Artificial Intelligence/Machine Learning/Deep Learning: 'Bridging the Skills Gap'

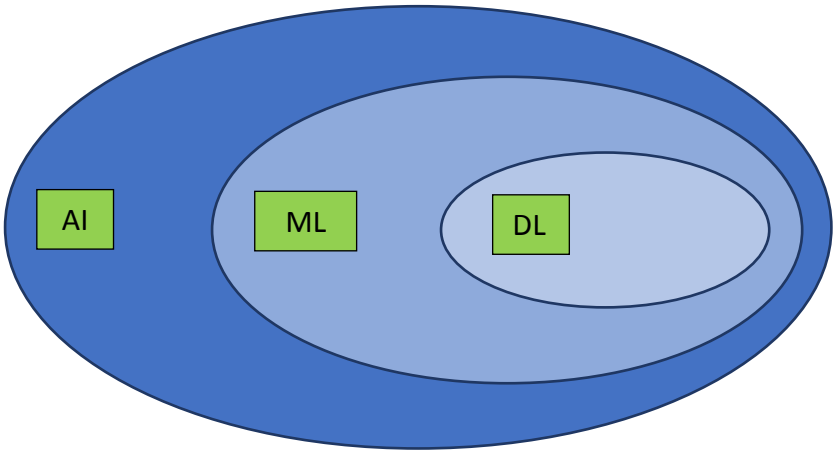
Lesson 1: History, Definitions and Basic Concepts
Hello, welcome back to my course on AI and Machine Learning

Today we will address following items:

1. What is the difference between AI, Machine Learning (ML) and Deep Learning (DL) ?
2. History of AI/ML/DL
3. Definitions & Basic concepts of ML/DL (Loss/Cost Function, Mean Squared Error (MSE), Regression, Classification, Supervised Learning, Unsupervised Learning)

What is the difference between AI/ML and DL?

Last week we saw that AI (and ML/DL) are hyped under the digitalization umbrella to the extend, that people believe that their jobs will be taken over by robots in a few years. The euphoria/fear comes from the lack of knowledge about the subject.



AI has been around for a while and can be defined as the '*automation of thought*'…for example a calculator and an xls spreadsheet can be considered as AI

ML: addresses the topic of a computer learning the rules of a system while looking at mere inputs and outputs. In **1957, Frank Rosenblatt** invented the Perceptron**,** the simplest form of a Neural Network (NN). The Perceptron created a lot of excitement…but this changed in 1969.

In **1969**, **Minsky and Papert** published a book highlighting the perceptron's limitations and that Frank Rosenblatt's predictions had been grossly exaggerated. For example, the Perceptron was not able to learn a XOR (exclusive OR) function. A XOR function is a logical operator that outputs TRUE if the 2 inputs differ. Research funding dried up and this led to the **first AI** (or **NN**) **winter**

1986: **Rumelhart and Hinton** solved the XOR problem and showed that backpropagation (training of NN) works → this was the basis of all subsequent NN and DL progress.

In **1989**, Yann LeCun (Bell Labs) combined the ideas of convolutional neural networks (CNNs) and backpropagation and applied them to classifying handwritten digits. The resulting network named **LeNet** was used by the US Postal Service to automate reading ZIP codes on mail envelops.

NNs however faced another **winter** in the **1990s and the early 2000s** mainly because of the popularity of Kernel Methods. Kernel Methods are a group of classification algorithms, the best known of which is the Support Vector Machine (SVM). In fact, in the early 2000s, most NN papers were rejected at AI conferences. NN were still considered too expensive to train.

DL: DL is a subset of ML that puts emphasis on learning **successive layers** of increasingly meaningful representations. In **2006, Geoffrey Hinton** launched the term **'Deep Learning'** to explain new algorithms that let computers 'see' and distinguish objects and text in images and videos. Suddenly all other ML algorithms were branded as shallow learning.
The rebranding trick and the technological developments of GPUs revived research with respect to NNs.

