Introduction to Artificial Intelligence

History, applications & theoretical approach

Clément Vaes

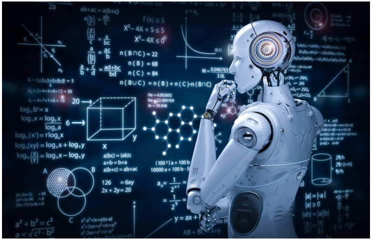
28th April 2022

HeadMind Partners

NYU

Artificial Intelligence?

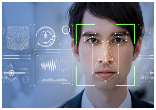
A robot who thinks?
A robot that mimics human behavior?



Artificial intelligence does not exist!

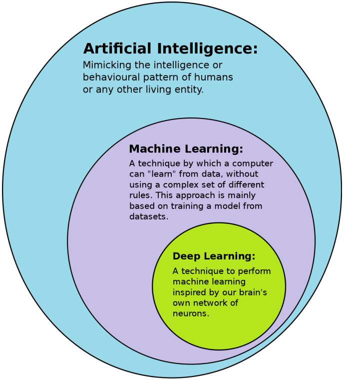
Real AI!

- **General purpose AI** is incredibly hard to make
  - Human brain appears to have lots of special and general functions, integrated in some amazing way that we really do not understand (yet)
- **Specific purpose AI** is more doable
  - For instance, chess/poker/Go playing programs, speech & image recognition



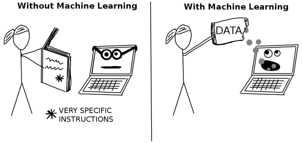
Key dates of AI

- In the 1950s: beginning of **Artificial Intelligence**
- In the 1980s: success of expert systems
- In the 1990s: introduction of **Machine Learning** algorithms
- In the 1990s: exponential development of computer performances
- In the 2010s: introduction of **Deep Learning** algorithms
- In 2010+: many business applications of AI appear



Let's talk about Machine Learning

Machine learning is about giving the ability to a machine to learn without explicitly coding it.

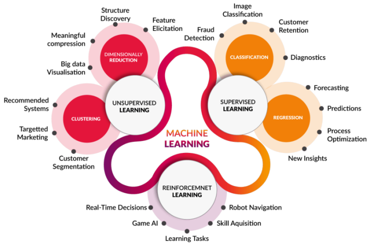


Allows to:

- Uncover behaviors from large datasets;
- Automate certain tasks that still require some intelligence;

There are 3 main types of machine learning:

- **Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**



Machine Learning and function approximation

Most ML methods **approximate a function** via a learning algorithm.

Core concept:

- We have n records of **features** $\{X_1, \ldots, X_k\} \in \{S_1 \times \ldots \times S_k\}$.
- For each of these records, we have a set of **target variables** $\{Y_1, \ldots, Y_l\} \in \{N_1 \times \ldots \times N_l\}$.
- We suppose there exists a function $\mathbf{f} : \{S_1 \times \ldots \times S_k\} \rightarrow \{N_1 \times \ldots \times N_l\}$ that fits the records.
- We use a learning algorithm to approximate **f**.

Methodology:

- Split the n records into a **training set** and a **test set**.
- Build a learning algorithm with the **training set** by minimizing a **loss/objective function**
- Compute metrics on the **test set** to verify that the learning algorithm generalizes well to **unseen cases**

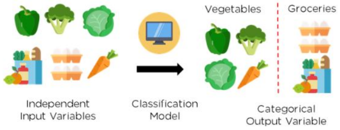
Supervised Learning

Regression:

- You wish to predict a **continuous variable** for any upcoming **input data** from the problem domain.

Classification:

- You wish to predict a **class label** for any upcoming **input data** from the problem domain.

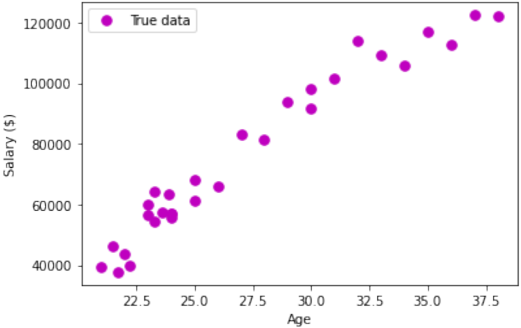


A regresssion problem

The data below is representative of american **salaries** as a function of **age**. We may observe that these features seem highly correlated.

