Machine Learning for Applied Physics and High Energy Physics

Lecture 1 An Introduction to Machine Learning

1.1 PRESENTATION OF THE COURSE

48 hours of lectures (6 CFU) of which

24 hours read by Dr. Nocera (12+12)

24 hours read by Prof. Badger (24)

Timetable: Mon. 9-11 am; Tue 2-4 pm always in AULA D

Check whether changes are possible to avoid overlap.

Goals: Provide students with the specific knowledge to understand modern numerical techniques used in scientific and technological innovation. Make students familiar with the language, the algorithms and the software used by Machine Learning in fundamental and applied research.

Pre-requisites: Knowledge of python (v 3.9 or higher), in particular of the numpy and scipy libraries. Familiarity with the concepts of functional and object programming.

Expected learning goals:

- Describe the ML techniques used in solving complex physical problems; identify the features of each technique and their appropriateness for solving different problems; apply specialised software (e.g. Tensorflow/Keras) to the solution of physical problems.
- Relate a physical problem with Machine Learning techniques appropriate to its resolution; evaluate their effectiveness.
- Use the language of ML appropriately and precisely.

Program: see CAMPUSNET; the program is flexible.

Reading: see CAMPUSNET. A living review is provided with papers from which one can take inspiration.

Examination: ORAL, see CAMPUSNET for details. The course and the examination will be in English.

Bibliography:

- Living Review
- ~~Hastie~~, Tibshirani, Friedman, "The Elements of Statistical Learning: Data mining, Inference and Prediction", Springer (2018) 2nd Ed.
- James, Witten, Hastie, Tibshirani, "An Introduction to Statistical Learning" Springer (2021) 2nd Ed.
- Murphy, "Probabilistic Machine Learning: An Introduction", MIT Press (2022) 1st Ed.

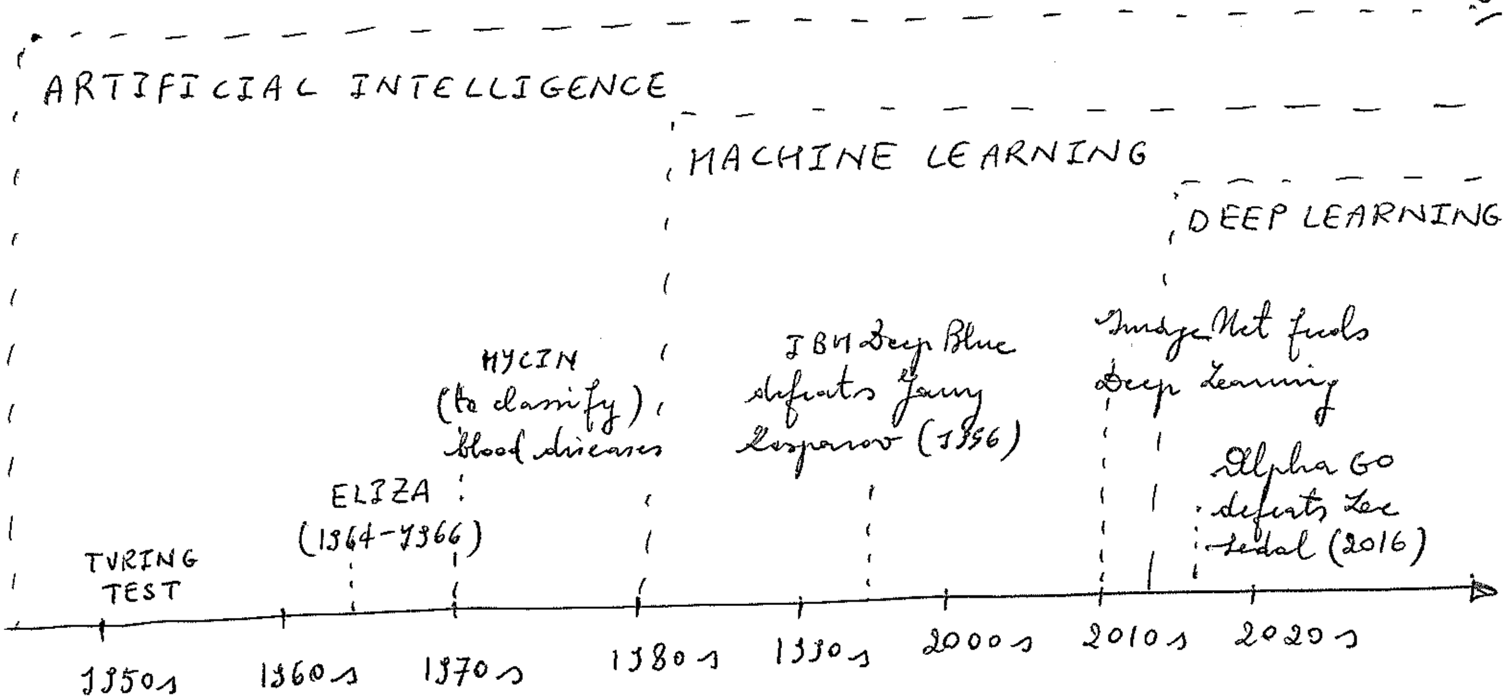
QUESTIONS?

