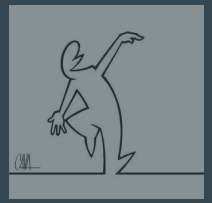
Reinforcement learning

why and how?

...



Nima H. Siboni
https://github.com/nima-siboni

Course outline

- Introduction

  What sort of problems you can solve with it? How is it new to you?

- RL problem formulation

  Modeling: Lots of new terms to be defined and connected to each other

- Solution of a RL problem

  Sim. Env. and a zoo of methods

Introduction

How do we make good decisions?!

What is the situation?!

State

What actions do I have?

Action space

What are the consequences of each action?

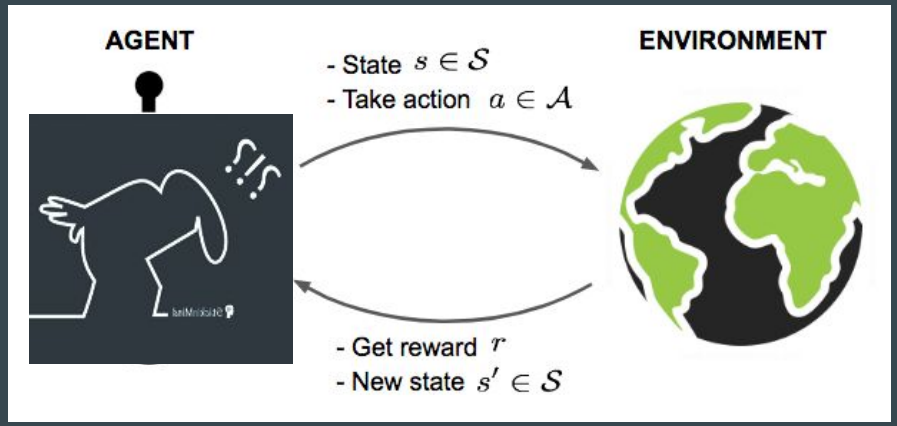
- How rewarding/costly is each action?
- Where do I end up after this action?

Environment

Policy

The mapping between the state and the actions

State → Policy → Action



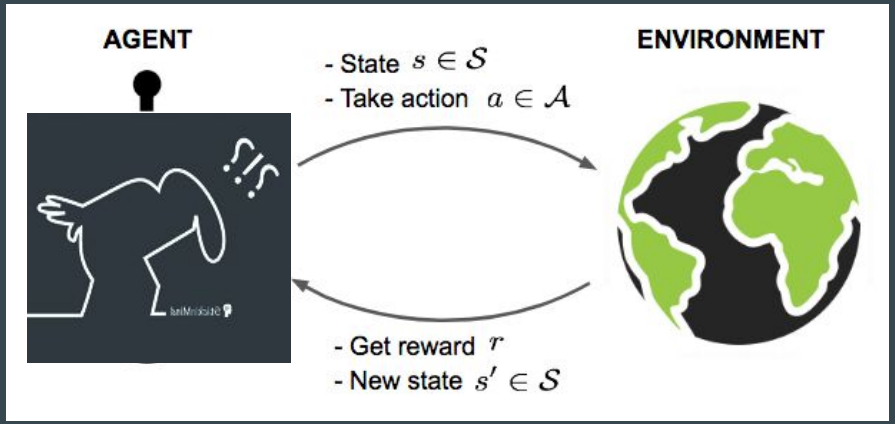
Introduction

Optimization formulation

Given the state,

what is the action that maximizes

my expected reward



How is it a machine learning problem?

Reinforcement Learning

Learning the ***OPTIMAL*** policy from ***EXPERIENCE***

Anatomy of a RL solution:

- Start with a policy
- Make it better (!!)

  until you reach the optimal policy

Examples

Give me examples!

- What are the states?
- What are the actions?
- How is the dynamics?
- What are the rewards at each step?
- What is your policy?

FfT: Can you always cast your problem into a RL problem?

- Reward hypothesis

  That all of what **we** **mean by goals and purposes** can be well thought of as **maximization** of the expected value of the **cumulative sum** of a received **scalar signal** (**reward**).

FfT: Can you think of a badly designed reward?

How to make good decisions?

