Week 2

Reinforcement Learning (Por Refuerzo)
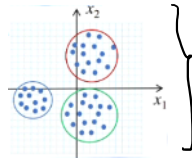↳ learning by prices/punishment
↳ It is an <u>Artificial Learning</u> type.
                        also known as
                        Machine Learning

3 types:

1. Non-supervised Learning

   ° The problems of this kind must be defined in a <u>space</u> <u>of</u> <u>States</u>

   This is a simple state space,
   a space with 2 variables (x, x_2)
        ↳ Regularly, the problem is a series
           of points in this state space.
           i.e. one point is an instance of
                   the problem.

   given a new point,
   the machine is expected to classify it
   ⌐ ° The objective of NS Learning is to group these points in the
   └ state space, forming CLUSTERS (Groups).
                  ↳ This groups come from the assumption that there
   The machine finds   Exists a <u>distance</u> that separates the instances of
   out on its own the   the problem
   ⌐ classes,
   └ ° The data is not previously classified or labeled, thus
        the machine gives the data a group by their similarities

2. Supervised Learning

   ° Once again the problem is inside a state space, and we
     got the same space.

   ° The difference with these problems is that we provide
     the input to the machine for it to learn, and this
     input is already classified inside groups
              ↳ instances that belong to a <u>known</u> class

                      =

   The machine learns   The machine thus learns what does
   this by building ←───── an entity of class c_i look like.
   a curve inside
   the state space ° We expect that, if we got an unknown point (test),
        ↓   the machine is able to classify it correctly.
      that curve
      separates
      the data   ° Thus, by the data being classified, we say it is LABELED data.

3. Reinforcement Learning:

   ° In this case, there is not a series of instances given
        to the machine,
              ↳ unlike the NS and S learning

        instead, what we want is the machine to learn
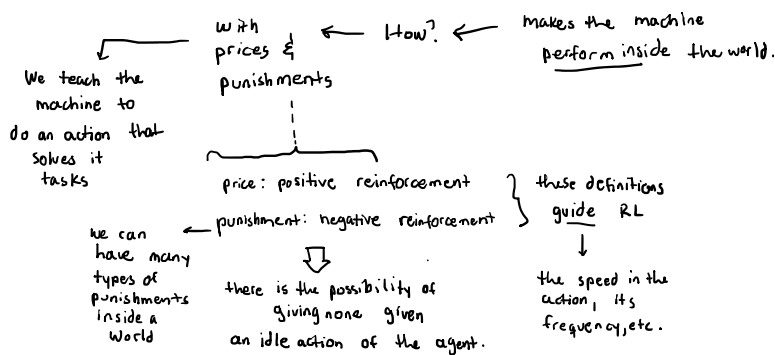   to perform a series of <u>actions</u> in the world
                      ↳ in order to solve
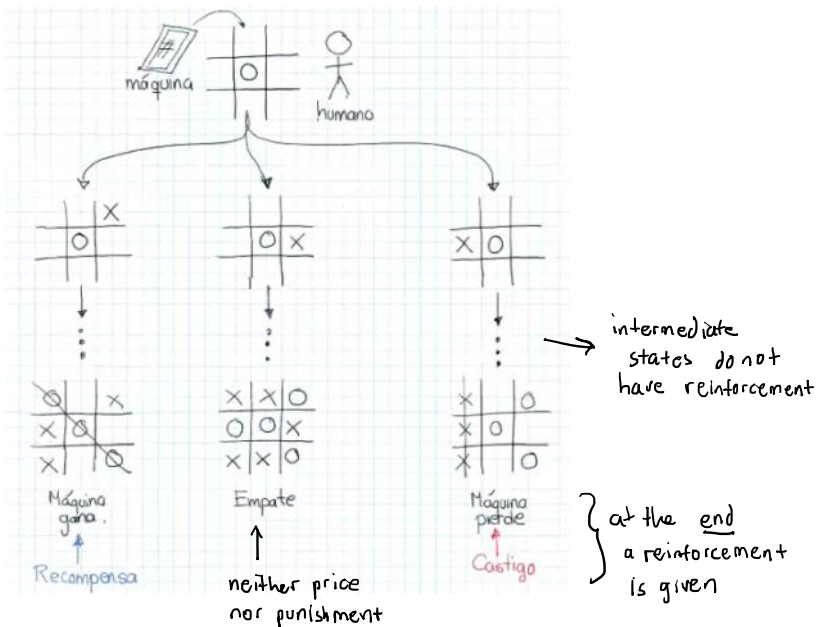   a PROBLEM.

   i.e. a sweeper robot:

      } the NS and S Learning
                            recognize things
                            in the world
                              ↓
                          Reinforcement Learning

We teach the machine to do an action that solves it tasks ← with prices & punishments ← How? ← makes the machine perform inside the world.

price: positive reinforcement
punishment: negative reinforcement } these definitions guide RL

↓

we can have many types of punishments inside a World ← there is the possibility of giving none given an idle action of the agent.

the speed in the action, its frequency, etc.

o In order to do RL, we need to define:
→ a World
→ a series of actions, available to the agent
→ the Reinforcement (price/punishment)

o Another example is: the game #

intermediate states do not have reinforcement

Máquina gana.
Recompensa

Empate
neither price nor punishment

Máquina pierde
Castigo

} at the end a reinforcement is given

NS:
not classified
a) data: NON-LABELED, also known as NON STRUCTURED data.
b) Function: Grouping data (clustering) as defined entities
   Reduce dimensionality, since high dimensionalities slow down ML algorithms
c) Objective: find a structure in the data, give it a label

no feedback at any time.