1.2. WHAT IS MACHINE LEARNING?

T. Mitchell (1997)

"A computer program is said to learn from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

The term Machine Learning was introduced by Arthur Samuel in 1959 in the context of computer gaming.



The course presents Machine Learning from a probabilistic point of view (statistical learning). All observable quantities $\vec{x}$, which are estimated and/or predicted, are seen as a realisation of a random variable X, to which we associate a probability distribution. Given a model, denoted by a set of parameters $\vec{\vartheta}$, we seek the probability of observing $\vec{x}$ given $\vec{\vartheta}$, i.e. $p(\vec{x}/\vec{\vartheta})$. The aim of ML is to fit the $\vec{\vartheta}$ to some $\hat{\vec{\vartheta}}$, that is the set of parameters of the model that best predict $\vec{x}$.

Depending on the nature of T, P, and E, ML is typically divided in three main families.

$\boxed{A}$ SUPERVISED LEARNING

$\boxed{B}$ UNSUPERVISED LEARNING

$\boxed{C}$ REINFORCEMENT LEARNING

The course focuses on $\boxed{A}$ and $\boxed{B}$.

1.3. SUPERVISED LEARNING

The task T is to learn a mapping f from inputs $\vec{x} \in \mathcal{X}$ to outputs $\vec{y} \in \mathcal{Y}$. The inputs $\vec{x}$ are called FEATURES, COVARIATES or PREDICTORS; the input consists of a fixed-dimensional vector of numbers, such as the pixels of an image. In this case, $\mathcal{X} = \mathbb{R}^D$, where D is the dimensionality of the vector. The outputs $\vec{y}$ are called LABELS, TARGETS, or RESPONSES. The experience E is given in the form of a set of N input-output pairs $\mathcal{D} = \{(\vec{x}_n, \vec{y}_n)\}_{n=1}^{N}$, known as TRAINING SET; N is called SAMPLE SIZE. The performance P depends on the type of output we are predicting, as we discuss below.

1.3.1. CLASSIFICATION

In classification problems, the output space is a set of C unordered and mutually exclusive labels known as classes $\mathcal{Y} = \{1, 2, \dots, C\}$. The problem of predicting the class label given an input is also called PATTERN RECOGNITION. If there are just two classes, often denoted by $y \in \{0, 1\}$ or $y \in \{-1, +1\}$, the problem is called binary classification.