- a table,
- a function, or
- a deep neural network (*Deep RL*)

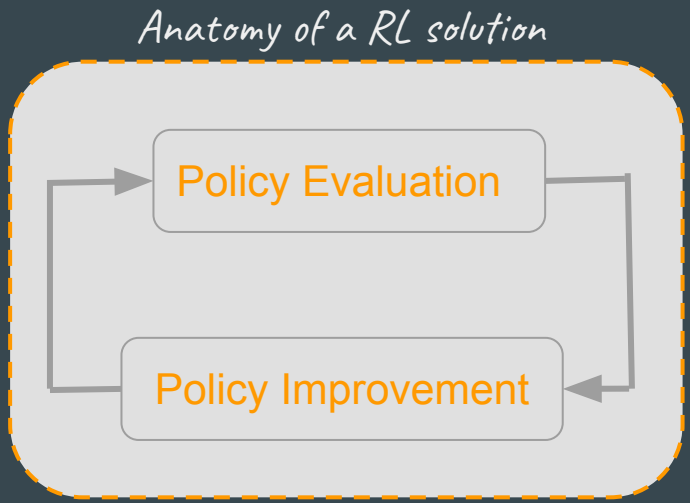
The inner mechanism of finding the optimal solution

- Change the policy a bit and evaluate the consequences!
- Find the *value* of each (state, action) pair from (your experience or other's) and take the best action!

DQN,
DDQN,
AC,
A3C,
SAC,
REINFORCE,
...

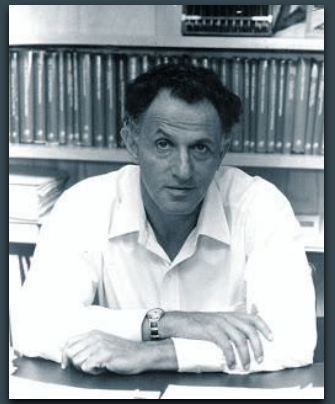
The core questions

- How good is a policy?
- How can you make a policy better?



The optimization problem is solved you can find the V(s)
for a given policy

Policy Evaluation



Q: A measure of how good a policy at a given state is?

- Immediate reward,
- Expected value of the state,
- Discount

Q: How to find out V(s) for all the states? Bellman Equation

- Directly solving the Bellman Equation
- Guessing based on the experimenting [Monte Carlo method, Q-Learning, DQN]

(Finally) Bellman Equation

- What happens until eternity is what happens now plus what happens after that

Let's derive the Bellman equation together!

- Assumptions:
  - A Markovian process (is it an important assumption?)
  - The dynamics is known!
  - States and actions are discrete!

- Bellman equation has a good mathematical property! A lovely one!

How to solve the Bellman equation?

How to solve an equation?

- Direct analytical methods
- Numerical (iterative/approximate) methods: Not *"the"* solution but a *good enough* solution

When to use which?

- exists an analytical approach and if it is computationally feasible → analytical
- otherwise → numerical methods

How to solve the Bellman equation? (Cntd.)

$$x = f(x)$$

- Drawing (yes, why not?!)
- Midpoint method
- Iterative fixed-point methods
- FfT: Perturbative methods (start from where you know the best!)

How to solve the Bellman equation? (Cntd.)

Exercise

- Choose a policy
- Evaluate the policy
- (Use the evaluation to) Improve the policy
  - Unless you are happy(!) go back to Evaluation Step

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Summary so far

- Basics of an RL problem
- The cornerstone of RL → Bellmann Eq.
- An iterative evaluation of a policy & Improving the policy to the optimal one

What if?

- What if the environment is not fully observable?
- What if the optimal policy is stochastic?
- What if the environment is stochastic?
- What if the process is not Markovian?
- What if the states/actions do not fit into a table?
- What if we do not know the dynamics of the environment?

Learning from experience

Goal: finding the optimal behavior, without prior knowledge of the environment.

There are methods which require only experience!

Experience: sample sequences of states, actions, and rewards (from interaction with the environment).

Monte Carlo

Algorithms relying on repeated random sampling!

Applications: any problem having a probabilistic *interpretation*.

Give me examples!

Examples:

- Finding a probability distribution of dice
- Finding the area of lake
- Finding the value function!

Monte Carlo Control

Exercise

Same structure as the previous exercise

- *Choose a policy*
- *Evaluate the policy using MC*
- *(Use the evaluation to) Improve the policy*
  - *Unless you are happy(!) go back to Evaluation Step*

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

?

Go for Q(s, a) directly

What if policy doesn't let you see all states?

Q-learning

How could we make the previous algorithm better?

- What was in-efficient?
- How is it different from the way we learn?
- When did it took so long?

Let’s Bootstrap!

Let’s make the best out of what we have learned!

SARSA method

Q-learning (Cntd.)

"SARSA" method for policy evaluation

lets derive SARSA together!

We can use replace the MC policy evaluation with "SARSA" in the general scheme.

But we can also do better: Q-learning!

lets derive Q-learning together!

Q-learning (Cntd.)

Exercise