code-along tutorial to explore the concepts
- After each coding block, we will split the audience into breakout rooms (each with a tutor) for Q&A and clarifications

Big thanks to the helping tutors: Rute, Maura, Tiago!



How this tutorial will proceed
outline

- Part I: What is Machine Learning?
  - Types of learning -> Focus on classification tasks
  - Introduction to Scikit-Learn package
- Part II: Ensembles and Neural Networks
  - Ensembles of trees
  - Intro do Neural Networks
  - Keras/TF
- Part III: Best practices and Higgs Dataset
  - Hyperparameter and model choice
  - Best practices
  - Higgs Dataset

Part I - What is Machine Learning?

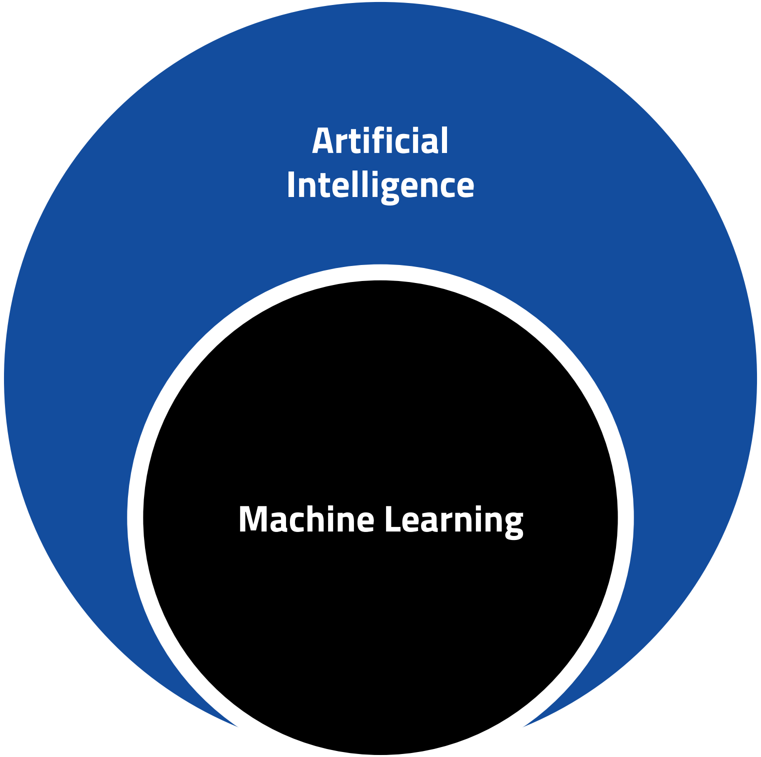
From an Artificial Intelligence Perspective

“ *Artificial Intelligence is the quest of creating machines that think and act intelligently*

Artificial Intelligence is a big topic
and covers many problems

- Reasoning and Problem-solving
- Knowledge Representation
- Planning
- **Learning**
- Natural Language Processing
- Perception
- Motion and Manipulation
- Social Intelligence
- “General Intelligence”

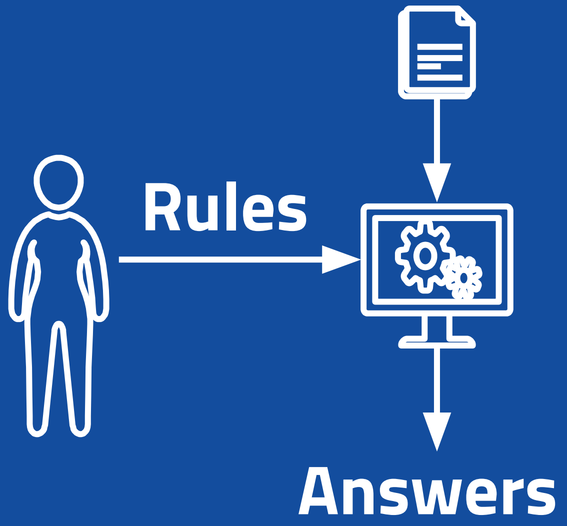
Machine Learning is the subfield of AI that concerns how a machine can learn to perform tasks



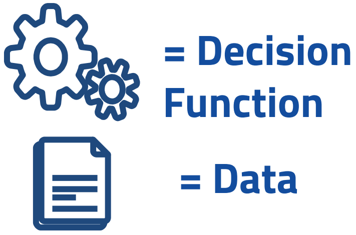
Self-Taught Code

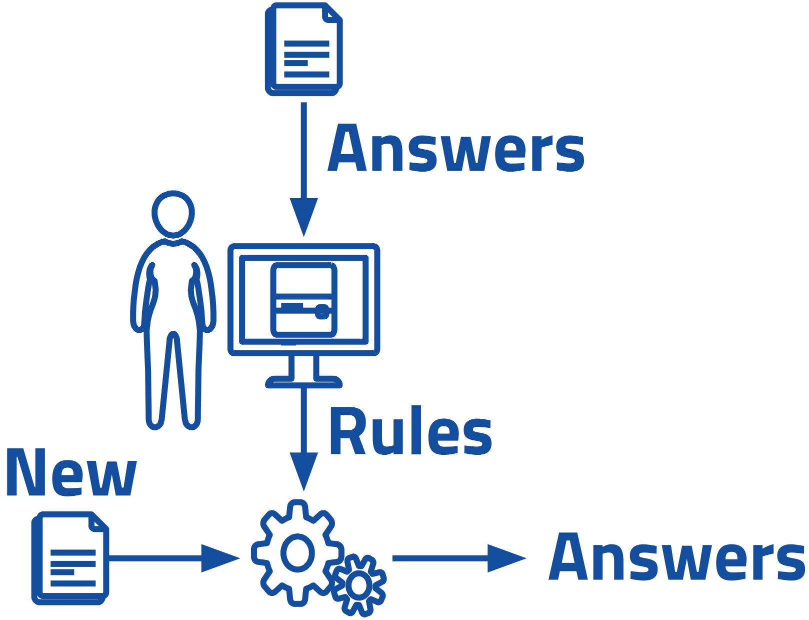
Machine Learning is a different paradigm of computing: a program that learns what it has to do

Classical Programming



Machine Learning





Machine Learning Taxonomy

What is out there and what tasks can we solve?