<u>Example</u>: Classify observations of space taken by the SDSS (Sloan Digital Sky Survey) [2112.02026] into one of three classes: star, galaxy, or quasar (a luminous supermassive black hole). In general, these observations come in the form of PHOTOGRAPHS, therefore

$$\mathcal{X} = \mathbb{R}^D, \quad D = C \times D_1 \times D_2 \quad C = 3 \text{ channels (RGB)}$$

$$f : \mathcal{X} \to \mathcal{Y} \quad \text{(is the photograph depicting a star, a galaxy or a quasar?)}$$

Astronomers / Astrophysicists have identified some relevant features which simplify the problem. These are: ascension and declination (ascension tells how far left or right the object is in the celestial sphere)(declination tells how far up or down the object is in the celestial sphere); filters of photometric system (the colours of the system used to measure the brightness of the emitted light); the redshift (the shift in the light wavelength due to GR effects). Therefore $X = \mathbb{R}^{2+5+1=8}$.

2: ascension and declination; 5: filters: u, g, r, i, z; 1: redshift

The training set is a collection of 100'000 examples of the three classes

index	ascension	declination	filters u, g, r, i, z	redshift	label
0	⋮	⋮	⋮	⋮	STAR
1	⋮	⋮	⋮	⋮	GALAXY
⋮					⋮
100	⋮	⋮	⋮	⋮	GALAXY
⋮					⋮
100'000				⋮	QUASAR

D

TABULAR DATA (OR DESIGN MATRIX) $\begin{cases} \text{BIG DATA: } N \gg D \\ \text{WIDE DATA: } D \gg N \end{cases}$

It is a good idea to explore the data (EXPLORATORY DATA ANALYSIS) to see if there are patterns that are obvious, for instance box plots or pair plots. This is useful for DIMENSIONALITY REDUCTION.

$$f(\vec{x}, \vec{\vartheta}) = \begin{cases} \text{STAR} & \text{if redshift} = 0 \\ \text{GALAXY or QUASAR} & \text{otherwise} \end{cases} \qquad \begin{cases} \text{GALAXY} & \text{if } \ell_{iz} = 1 \\ \text{QUASAR} & \text{otherwise} \end{cases}$$

$\vec{\vartheta}$: THRESHOLD PARAMETERS

DECISION TREE

The goal of supervised learning is to automatically come up with classification models to reliably PREDICT the labels for any given input. A common way to measure performance on this task is in terms of MISCLASSIFICATION RATE on the training set:

$$\mathcal{L}(\vec{\vartheta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left(y_n \neq f(\vec{x}_n; \vec{\vartheta})\right) \quad (\mathcal{L} = \text{LOSS})$$

where

$$\mathbb{I}(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

Some errors may be more costly than others, for instance suppose to misclassify a GALAXY: you may miss discovery. We can then define the EMPIRICAL RISK

$$\mathcal{L}(\vec{\vartheta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \ell\left(y_n, f(\vec{x}_n, \vec{\vartheta})\right)$$

where ℓ is an asymmetric loss function ($\ell_{01}(y, \hat{y}) = \mathbb{I}(y \neq \hat{y})$)

One way to define the problem of model fitting or training is to find a setting of the parameters that minimises the empirical risk on the training set

$$\hat{\vec{\vartheta}} = \underset{\vec{\vartheta}}{\operatorname{argmin}}\, \mathcal{L}(\vec{\vartheta}) = \underset{\vec{\vartheta}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \ell\left(y_n, f(\vec{x}_n; \vec{\vartheta})\right)$$

This is called EMPIRICAL RISK MINIMISATION. However the true goal of ML is to minimise the empirical risk on unseen samples. That is, we want to GENERALISE (or PREDICT), not just ~~obtain~~ ESTIMATE (or POSTDICT).