Regression question:

How can we predict the salary of an American citizen who is 30 years old?



Regression with Linear regression

The most famous regression algorithm is **linear regression**.

Idea:

Let us consider a set of data points $X_i \in \mathbb{R}^n$ to which a set of points $Y_i \in \mathbb{R}$ is associated $\forall i \in \{1, \ldots, n_{tr}\}$. We consider that there is a **linear relation** between $\{X_i\}$ and $\{Y_i\}$.
We may hence write $Y_i \approx \sum_{j=1}^{n} \beta_j X_{ij} + \beta_0$.

Solve a linear regression problem:

We wish to find the hyper-parameters $\{\beta_0, \beta_1, \ldots, \beta_n\}$ that minimize the **sum of squared residuals** (+**potential regularization**).
We call the $SSR = \sum_i^{n_{tr}} (Y_i - \sum_{j=1}^{n} (\beta_j X_{ij}) - \beta_0)^2$ the **loss function** we wish to minimize. This leads to the normal equations (seen in chapter 5)

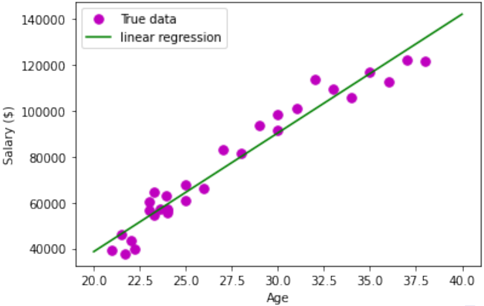
Solution of the regression problem

By minimizing the **SSR** for our salary prediction problem, we get:

$$Sal = -65000 + 5200 \times Age$$

Answer to the regression question:

We predict that a citizen of 30 years old has a salary of
$Sal = -65000 + 5200 \times 30 = 91000$ \$

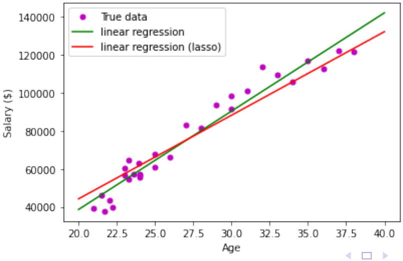


Regularizations in Linear Regression

What is regularization?
Regularization is a technique that modifies the loss used for training a learning algorithm such that the model generalizes better to unseen data.

Regularization techniques for linear regression:

- Lasso regression (loss function + $\lambda\|\beta\|_1$)
- Ridge regression (loss function + $\lambda\|\beta\|_2$)
- ElasticNet regression (loss function + $\lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2$)

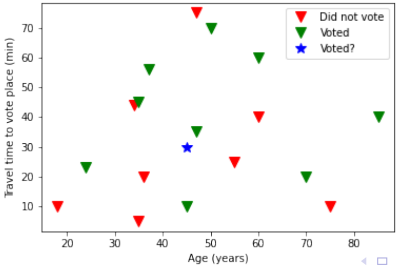


A classification problem

We have some representative data for the action of voting (voted/did not vote), which is put in correlation with two **features**: **age** and **travel time to vote place**.

Classification question:

How could we predict if a **citizen** voted or not knowing his age (44) and travel time to vote place (32 min)?



Classification with K-nearest neighbors (KNN)

Let us consider a set of **training** points $X_i \in \mathbb{R}^n$, $\forall i \in \{1, \ldots, n_{tr}\}$ such that each point is assigned one class from $\{c_1, \ldots, c_{n_{cl}}\}$.

In order to classify a point $\boldsymbol{X}$, KNN works as follows:

1. Select an integer $\boldsymbol{k}$ defining the number of neighbors to consider
2. Compute the *Euclidean distance* btw $\boldsymbol{X}$ and all training points
3. Select the $\boldsymbol{k}$ closest neighbors to $\boldsymbol{X}$
4. Among these $\boldsymbol{k}$ neighbors, count the number of points belonging to each class
5. Assign to $\boldsymbol{X}$ the most represented class among its $\boldsymbol{k}$ neighbors

Hyper-parameter selection with KNN

How to choose **k**?