AlexNet (2012): Alex Krizhevsky (advised by Hinton) created a broad and deep CNN that won the 2012 ImageNet competition (image classification competition: 1.4M high resolution color images into 1000 categories). 2012 marked the first year where a CNN was used to achieve a top 5 test error rate of 15.4%.

Since then many enhancements where developed to make NNs perform better and solve new problems. By 2015, the winner reached an error rate of 3.6%. NNs have completely replaced SVMs and Decision Trees in a wide range of applications. In 2016 and 2017, Kaggle was dominated by gradient boosting machines (where structured data is available) and DL (used for perceptual problems such as image classification). Most popular libraries are XGBoost library and keras.

Most industries do not need or use DL. DL is not always the right tool for the job. Today however, for image classification, speech recognition or machine translation, nobody would dream of trying to do it without NNs.

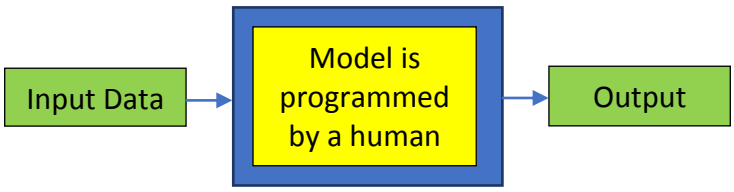
Are we close to another AI winter? Unlikely because:

1. AI is still not central to the way we work, think and live
2. AI research has benefited from huge investments over the past 10 years and most of the research findings of DL have not been applied.

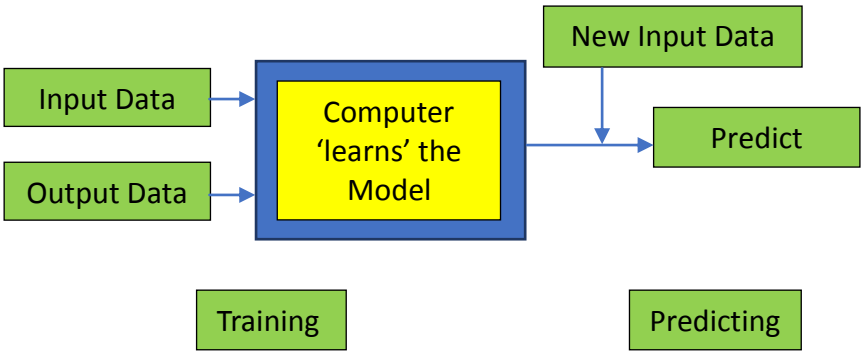
In **March 2019**, Hinton, LeCunn and Bengio won the Turing Award for their contribution to deep learning: https://www.nytimes.com/2019/03/27/technology/turing-award-ai.html

Definitions & Basic concepts:

Pre-ML:



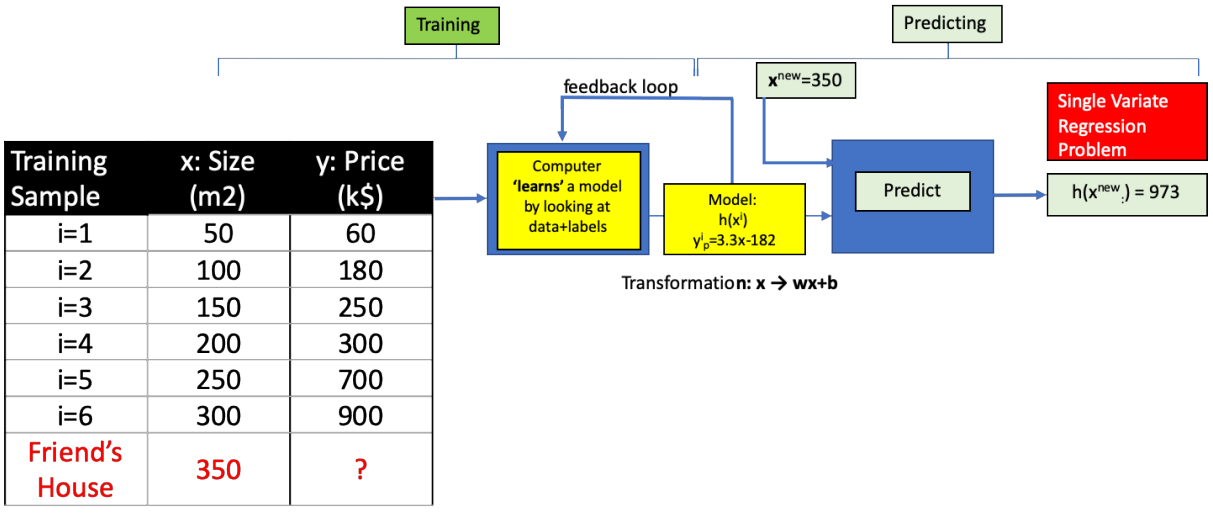
ML:



