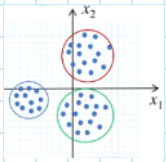
Week 2

Wednesday, February 23, 2022 4:22 PM

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

3 types:

1. Non-supervised Learning
   - The problems of this kind must be defined in a space of states



This is a simple state space, a space with 2 variables (x_1, 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 exists a distance that separates the instances of the problem
- The machine finds out on its own the 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 Known class

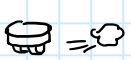
=

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

makes the machine perform inside the world. ← How? ← with prices & punishments

We teach the machine to do an action that solves it tasks

price: positive reinforcement

punishment: negative reinforcement

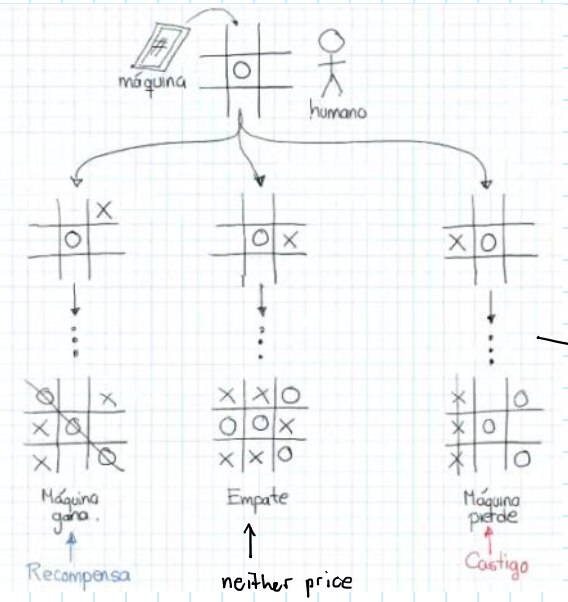
these definitions guide RL → the speed in the action, its frequency, etc.

we can have many types of punishments inside a World

there is the possibility of giving none given an idle action of the agent.

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

- Another example is: the game



→ intermediate states do not have reinforcement

at the end a reinforcement is given

neither price nor punishment

NS: (not classified)

no feedback at any time.

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

S:

feedback: we tell the class of data

a) data: LABELED, also Known as STRUCTURED data

b) Function: classification or regressions

c) Objective: prediction (for new data), since the class already exists.

RL:

feedback: the reinforcement is the feedback ↓ may be only at the end, or at certain times

a) data: experiences from the world, that is, in which STATE it is and what REINFORCEMENT it gets.
↳ What stimulates the program?
1. State: what the program (agent) can perceive
2. Reinforcement: price/punishment, +/-.

b) Function: associate actions to states
↳ what can the agent do? $a_0, a_1, a_2, \cdots, a_n$
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

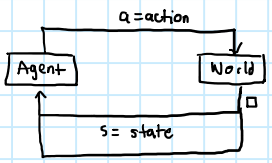
machine performs the best it can perform.
↳ programs must look for the most price at all times — so that
this is the "instinct" inside the program

maximize the price or benefit and minimize the punishment

obtain the maximum positive reinforcement it can.

Reinforcement Learning:

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



Components:
1. Agent: the program that will solve the Task.
2. World/Environment

- 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.

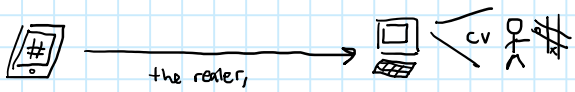
(the agent perceives both at all times)

- 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

↳ The purpose of RL is teaching the best (maximum benefit) action to perform at any state

- The complexity of these problems can be very different:

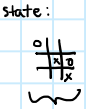


Simplest (the PC owns the world) — the router, the most complex — most complex (the PC needs to capture the world)

→ 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
- booleanos V,F
- arrays: 2D, 3D, ...

(use this for representations)

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

"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 = reduction of the world to its key aspects.

knows the state

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

in order for it to be inside ← variables

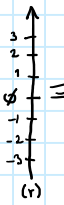
↓ so that the agent knows how to proceed

which is

how much memory? Use the least granularity (=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

Ø: 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"

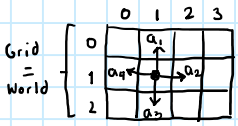
↳ avoid losing states

↳ 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:



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.