- Not too small (too sensitive to noise)
- Not too big (not sensitive to class boundaries)

⇒ We need to try out different possibilities & select the **k** maximizing some metric (i.e. accuracy)

Answer to the classification question:

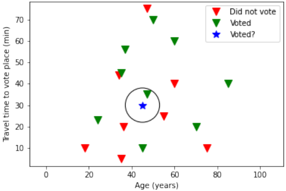


Figure: With $k = 1$, we would predict: **Voted**

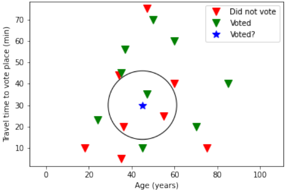


Figure: With $k = 3$, we would predict: **Did not vote**

Classification with Logistic Regression: introduction

Idea

Let's use a **linear regressor** in order to build a **classifier**!

Classification with Logistic Regression: introduction

Idea

Let's use a **linear regressor** in order to build a **classifier**!

Intuitive implementation:

Let's consider a binary classification problem with samples $(x_i, y_i) \in \mathbb{R}^p x\{0, 1\}$. We build a linear regressor and fit coefficients β according to

$$min_{\beta \in \mathbb{R}^p} \sum_{i=0}^{n} (y_i - \beta^T x_i)^2?$$

Then, we round $\beta^T x, x \in \mathbb{R}^p$ to the closest value (0 or 1) in order to build the classifier.

Classification with Logistic Regression: Introduction

Several problems arise with this idea:

- We can not use routines for statistical inference (confidence intervals, ...)
- It is hard to interpret the coefficients β
- How to handle when we have more than 2 classes?

Here is where **logistic regression** comes into play!

Classification with Logistic Regression: idea

Instead of modelling Y directly from X, we will instead model $\mathbb{P}(\boldsymbol{Y} = 1|\boldsymbol{X} = \boldsymbol{x})$ that we will note $\boldsymbol{p}(\boldsymbol{x})$.

We suppose a **logistic model**:

$$p(x) = \frac{exp(\beta^T X)}{1 + exp(\beta^T X)} = \sigma(\beta^T X)(\textbf{sigmoid function})$$

such that $p : \mathbb{R}^p \to [0, 1]$.

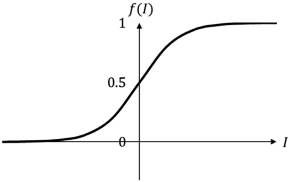


Figure: Sigmoid function (σ)

Solving the Logistic Regression problem

How to find the coefficients β?

Given n samples

$$(x_i, y_i) \in \mathbb{R}^p x\{0,1\}, i = 1, \ldots, n,$$

We might write the **likelihood** associated to our logistic model (probability function):

$$\begin{aligned} L(\beta) &= \Pi_{i:y_i=1} p(x_i) \times \Pi_{i:y_i=0}(1 - p(x_i)) \\ &= \Pi_i p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \end{aligned}$$

and the **log-likelihood** is then written:

$$log(L(\beta)) = \sum_{i=1}^{n} y_i log(p(x_i)) - (1 - y_i) log(1 - p(x_i)).$$

This is the **binary cross-entropy**!
As maximizing likelihood = maximizing log-likelihood, we just have

Representation of logistic regression

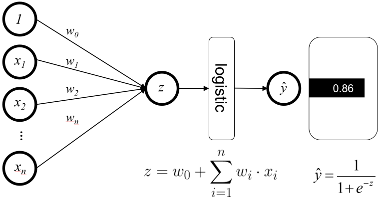


Figure: logistic regression visualized

Representation of logistic regression

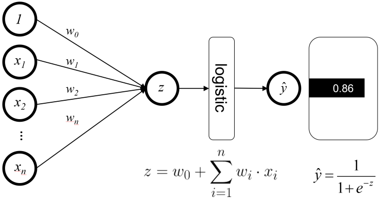


Figure: logistic regression visualized

Representation of logistic regression

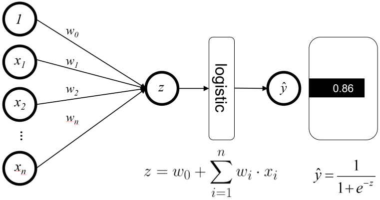


Figure: logistic regression visualized

Representation of logistic regression

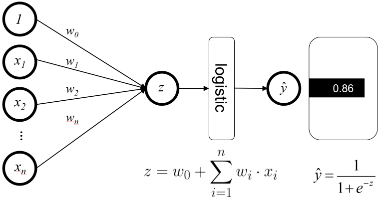


Figure: logistic regression visualized

...