Example: Supervised Learning - Regression Problem

Algorithm that predicts the value (k$) of a house in a specific city given its size (m^2)

- x^i is the size of (house)^i (m^2)
- y^i is the price of the (house)^i (k$)
- m: number of training samples - m=6
- $h(x^i)=y^i_p$ hypothesis or prediction – a function learned during the training phase allowing us to predict the price of any house given its size.

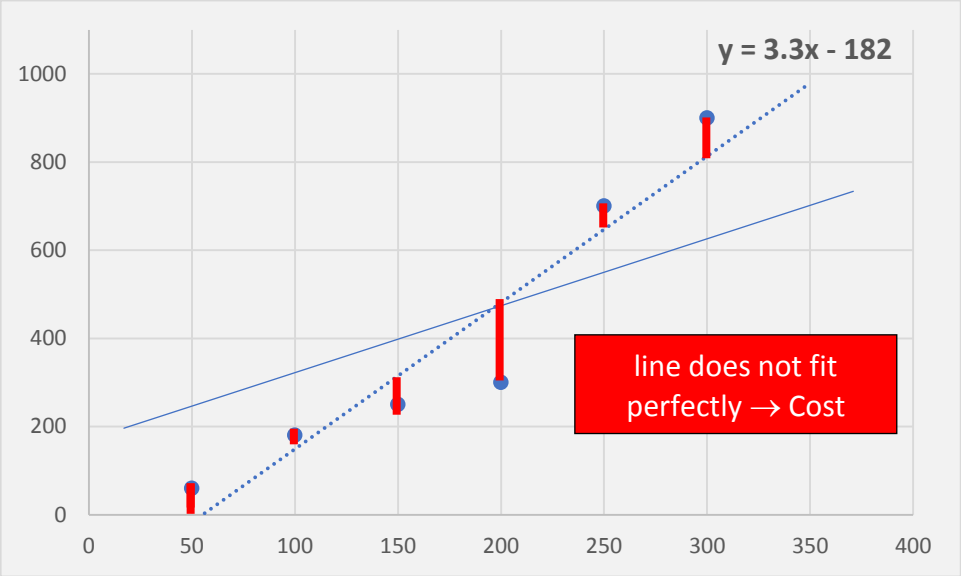


Training Sample	x: Size (m2)	y: Price (k$)
i=1	50	60
i=2	100	180
i=3	150	250
i=4	200	300
i=5	250	700
i=6	300	900
Friend's House	350	?

