2. Players and adversaries

II. Correlating Skills with Strategies

Giacomo Bergami

September 12, 2021

Newcaslte University

Objectives

- Modelling the behaviour of one single agent for assessing the meaningfulness of its decision.
- Characterizing agent learning a plan for a specific problem through positive/negative feedback.
- Characterizing expertise via farsightedness.
- Identifying the limits of a pure application of the RL model.

Player versus Environment (PvE)

- They are also referred as *One-person games*, or *Games against nature*.
- Before tackling the problem of an interaction with another agent, we should first ask ourselves whether we are able to model agent that is capable of taking meaningful decisions:
  - No decision is with no effect,
  - but, one among them, will lead to the optimal solution.
- This person may take the wrong decision, but there does not exist a conscious opponent.
- One-person games are greatly investigated in AI (robotics) and in Planning Theory.

Modelling Skills through Reinforcement Learning

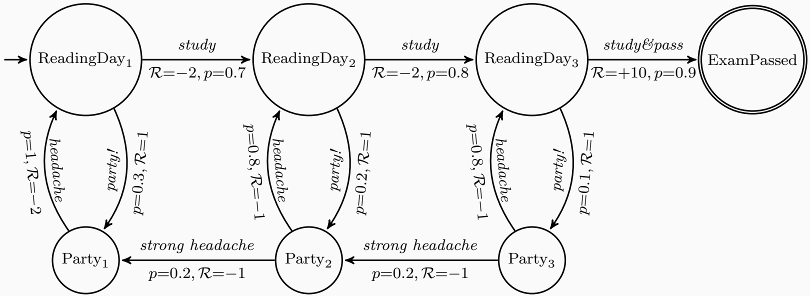
We build skills in time by seeing a lot of circumstances happening. I would be then able to extract a mental plan *only after* summarising all the possible situations that I encountered.

Reinforcement Learning characterizes skills as follows:

1. *discount rate*, $0 \leq \gamma \leq 1$: how much can I forecast in the future, so to maximise the total reward and ignore local "negative rewards"?
2. *plan*, $q_\pi(a|s) \in \mathbb{R}$: which is the "likelihood" of performing a given action a when the player with a specific discount rate γ is in one specific state s?

As an intermediate learning outcome, the agent learns a numerical *value* $\mathcal{V}(s)$ associated to each state s, describing the agent's perceived desirability of being in a current state in the long run. In the GOAP jargon, this value is called insistence value.

Passing an Exam as a Markov Decision Process



Markov Decision Processes (MDP) extend DTMC with the following:

- Each transition is labelled with an action name (e.g., *study*)
- and is associated with a reward value $\mathcal{R}$

In this simple scenario, the reward is just the one percieved by the student while studying for passing the exam (no supervision).

A Well-Balanced Game

This simple "game" models a well-balanced game:

- The player doesn't get an irretrievable advantage until they almost won ($\mathit{ReadingDay}_3$).
- Early mistakes should not make a game unwinnable (we can always go back and study!).
- Uncertainty over the reward makes the game more desirable for many players.

In addition to that, a well-balanced game should increase the rewards towards the end, so to give the player the feeling they are actually progressing in the game.

Learning How to Pass an Exam (1/2)

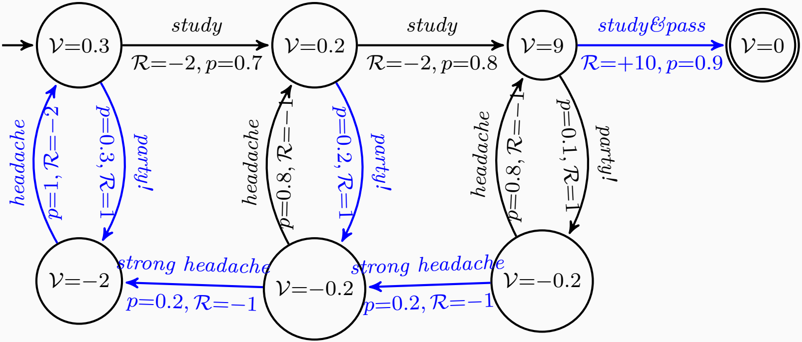
The following learning algorithm takes as an input only γ and the MDP, and returns the numerical value $\mathcal{V}$ and a reward-weighted probabilistic plan $q_\pi(a|s)$.

- given the probabilistic plan, we might get the desired deterministic version as $\pi(s) = \max \arg_a q_\pi(a|s)$
- an user $\gamma = 0$ will be a beginner exploiting a greedy reward approach on the local actions (short-sighted), while $\gamma = 1$ will have perfect foresight in maximizing the total reward.