Machine Learning
Taxonomy: Types of Learning

The main differentiator is the type of learning, i.e. by **task**

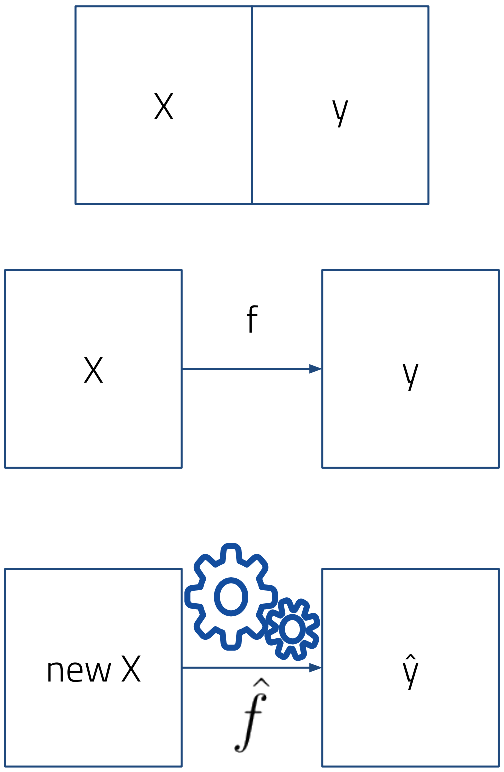
- Supervised
  - Data includes the answers
- Unsupervised
  - Algorithm embodies the answers
- Other types
  - Semi-supervised
  - Self-supervised
  - Reinforcement

Machine Learning
Taxonomy: Supervised Learning

- The training data includes the answer we want to reproduce
  - $\mathcal{D} = \{(X_i, y_i)\}$
  - X: Independent Variables/Features
  - y: Target Variables/Labels
- Assume (hope?) there exists a relation such that

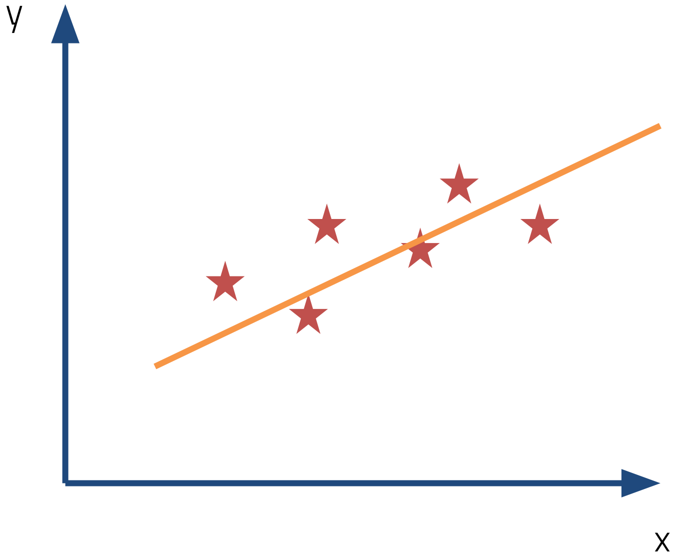
$$f : X_i \mapsto y_i$$

- The model will approximate f, $\hat{f}$
- The type of y defines two sub-classes
  - y is a real variable: **Regression**
  - y is categorical: **Classification**



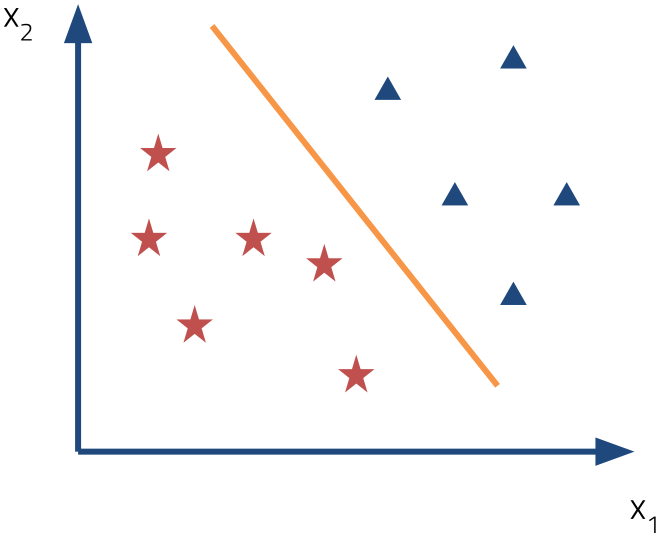
Regression Example

Linear Regression



$$y = wx + b$$

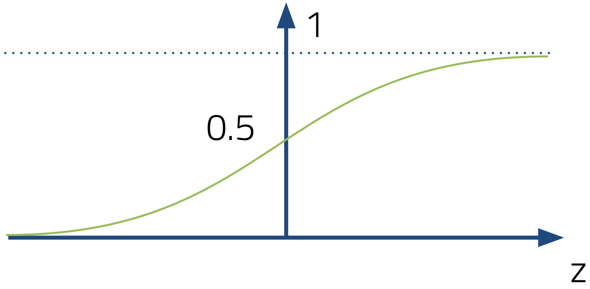
Classification Example
Logistic Regression



Sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-z}}$$

$$z = \vec{w} \cdot \vec{x} + b$$



Classification Example
Logistic Regression Training

- Measure the quality of the predictions with a differentiable function: **Loss function**
  - For classification: **Cross-entropy**

$$L = -\frac{1}{N}\sum_{i}^{N}\sum_{c}^{K} y_{i,k} \log p_{i,k}$$

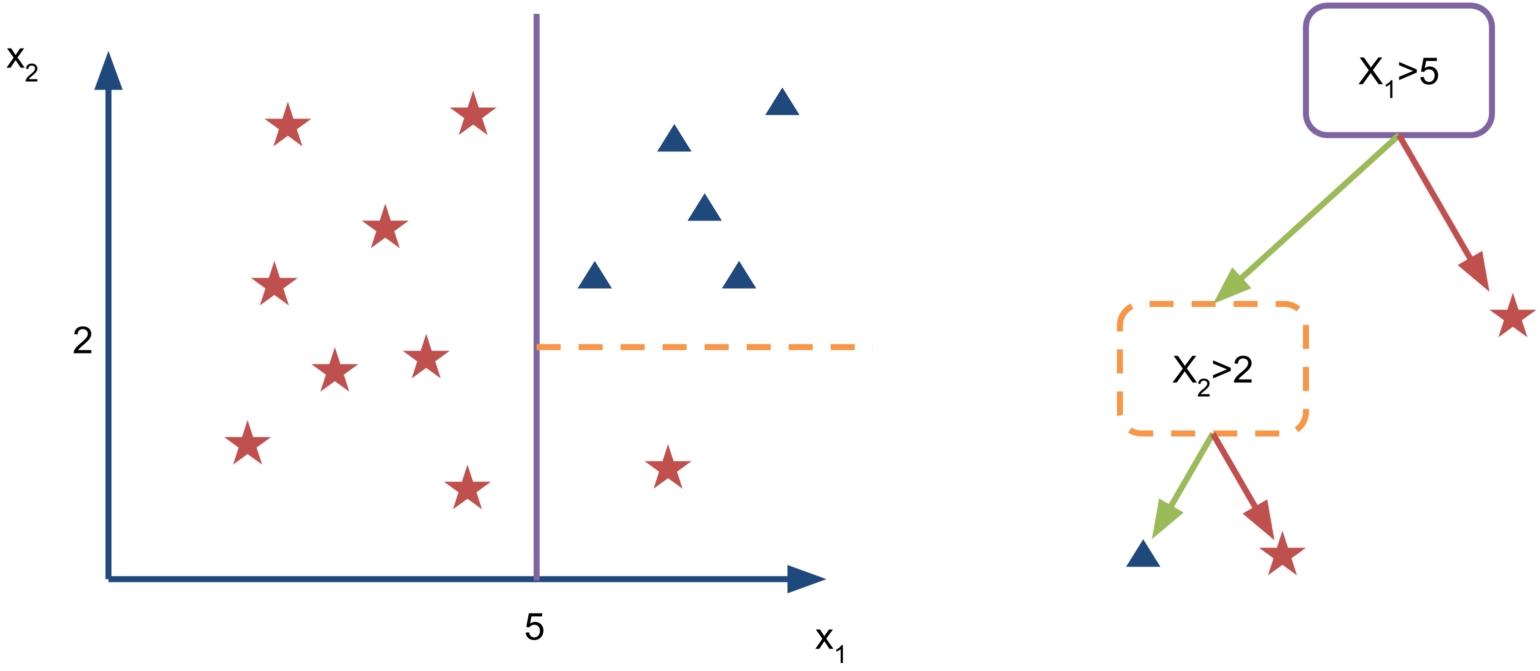
  - For the binary case: **Binary Cross-Entropy**

$$L = -\frac{1}{N}\sum_{i} y_i \log p_i + (1-y_i)\log(1-p_i)$$

- Iteratively correct the weights using **gradient descent**

$$w^{t+i} = w^t - \eta \nabla L$$

Classification Example
Decision Tree



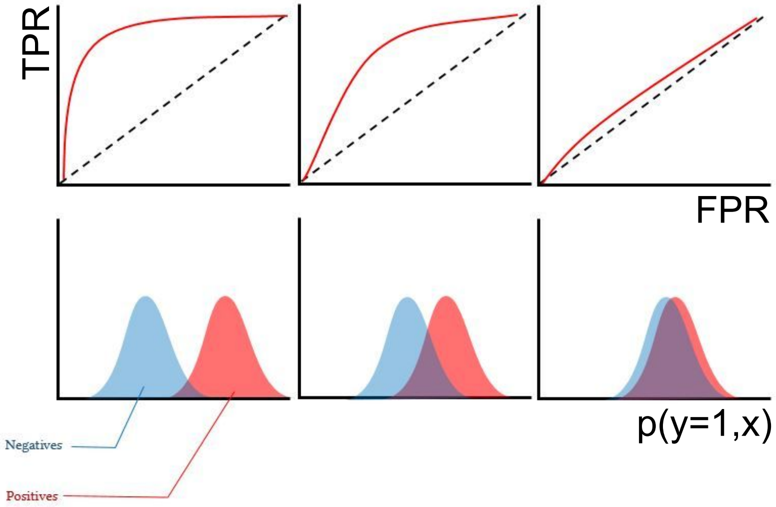
Classification Example
Decision Tree Training