When fitting probabilistic models, it is common to use the log probability as loss function

$$\ell\left(y, f(\vec{x}, \vec{\vartheta})\right) = -\ln p\left(y / f(\vec{x}, \vec{\vartheta})\right)$$

The reason (in short) is that a good model (with low loss) is one that assigns a high probability to the true output $\vec{y}$ for each corresponding input $\vec{x}$. The average negative log probability of the training set is given by

$$NLL(\vec{\vartheta}) = -\frac{1}{N}\sum_{n=1}^{N} \ln p(y_n / f(\vec{x}_n; \vec{\vartheta}))$$ NEGATIVE LOG-LIKELIHOOD

If we minimise the NLL, we compute

$$\hat{\vec{\vartheta}}_{MLE} = \underset{\vec{\vartheta}}{\operatorname{argmin}}\, NLL(\vec{\vartheta})$$ MAXIMUM LIKELIHOOD ESTIMATE

1.3.2 REGRESSION

In a regression problem, the space of outputs is $y \in \mathbb{R}$
Example: y could be the brightness of a luminous body in the sky. Regression is very similar to classification. However, because the output is real-valued, we need to use a different loss function. A usual choice is the QUADRATIC LOSS (also called l_2):

$$l_2(y, \hat{y}) = (y - \hat{y})^2$$

This penalises large residuals $(y - \hat{y})$ more than small ones. The empirical risk when using quadratic loss is equal to the mean squared error or MSE

$$MSE(\vec{\vartheta}) = \frac{1}{N}\sum_{n=1}^{N} (y_n - f(\vec{x}_n, \vec{\vartheta}))^2$$

In regression problems, it is common to assume the output distribution to be a Gaussian distribution:

$$\mathcal{N}(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \quad \begin{cases} \mu & \text{MEAN} \\ \sigma^2 & \text{VARIANCE} \end{cases}$$

In the context of regression, we can make μ depend on the inputs : $\hat{\mu} = f(\vec{x}_n; \vec{\vartheta})$. Therefore

$$p(y/\vec{x}; \vec{\vartheta}) = N(y \mid f(\vec{x}, \vec{\vartheta}); \sigma^2)$$

If we assume, for the sake of simplicity, that σ^2 is fixed then the corresponding average (per sample) NLL is

$$NLL(\vec{\vartheta}) = -\frac{1}{N} \sum_{n=1}^{N} \ln \left[\left(\frac{1}{2\pi\sigma^2} \right)^{1/2} \exp \left\{ -\frac{1}{2\sigma^2} (y_n - f(\vec{x}_n; \vec{\vartheta}))^2 \right\} \right]$$

$$= \frac{1}{2\sigma^2} MSE(\vec{\vartheta}) + \text{const}$$

Therefore, the MLE is obtained by minimising l_2. This is called χ^2 minimisation.

1.3.2.1 LINEAR REGRESSION

$$f(x, \vec{\vartheta}) = \underset{\text{slope}}{b} + \underset{\text{offset}}{w} x \qquad \vec{\vartheta} = (w, b)$$

$$\hat{\vartheta} = \underset{\vec{\vartheta}}{\operatorname{argmin}} \, MSE(\vec{\vartheta}) \qquad \text{(LEAST SQUARE SOLUTION)}$$

In general $f(\vec{x}, \vec{\vartheta}) = b + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D = b + \vec{w}^T \cdot \vec{x}$

(MULTIPLE LINEAR REGRESSION)

1.3.2.2 POLYNOMIAL REGRESSION

$f(x, \vec{w}) = \vec{w}^T \Phi(x)$ with $\Phi(x) \doteq [1, x, x^2, \ldots, x^D]$

If $D = N-1$, then we talk of INTERPOLATION ($MSE = 0$)

$$f(\vec{x}; \vec{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + \ldots$$

1.3.2.3 NEURAL NETWORKS

$f(\vec{x}, \vec{w}, \vec{V}) = \vec{w}^T \Phi(\vec{x}, \vec{V})$ with $\vec{V}$ a set of parameters for $\Phi(x)$

$f(\vec{x}, \vec{\vartheta}) = f_L(f_{L-1}(\cdots(f_1(\vec{x}))\cdots))$ with $f_l(\vec{x}) = \vec{w}_l^T f_{l-1}(\vec{x})$