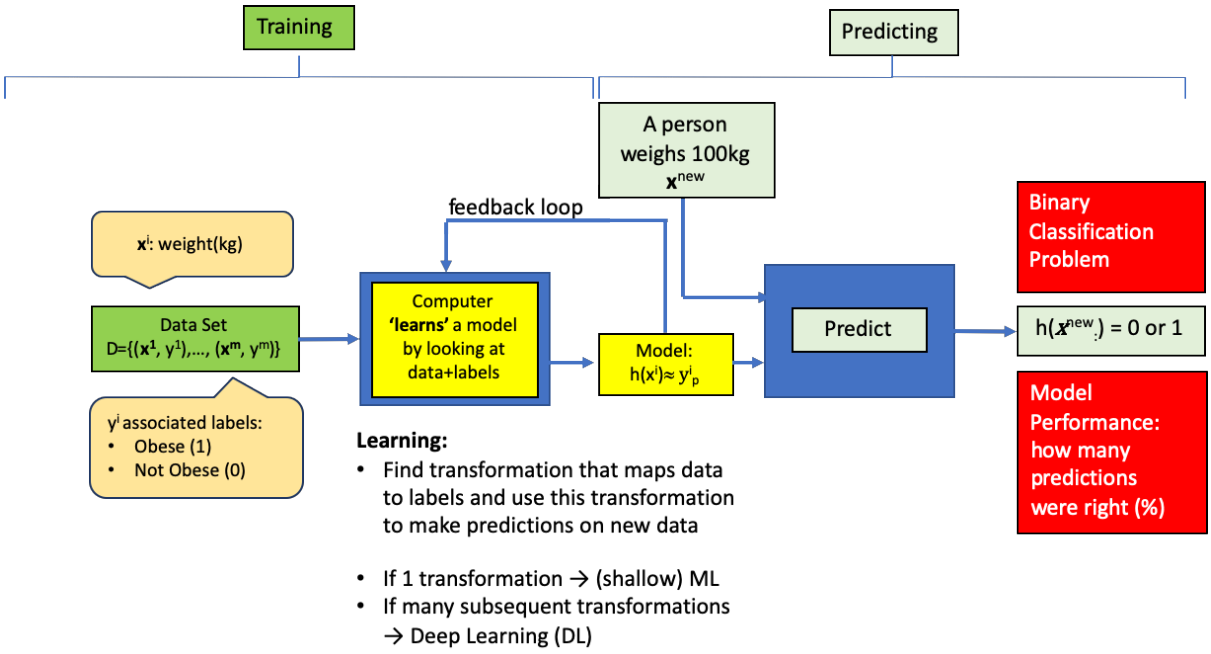
In order to measure the performance of the computer's model we introduce a **Cost Function C** indicating how far the output is from what we expect. ML learning aims at using the loss as a feedback signal and adjust the weights of the model parameters (w and b) in order to minimize the loss ($y-y_p$) → **minimization problem** → **differentiation problem** → **function must be continuous and smooth!**

Cost Function C: **Mean Squared Error (MSE)**

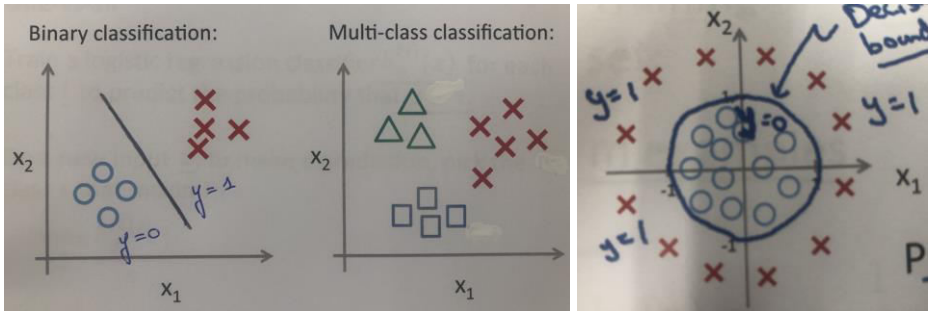
$C(w,b) = \frac{1}{6}\sum_{i=1}^{6}(y_p^i - y^i)$ with y_p the predicted housing price and y the actual housing price



Example: Supervised Learning - Classification Problem



A Classification Problem is an algorithm that takes as input the (x_1,x_2) coordinates of a point and outputs whether the point is to be classified as O or X. The performance of the algorithm can be defined as the % of the points that are correctly classified. In reality there are more potential performance KPIs that we will discuss later.



- Binary Classification: we only have 2 classes: y={0,1} equivalent to {Obese, not Obese}
- Multi-Class Classification (Softmax) where y={1, 2,…, k} → example handwritten digits classification 0→9

A boundary can also be non-linear

The ML learning classification problem can be defined as follows:

- $\mathbf{x}^i$: feature (weight of a person in this case)
- ($\mathbf{x}^i$,y^i): training data set – i-th sample
- Y^i label (or class) – in our case Obese (y=1) or not Obese (y=0)
- $h(x^i)=y^i_p$ hypothesis or prediction – a function (probability distribution) learned during the training phase allowing us to predict the label y^{new} of $\mathbf{x}^{new}$, not part of the training data.

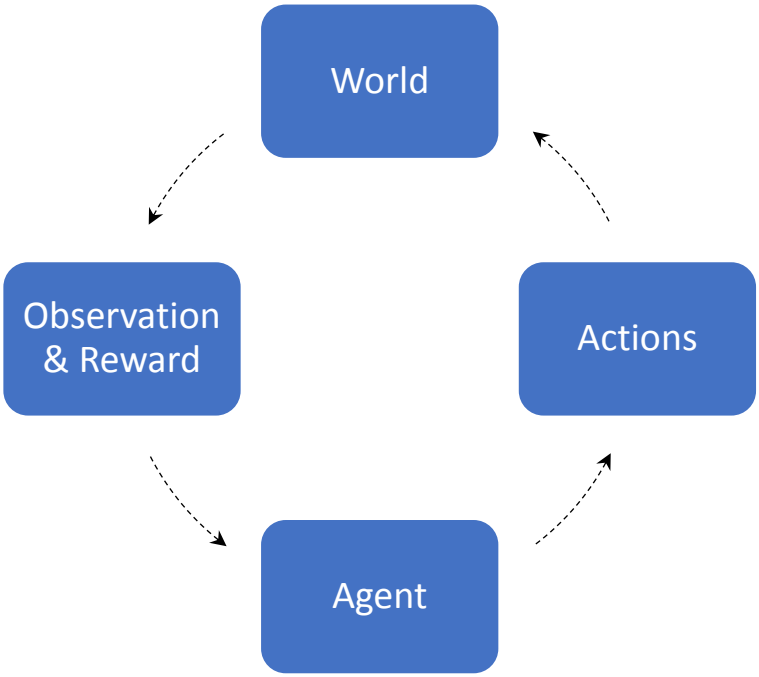
The examples that we discussed are examples of supervised learning because we fed the computer with features and the associated label. In supervised learning, we teach the machine 'by example'. We feed the machine with examples so that the machine can learn patterns in these examples. Once the patters are understood by the machine, we can have the machine apply the patters learned on new data.

Another set of algorithms, like clustering (K-Means), uses unsupervised learning. The computer is fed with unlabeled data. Based on a few parameters you ask the computer to find some structure in the data.

In reinforcement learning (RL), we let the machine (an agent) learn from experience as it observes 'a world'. In RL, data scientists need to design the algorithm, the world in which the agent operates and the reward structure. The machine's goal is to select a sequence of actions that maximizes the reward.

RL brings its own set of challenges such as:

1. How to design a reward structure that takes into account future rewards and avoid short term thinking? One option is to work with 'discounted' rewards
2. How can we let the agent learn from experience without compromising safety? It is unthinkable that we would let an autonomous vehicle 'gain experience' from operating on a public road. In this case, simulation becomes key.
3. Do we allocate rewards along the process of 'gaining experience' or do we only allocated a reward at the end when the mission is completed successfully?



Currently, reinforcement learning is mostly a research area.

Next session: Lesson 2
We will dedicate next session to the most widely used optimization technique: Gradient Descent! It makes sense to look at the math refresher on Calculus and Linear Algebra before embarking on Gradient Descent. Topics that will be covered are:

1. Gradient Descent, Multivariate Gradient Descent, Stochastic Gradient Descent, Mini Batch Stochastic Gradient Descent,
2. Optimizers: Momentum, RMSprop, Adam, Adagrad, Newton's method