- For each feature, order the points by their values
- Find a value for that feature that maximises purity of a class on each side of the split
  - You can measure this purity using Gini score or Entropy (NOT cross-entropy)
- Repeat until there are no more splits left -- either all truncations are pure in one class or each data point is in its own leaf

Machine Learning

How to evaluate a classifier

- There are many metrics in the Machine Learning literature that help you assess the performance of a classifier
- We will be focus on two
  - Accuracy: The percentage of instances that are correctly classified
  - Area under ROC (Receiver operator characteristic) curve



Cheatsheet:

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

1st hands-on

We will use Google Colab to run a few examples of classification algorithms using Scikit-Learn

Google Colab

- An online jupyter notebook host solution where you can do Machine Learning in Python
  - https://colab.research.google.com/
  - You do need a Google account
- It has all the relevant packages to do Data Science and Machine Learning pre-installed
- You can use GPU and TPU acceleration, for free

Scikit-Learn
and the python Machine Learning ecosystem

- Scikit-Learn (https://scikit-learn.org/) is the go-to ML package for python
- It defined the best practices for ML API development
- Has great documentation and tutorials
- **If this tutorial fails to teach you anything...**
  learn ML from Scikit-Learn documentation!

Scikit-Learn
and the python Machine Learning ecosystem

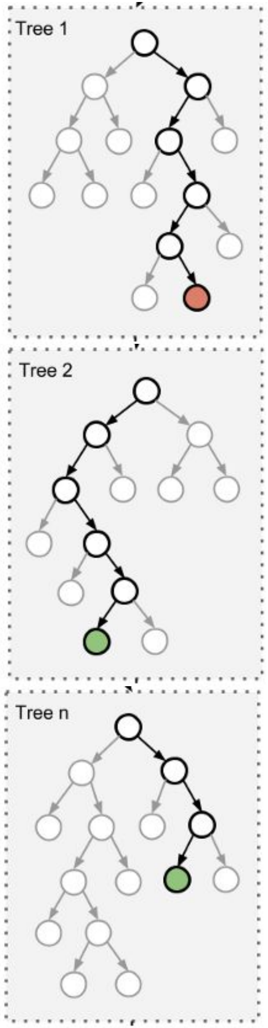
- We will start by implementing a logistic regression and a decision tree
  - `sklearn.linear.LogisticRegression`
  - `sklearn.tree.DecisionTreeClassifier`
- Not estimator modules worth remembering:
  - `sklearn.preprocessing`
  - `sklearn.model_selection`
  - `sklearn.metrics`

Ensembles and Neural Networks

Forests, neurons, and all that jazz

Ensembles
Strength in numbers

- An Ensemble is an... ensemble of ML models
- The idea is that the many weaker learners perform better together and produce a stronger learner
- Example: Random Forest is a collection of smaller trees (with a maximum depth) trained on subsamples of the data (bootstrapping)
    - The final prediction is given by average of the predictions -> This gives better generalisation than using a big tree alone
    - `from sklearn.ensemble import RandomForestClassifier`



Ensembles
Come in different shapes

- Although most of the ensembles techniques are based in Trees as the base model, there are many ways of building
  - I already mentioned Forests (a type of Bagging)
  - Another famous class are the Boosted ensembles (e.g. Boosted Decision Trees and Gradient Boosted Trees):
    - A sequence of trees that learn progressively more difficult cases
    - `from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier`

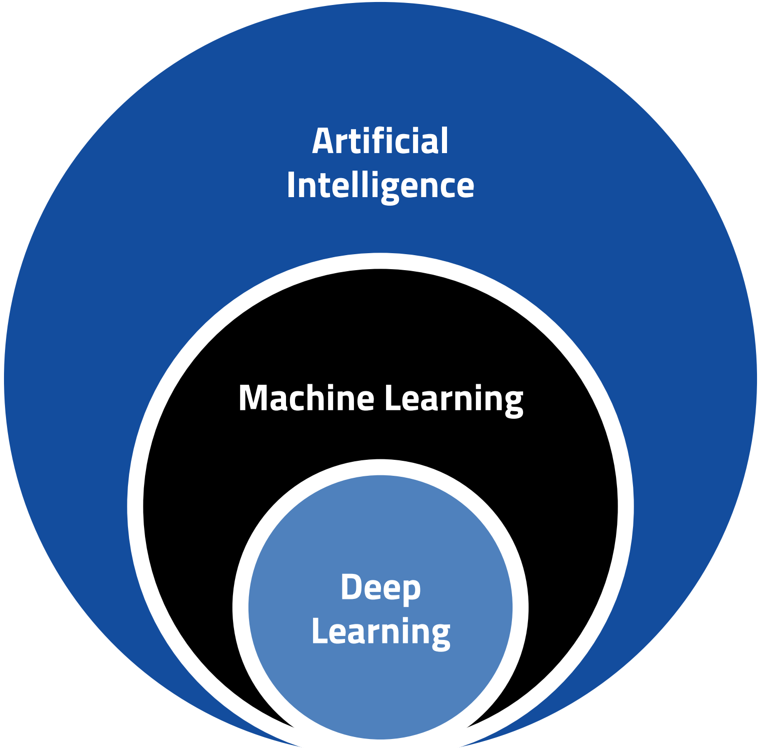
Ensembles
They are better than individual models

- Ensembles of Trees are **very good baseline models** and should be your first go-to choice for tabular data (i.e. excels, csv, etc)
- They improve generalisation of the base estimator and reduce the risk of overfitting
- They **require little to no data preprocessing** (when based on Trees), making them very attractive as out-of-the-box solutions

But trees (and respective ensembles) are too strict

1. They do not perform that well for non-tabular data (images, video, sound, text, etc)
2. Although they provide great supervised models, they lack versatility for other tasks
3. They are not intrinsically compatible with multiclass and multilabel problems
4. etc

Deep Learning is a subclass of Machine Learning algorithms that train Neural Networks to perform tasks



Deep Learning and Neural Networks

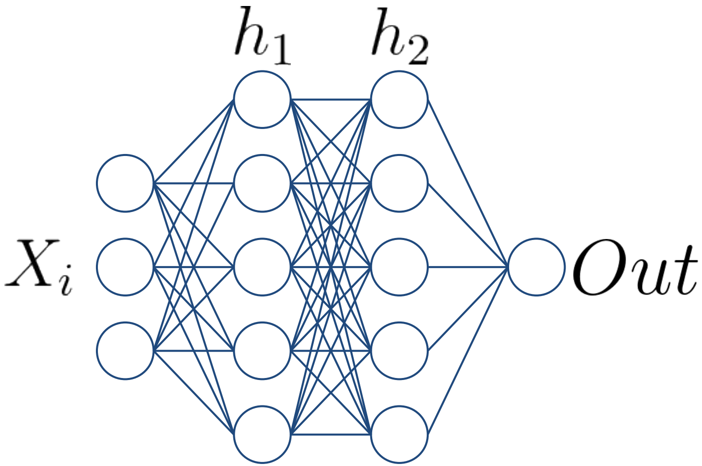
Terrible name, great idea

Differentiable models that can be trained with **Stochastic Gradient Descent**

Unmatched **representational power** and are capable of **feature abstraction**: deeper layers abstract more complex relations

Extremely versatile and can take in **data of many different shapes and formats**

All state-of-the-art Machine Learning applications are based on Deep Learning and implement Neural Networks



$$\vec{h}_1 = a_1(\mathbf{w_1} \cdot \vec{x} + \vec{b}_1)$$

$$\vec{h}_2 = a_2(\mathbf{w_2} \cdot \vec{h}_1 + \vec{b}_2)$$

$$Out = a_{Out}(\vec{w}_{Out} \cdot \vec{h}_2 + b_{Out})$$

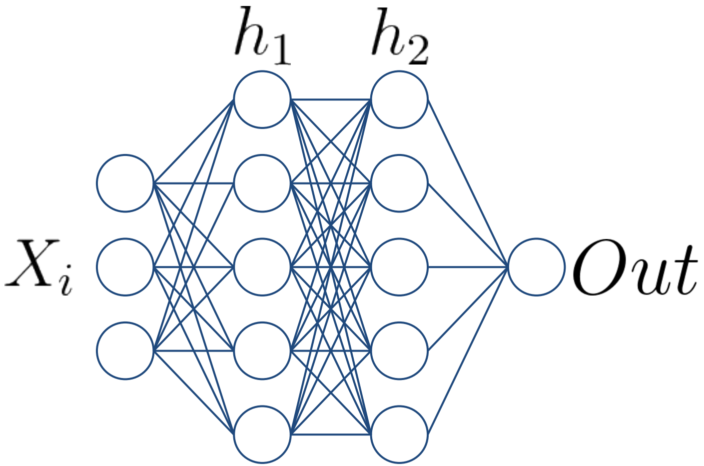
$$a_i = \{\tanh, \sigma, \mathrm{ReLu}, \dots\}$$

$$NN = Out \circ \vec{h}_2 \circ \vec{h}_1$$

Deep Learning and Neural Networks

Defining and training

- Define how many layers and how many units (neurons) are in each layer, in addition to the non-linear activation
- Define the output
  - For binary classification: sigmoid
- Define the Loss function
  - For binary classification: binary cross-entropy
- Iteratively train on mini-batches of data. This is performed by an optimisation algorithm (we won't be able to cover these in detail)



$$\vec{h}_1 = a_1(\mathbf{w_1} \cdot \vec{x} + \vec{b}_1)$$

$$\vec{h}_2 = a_2(\mathbf{w_2} \cdot \vec{h}_1 + \vec{b}_2)$$

$$Out = a_{Out}(\vec{w}_{Out} \cdot \vec{h}_2 + b_{Out})$$

$$a_i = \{\tanh, \sigma, \mathrm{ReLu}, \dots\}$$

$$NN = Out \circ \vec{h}_2 \circ \vec{h}_1$$

Deep Learning and Neural Networks
Preprocessing: Standartisation

- Unlike trees, Neural Networks require some preprocessing
- The most common requirement is to standartise the inputs: **set mean to 0 and standard deviation to 1**

$$X \to \frac{X - \bar{X}}{\sigma_X}$$

- The reason for this is that the SGD applies weight updates layer-by-layer (chain rule over function composition), and too large activations will lead to too large updates => **gradient explosion and unstable learning**
- Scikit-Learn is your friend
  - `from sklearn.preprocessing import StandardScaler`
  - `from sklearn.pipeline import make_pipeline`