S:
a) data: LABELED, also known as STRUCTURED data
b) Function: classification or regressions
c) Objective: prediction (for new data), since the class already exists.

feedback: we tell the class of data

RL:
a) data: experiences from the world, that is, in which STATE it is and what REINFORCEMENT it gets.
   ↳ What stimulates the program?
      → agent
      1. State: what the program can perceive
      2. Reinforcement: price/ punishment, +/-.
b) Function: associate actions to states
   ↳ what can the agent do? $a_0, a_1, a_2, \ldots, a_n$
      actions that tell the machine what to do at every moment (politic)
      ↳ or state
   * State: the configuration of the world at an instant of time
c) Objective: Find the best actions: find the set of action where the

feedback:
the rein-forcement is the feedback

↓

may be only at the end, or at certain times

obtain the maximum positive reinforcement it can. ⇐ maximize the price or benefit and minimize the punishment ⇐ machine performs the best it can perform.

↳ programs must look for the most price at all times _____ so that

this is the "instinct" inside the program

Reinforcement Learning:

○ This type of learning has a structure, which is the abstract structure of the problem itself. This is the structure:
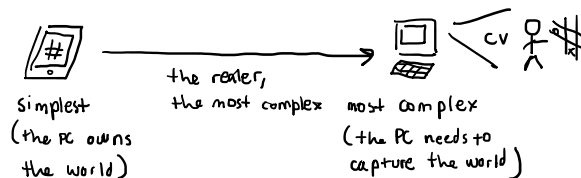
a = action

| Agent | | World |

S = state

r = price (reinforcement)
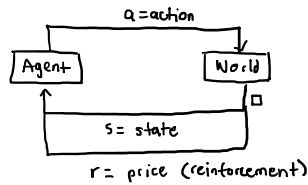
Components:

1. Agent: the program that will solve the Task.

2. World / Environment

{
the agent perceives both at all times
}

○ There is an interaction between the Agent and World:
  □ the agent can perceive the configuration of the world, the agent perceives the state of the world : i.e. any # possible board config is a world state, called S.
    ↳ the state (S) is what the agent perceives of the World.
○ The other thing the agent perceives is the reinforcement (r), which can be positive / negative. This also comes from the World.
○ Once these two are perceived, the agent performs an action (a) to the world, depending on the state it is in.

  — the agent and the World are in a cycle: the agent perceives and then acts

  maximum benefit
  ↗

  ⇨ The purpose of RL is teaching the best action to perform at any state

○ The complexity of these problems can be very different:

Simplest
(the PC owns the world)

the easier, the most complex

most complex
(the PC needs to capture the world)

→ How can we represent the Reward (reinforcement), the State and the action?
  ↳ The representation must be something that is in the machine: numbers or strings?

a) The State of the machine: it is the config of the world.
  ↳ If we use the example of # game, and we have the following state:

  PC = "O"
  Me = "x"
  Empty = " "

  the board is the World

  {
  we can represent the states with a mathematical object MATRIX
  }

  $\begin{bmatrix} 1 & \emptyset & \emptyset \\ \emptyset & 2 & 1 \\ \emptyset & \emptyset & 2 \end{bmatrix}$

  ⇩

  Every problem's states must be represented by any data structure we can think of

All representations of States will be variables
inside RAM memory (variables)
  ↳ addresses of memory

→ The variables can be:
  { Strings "abc"
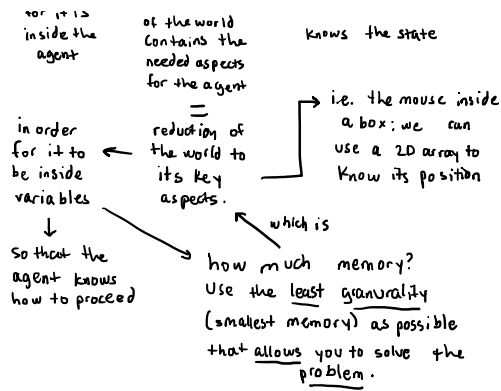   numbers —[integers / floats
    booleans V,F
    arrays: 2D, 3D, ...
  }
  use this for representations

  objects also
  ↳ the arrays are the most used for representing the World State
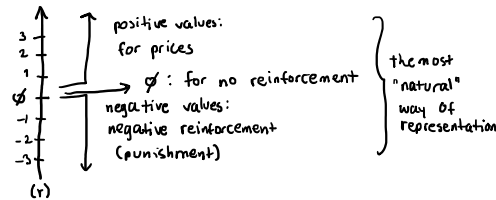   ↳ the set of variables is an internal representation of the world (State)
      agent

  "internal" ← the internal representation ← it is necessary, since that's how the agent

for it is inside the agent | of the world contains the needed aspects for the agent | knows the state

in order for it to be inside variables ← $=$ reduction of the world to its key aspects.

⌐→ i.e. the mouse inside a box: we can use a 2D array to know its position

↓
So that the agent knows how to proceed

↖ which is

how much memory? Use the least granurality (smallest memory) as possible that allows you to solve the problem.

→ How to represent the reinforcement? (reward)
↳ the easiest is to use a numeric scale:

positive values: for prices
$\emptyset$: for no reinforcement
negative values: negative reinforcement (punishment)

⎫ the most "natural" way of representation

i.e. +2 if the PC wins
−1 if no one wins
−3 if the PC loses

⎫ the agent moves so that it gets the max benefit (the max reward) ⇐ "the agent's direction"
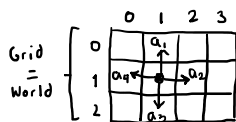↳ avoid losing states
↳ pursuing the winning states

→ How do we represent the actions?
↳ a set of actions the agent has and that it can perform over the world, changing the world.

In an example where an agent moves inside a grid:

Grid $=$ World

the • (agent) has actions a_1, a_2, a_3 and a_4 which involve a one-cell move, for example.
↳ the 'available' actions change every time the agent changes the state.