1.3.2.4 OVERFITTING AND GENERALISATION

We can rewrite the empirical risk as

$$\mathcal{L}(\vec{\vartheta}, \mathcal{D}_{train}) = \frac{1}{|\mathcal{D}_{train}|} \sum_{(\vec{x},\vec{y}) \in \mathcal{D}_{train}} \ell(\vec{y}, f(\vec{x}, \vec{\vartheta}))$$

where $\mathcal{D}_{train}$ is the subset of $\mathcal{D}$ on which the model is trained. If $\mathcal{L}(\vec{\vartheta}, \mathcal{D}_{train})$ is such that MSE = 0, we talk of OVERFITTING. To detect overfitting, let us assume to know the true (BUT unknown) distribution $p^*(\vec{x}, \vec{y})$ used to generate the training set. Then we compute the theoretical expected loss (or POPULATION RISK)

$$\mathcal{L}(\vec{\vartheta}, p^*) \equiv \mathbb{E}_{p^*(\vec{x},\vec{y})} \left| \ell(\vec{y}, f(\vec{x}, \vec{\vartheta})) \right|$$

The difference $\mathcal{L}(\vec{\vartheta}, p^*) - \mathcal{L}(\vec{\vartheta}, \mathcal{D}_{train})$ is called GENERALISATION GAP. If it is large, then the model overfits. In practice, we do not know p^*. Therefore we can partition the data into two subsets (TRAINING and VALIDATION). Then we can approximate the population risk using the test risk

$$\mathcal{L}(\vec{\vartheta}, \mathcal{D}_{test}) \triangleq \frac{1}{|\mathcal{D}_{test}|} \sum_{(\vec{x},\vec{y}) \in \mathcal{D}_{test}} \ell(y, f(\vec{x}, \vec{\vartheta}))$$

We pick the model with minimum loss on the validation set. Actually, a data set is divided into three subsets: TRAINING and VALIDATION (to optimise the model) and TEST (unseen, used to test the generalisation power of the model)

NO FREE LUNCH THEOREM: "All models are wrong, but some are useful" (George Box, 1987). There is no single best model that works optimally for all kinds of problems.

1.4 UNSUPERVISED LEARNING

The task T consists of finding features of the inputs $\vec{x} \in \mathcal{X}$ without knowledge of the outputs $\vec{y}$. In probabilistic terms, we aim to fit an unconditional model $p(\vec{x})$ (conversely, in supervised learning we fit $p(\vec{y}/\vec{x})$).

Differences between SL and UL

- UL: no need to collect large training sets
- UL: no need to partition $\mathcal{X}$
- UL: the model must "explain" the inputs

1.4.1 CLUSTERING and LATENT FACTORS

The problem consists in finding partitions of $\mathcal{X}$ that group similar $\vec{x}$. Example: cluster of luminous bodies in the sky, or cluster of stars into galaxies. The problem consists in projecting a high-dimensional data set into low-dimensional features. Example: Let us suppose that $x_n \in \mathbb{R}^D$ is generated by low-dimensional latent factors $z_n \in \mathbb{R}^K$: $z_n \to x_n$

Assuming a Gaussian prior:

$$p(\vec{x}_n / \vec{z}_n; \vec{\theta}) = \mathcal{N}(\vec{x}_n / \vec{W} z_n + \vec{\mu}, \vec{\Sigma})$$

FACTOR ANALYSIS

if $\Sigma = \sigma^2 \mathbb{1}$, PCA (principal component analysis)

if the model is non-linear, we talk of VARIATIONAL AUTOENCODERS

The goodness of the model is evaluated by measuring the probability assigned by the model on unseen samples

$$\mathcal{L}(\vec{\theta}, \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(\vec{x}; \vec{\theta})$$

A good model won't be surprised by samples that obey the model

1.5 REINFORCEMENT LEARNING

The system (or AGENT) has to learn how to interact with the environment. This can be encoded by means of a policy $a = \pi(x)$ which specifies which action to take in response to each possible input $\vec{x}$. The agent receives occasional rewards for its actions.