Neural Networks
In python

- Scikit-Learn has a simple implementation of a Neural Network for classification (usually called a Multi-Layer Perceptron)
    - `from sklearn.neural_network import MLPClassifier`
- But we will look into a very famous dedicated framework: TensorFlow/Keras

Keras

TensorFlow

Neural Networks
Are the present and the future

- Neural Networks has unleashed a revolution in Machine Learning applications
- Getting them to work requires some work and care, but the outcome is usually worth the trouble
- This is by no means a complete introduction, I recommend investing some time with the Keras documentation https://keras.io/examples/
- But this is not all! Also take a look at **PyTorch** and **Jax**, which might be more suitable to your needs and applications

2nd hands-on

Let’s implement some implement some ensembles and neural networks using both Scikit-Learn and TensorFlow

Neural Networks
In python using TensorFLow/Keras

- We will use Keras packaged with TensorFlow
- A model is initiated with a Model class. We will use the Sequential
  - It takes a sequence of layers (classes from the layers module)
  - It connects them automatically sequentially

```
model = keras.models.Sequential([
    keras.layers.Dense(100, activation='relu', input_shape=(2,)),
    keras.layers.Dense(1, activation='sigmoid')
  ])
```

- You then compile to define the Loss function, metrics, and the optimizer

```
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy', keras.metrics.AUC()])
```

- Which you can then fit

```
model.fit(X_train, y_train, epochs=100)
```

How SGD is implemented. Adam is always a good first choice

Best practices and the Higgs Dataset

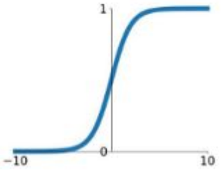
Because you only learn by doing

Model choice and Hyperparameter Tuning
Neural Network shape

- We saw how the shape of the network affects its performance
  - The deeper (more hidden layers) and wider (number of units) the greater is the capacity
- The performance of the Neural Network can also be affected by the choice of non-linear activation function
- How to choose?
- Is there a risk of using too large a network?

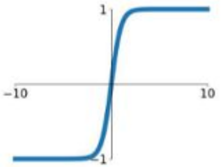
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



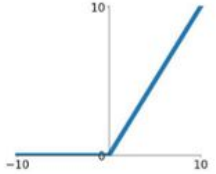
tanh

$\tanh(x)$



ReLU

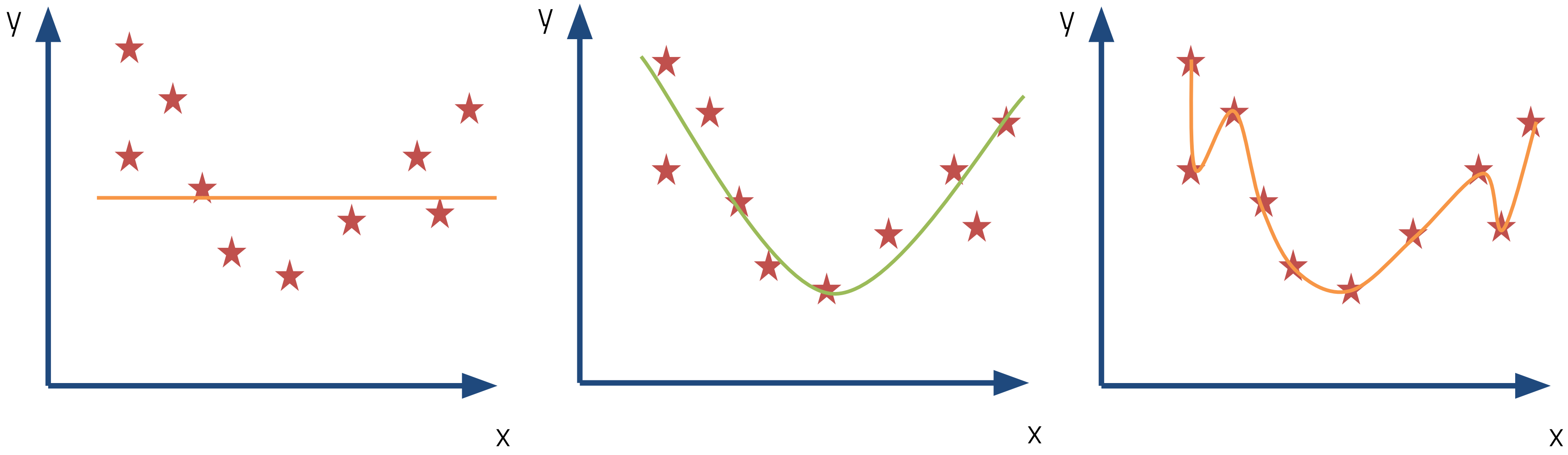
$\max(0, x)$



Model choice and Hyperparameter Tuning
Model Capacity

A model with insufficient capacity will fail to fit f: **underfitting.**

A model with too much capacity will fit the noise: **overfitting.**

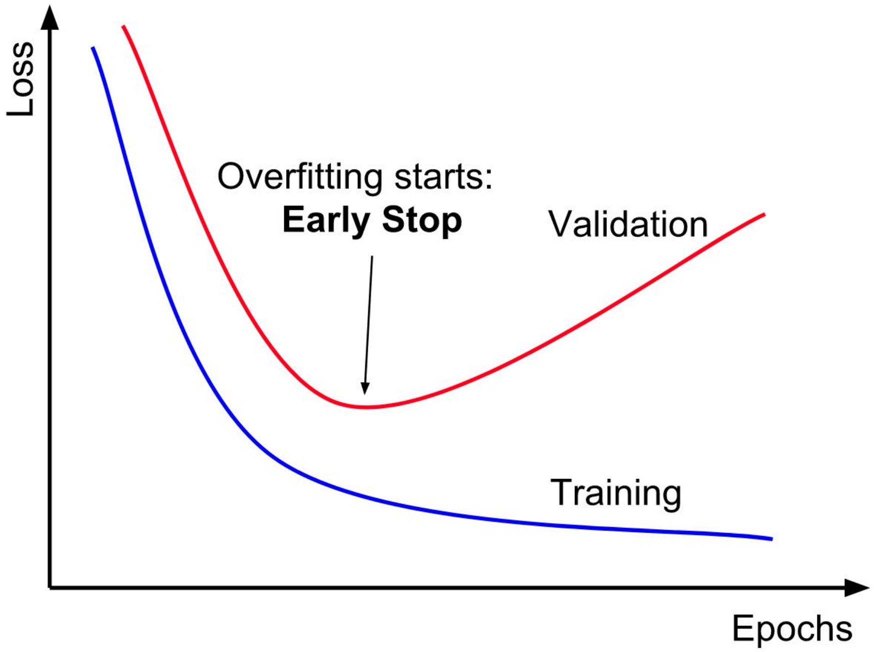


Regularisation

In practice, one usually overestimates the capacity needed and then applies regularisation to prevent overfitting

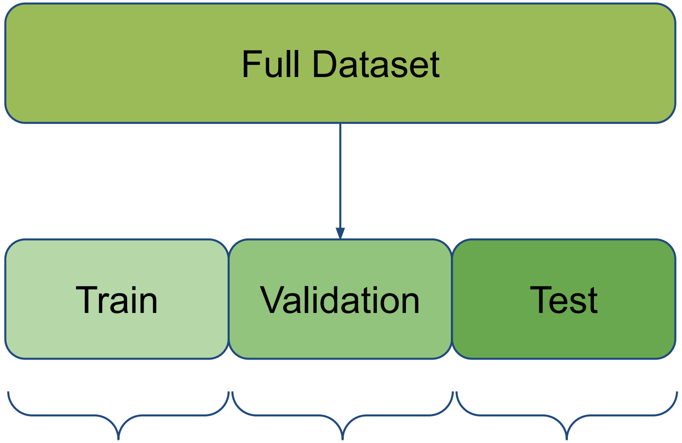
Model choice and Hyperparameter Tuning
Regularisation

- Many ways of regularising a ML model, which depend on the type of algorithm
- One that always helps with Neural Networks (and other iteration-based training algorithms) is **early stop**
    - Stop training when the loss/metric worsens on a validation set



Model choice and Hyperparameter Tuning
Best practices: Three different splits!

- Split the dataset into three sets
  - Train: for fitting
  - Val: for validation
  - Test: to derive the final performance
- **Never use the Test set at any stage of your training or validation => Information Leakage (a.k.a. cheating)**



In our case we want to retain a good statistical description of our data
1:1:1

Model choice and Hyperparameter Tuning
Choosing the final hyperparameters

- Try different combinations of hyperparameters. For each:
    - Train the network with the training set
    - Use the validation set to stop early
    - Measure the metrics on the validation set
- In the end: pick the hyperparameter combination with the best validation set metrics
- If you learn how to do this you can become a professional Machine Learning engineer in the industry

Machine Learning in New Physics Analyses

Finding a needle in a particle haystack

- Now that you are proficient Machine Learning engineers, let's do some physics with this!
- The idea is simple:
  - Data come
  - Data might have a signal we want to discover
  - Train a classifier to separate interesting events from the background
  - Make a discovery and profit (joking, someone else gets the Noble)

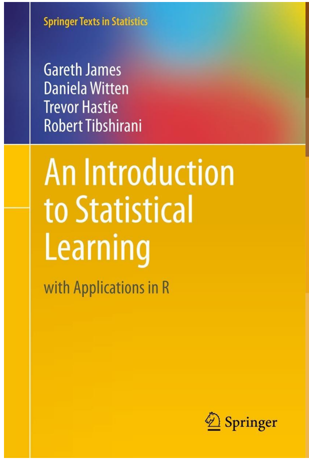
The Higgs Dataset

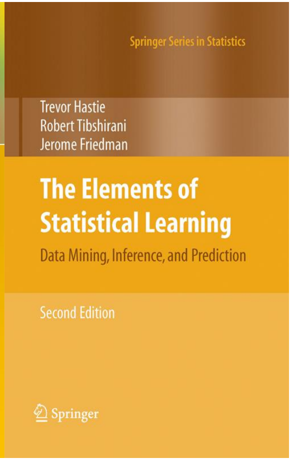
- Created in 2014 under the HiggsML challenge hosted by Kaggle
  https://higgsml.lal.in2p3.fr/
- The dataset is composed of pseudo-data (generated) Higgs (Signal) and other Standard-Model events (Background)
- The objective is to isolate as much signal as possible (Classification problem)
  - https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf

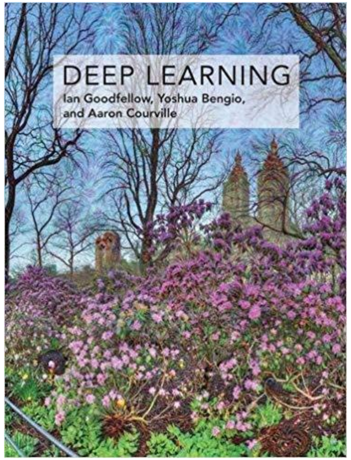
Further resources

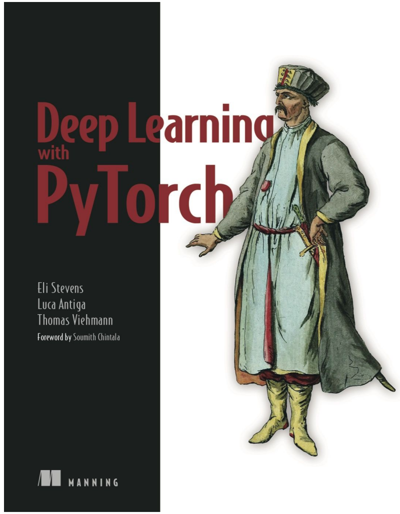
Some of them are free

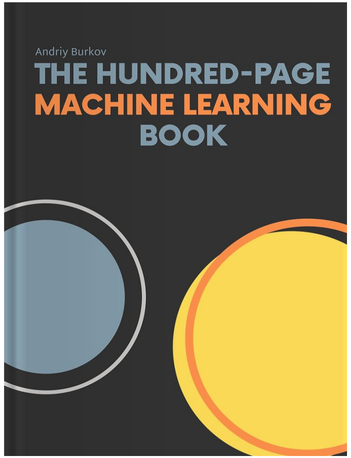
These are free



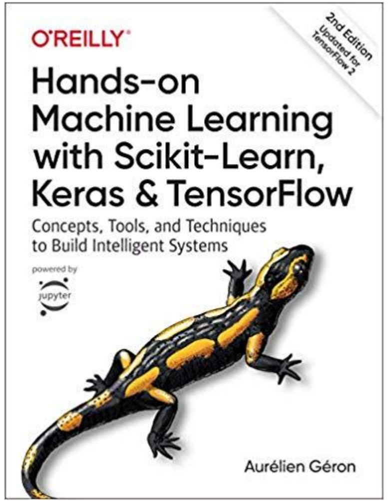


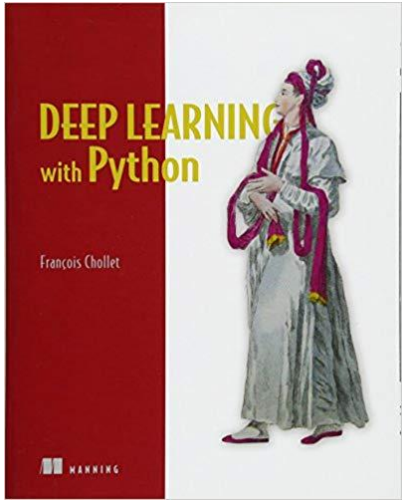


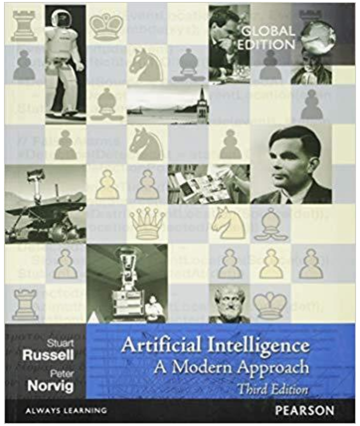


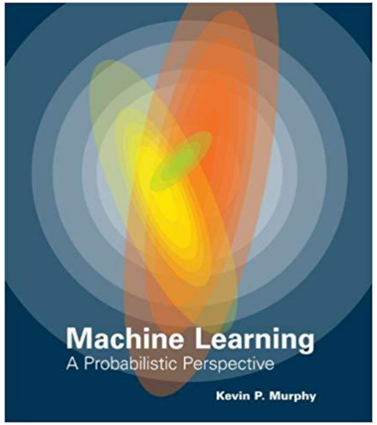


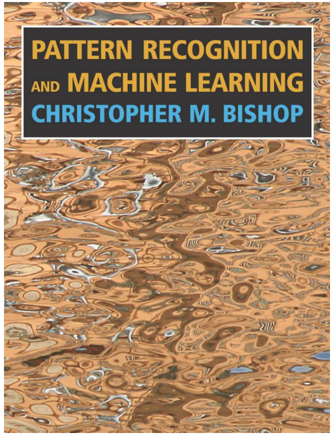
Not free, but very good

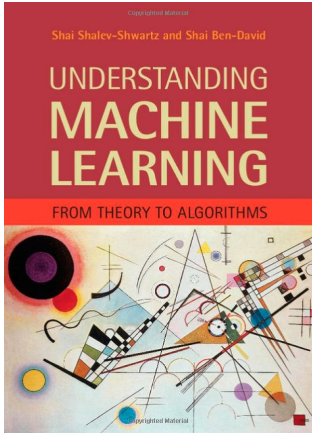












ML@LIP

For those interested in working on these things

ML@LIP

- There's a wide range of ML applications across the many groups at LIP
- I'm involved in applications that cover QCD pheno (Liliana's talk), BSM searches (Nuno's talk, Rute's tutorial), and BSM pheno/model building
  - We have many ongoing projects suitable for BSc, MSc and PhD aspiring students
  - Drop me a line if you are considering pursuing your studies/research in HEP using ML

Thanks!

I hope this was useful

mcromao@lip.pt

3rd hands-on

Let’s do some physics with all this malarkey!