Learning How to Pass an Exam (2/2)

function $\textsc{ValueIteration}(\gamma, MDP = (V, E))$
    for all $s \in V$ **do**
        if *accepting*(s) **then** $\mathcal{V}(s) := 0$ **else** $\mathcal{V}(s) :=$ `rand()`
    repeat
        $\Delta := 0$
        for all $s \in V$ **do**
            $v := \mathcal{V}(s)$
            $\mathcal{V}(s) := \max_a \sum \left\{ p(\mathcal{R} + \gamma \mathcal{V}(s')) \;\middle|\; s \xrightarrow[p,\mathcal{R}]{a} s' \in E \right\}$ $\Delta := \max\{\Delta, |v - \mathcal{V}(v)|\}$
    until $\Delta < \epsilon$
    $\mathbb{P}(a|s) = \sum \left\{ p(\mathcal{R} + \gamma \mathcal{V}(s')) \;\middle|\; s \xrightarrow[p,\mathcal{R}]{a} s' \in E \right\}; \; \pi(s) = \max \arg_a \mathbb{P}(a|s)$
    return $\mathcal{V}, \mathbb{P}, \pi$

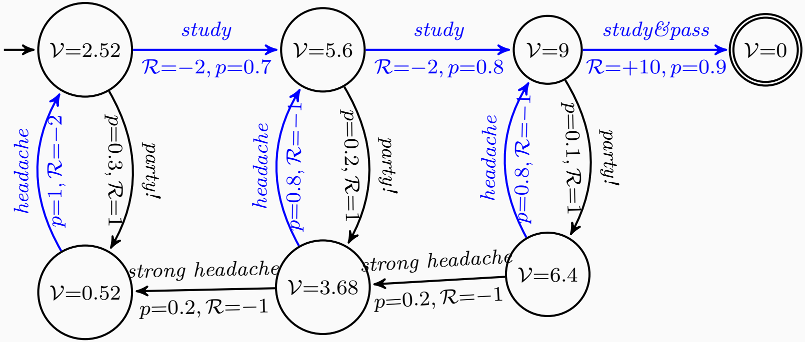
Novice Strategy



The above diagram describes the learning outcome for $\gamma = 0$:

- The weight associated to each state reflects the maximization of the combined provision of probability and reward of each single edge.
- If interpreted deterministically (π in blue), the agent will never reach the accepting state, as it will only *seize the day*.

Expert Strategy



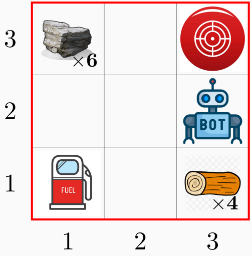
The above diagram describes the learning outcome for $\gamma = 1$:

- The choice of *studying* implies that the agent is able to completely ignore the local maximum.
- If it might happen to party, then favour the choice allowing the better "comeback".
- If interpreted deterministically (π in blue), the agent will always reach the accepting state: dominant strategy.

Relational Learning and Difficulty in PvE games

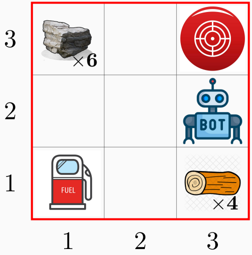
- In our simple game, any winning strategy (*plan*) will contain *study* actions.
- In more realistic games, there could be multiple possible actions for winning the game (more possible outgoing edges).
- Nevertheless, real world users' goal is to win the game, and not necessarily to collect the best score: so, different possible ways to reach the final solution might be considered:
  - $\Rightarrow$ we can extract different possible plan by varying γ and $\mathcal{R}$ (perceived reward).
  - We can also extract multiple possible plans, and provide suggestions/alternatives to the novice player.

State Explosion: Robot Campfire (1/3)



- A fuel-powered robot must build a campfire at $(3, 3)$ by taking at least 3 block of stones and 2 logs. To ignite it, he uses some of his fuel.
- The robot can transport only 1 item (log or stone) at a time.
- The robot has a tank containing 11 units of fuel. While moving towards one cell, the robot consumes 1 unit of fuel at a normal speed, and 2 units of fuel if moving at twice the speed. If it is transporting an object, it consumes 50% more.
- When refueling, it can only get full (max. 11 units).
- The robot can use some of his fuel over the campfire only if there is already some log (*simplification*).
- The robot has 50 time units to do finish.

State Explosion: Robot Campfire (2/3)



The robot is a simple mind:

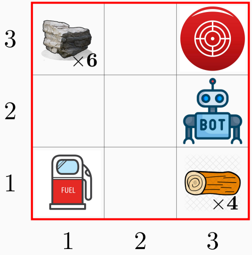
1. "*If I'm running low on fuel, I should prefer to move to the fuel station*".
2. "*If I'm bringing either a log or a stone, I should prefer to move towards the base*".
3. "*If I move, I not go back to the same cell where I was in the previous instant*".

It is unaware of the rules leading to the goal:

- it tries to perform actions out of the blue.
- The robot needs to try some actions, so to collect the reward associated to it. The final goal is still to maximise the total reward.

⇒ all the possible actions are on the table!

State Explosion: Robot Campfire (3/3)



For learning the strategy, the robot receives positive/negative feedback for each one of his actions:

1. The greater is the energy consumed, the lesser is the received feedback.
2. The greater is the amount of time required to achieve the goal, the lesser is the received feedback.
3. The bot receives a positive feedback when refueling.
4. The bot receives a positive feedback when setting up the campfire, but a negative one when it wastes resources (i.e., takes more stuff than required).

State explosion

With the previous game configuration:

- We generate ≈ 11M possible states,
- of which $\approx 42\%$ are final states, but
- less than 1% of those are accepting states

By just removing the fast-paced action, we considerably decrease the number of the final states:

- We generate ≈ 1.67M possible states,
- of which $\approx 40\%$ are final states, but
- even fewer of those are accepting states

Some strategies to reduce the state explosion:

- Reduce the size of the resulting graph by minimizing the automaton (DFA minimization).
- By doing so, use numerical variables (*fluents*) to store agent (e.g., fuel level) and world (e.g., number of resources) numerical information.
- Use *Non-Markovian* formulations of the problem.