Representation of logistic regression

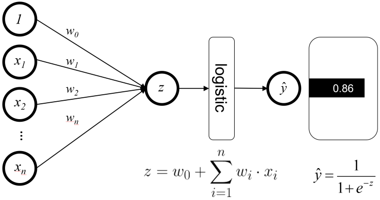
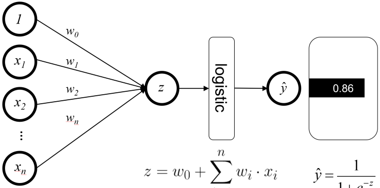


Figure: logistic regression visualized

....

Representation of logistic regression



$$z = w_0 + \sum_{i=1}^{n} w_i \cdot x_i \qquad \hat{y} = \frac{1}{1+e^{-z}}$$

Figure: logistic regression visualized

.....

Representation of logistic regression

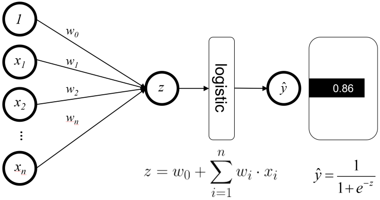


Figure: logistic regression visualized

..... WHAT???

Representation of logistic regression

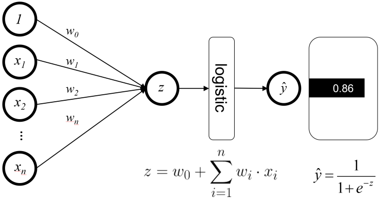


Figure: logistic regression visualized

..... WHAT??? .

Representation of logistic regression

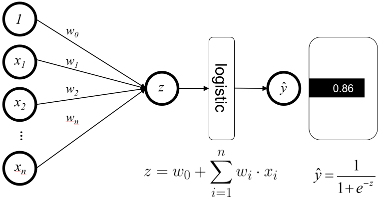


Figure: logistic regression visualized

..... WHAT??? ..

Representation of logistic regression

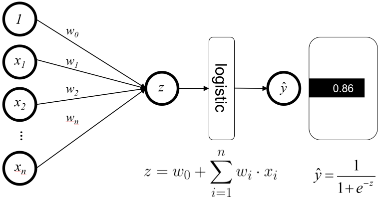


Figure: logistic regression visualized

..... WHAT??? ...

Representation of logistic regression

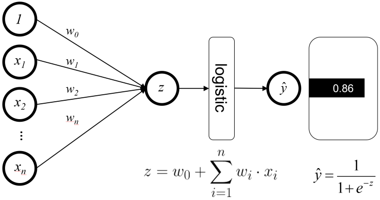


Figure: logistic regression visualized

..... WHAT??? ....

Representation of logistic regression

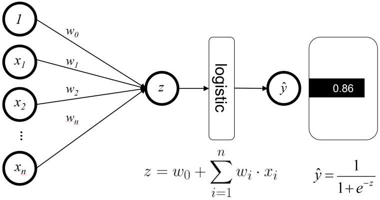


Figure: logistic regression visualized

..... WHAT??? .....

A wild neural network appears!

Introduction to deep learning

Neural networks imitate the functioning of the human brain.
They consist of:

- Neurons (represented as circles)
- Activation functions associated to neurons (e.g. sigmoid)
- An input layer/an output layer & 0+ hidden layers
- Vertices connecting neurons of different layers to which **weights** (to be learned) are associated

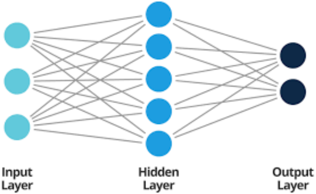


Figure: Representation of a deep neural network

Training a neural network

Neural networks might be used for a wide variety of tasks (i.e. regression/classification, . . .).
The objective is to find the **vertex weights** that will **minimize some loss function** evaluated on the **training data**.

The steps followed when training a neural network are:

1. Initialize vertex weights
2. Compute the **derivatives** of the loss associated to the different weights and update these (*backward propagation*)
3. Evaluate a loss function on the training data (*forward propagation*)
4. Repeat from step **2** until satisfied with loss or other metrics...

Neural network architectures

Theoretically, deep neural networks can approximate any function.
In practice, it is sometimes difficult to converge to an **optimum**.
This is why different NN architectures appeared: