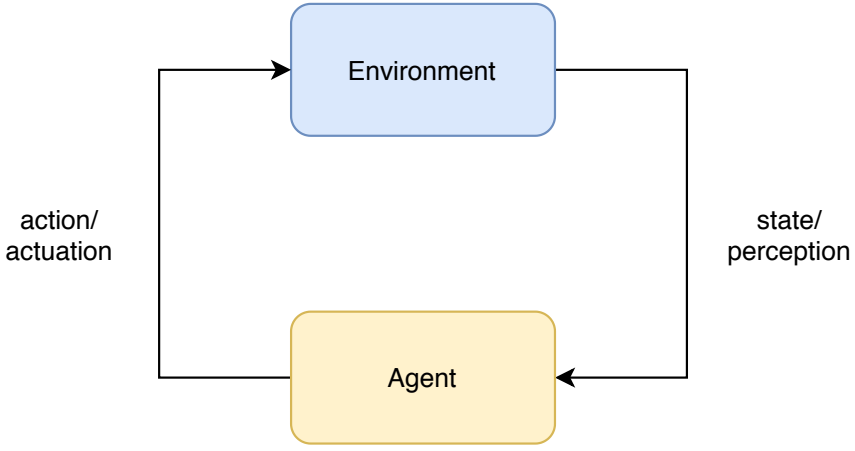
facebook
Artificial Intelligence Research

How to model an RL problem: Markov Decision Processes

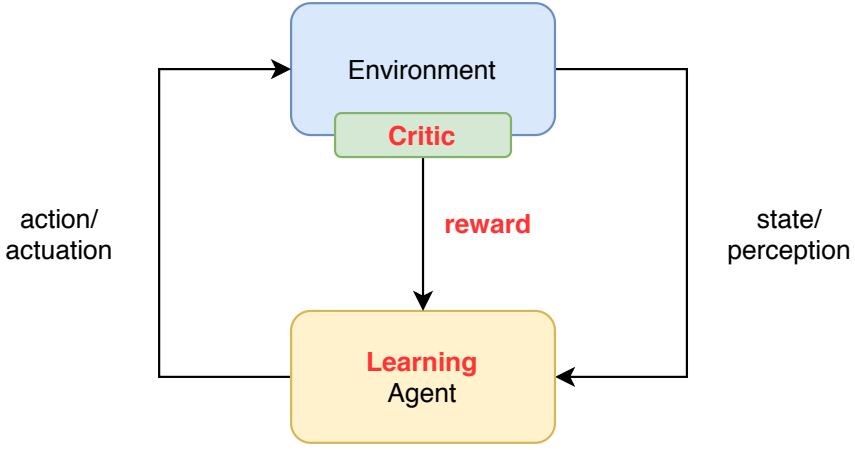
Alessandro Lazaric
Facebook AI Research

Outline

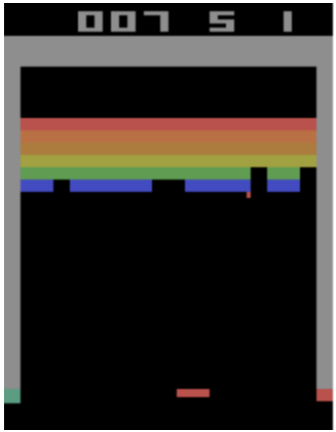
The Reinforcement Learning Model



The Reinforcement Learning Model



ATARI Breakout



- Agent: *paddle*
- Environment: *ball and bricks*
- Reward: *points*

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** is defined as a tuple $M = (S, A, p, r)$ where

- S is the *state* space,
- A is the *action* space,
- $p(s'|s, a)$ is the *transition probability* with

$$p(s'|s, a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a),$$

- $r(s, a, s')$ is the *reward* of transition (s, a, s').

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** is defined as a tuple $M = (S, A, p, r)$ where

- S is the *state* space,
- A is the *action* space,
- $p(s'|s, a)$ is the *transition probability* with

$$p(s'|s, a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a),$$

- $r(s, a, s')$ is the *reward* of transition (s, a, s'). } simplified to $r(s, a)$ or $r(s)$

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** is defined as a tuple $M = (S, A, p, r)$ where

- S is the *state* space,
- A is the *action* space,
- $p(s'|s, a)$ is the *transition probability* with

$$p(s'|s, a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a),$$

- $r(s, a, s')$ is the *reward* of transition (s, a, s'). } simplified to $r(s, a)$ or $r(s)$

☞ The process generates **trajectories** $h_t = (s_1, a_1, \ldots, s_{t-1}, a_{t-1}, s_t)$, with $s_{t+1} \sim p(\cdot|s_t, a_t)$

Markov Decision Process: the Assumptions

Markov assumption: the current state s and action a are a sufficient statistics for the next state s'

$$p(s'|s,a) = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$$

Possible relaxations

- Possible to extend to continuous state-action space
- Define a new state $x_t = (s_t, s_{t-1}, s_{t-2}, \ldots)$ (i.e., k-order MDP)
- Move to partially observable MDP (PO-MDP)
- Move to predictive state representation (PSR) model

ATARI Breakout

$$\mathbb{P}\left[s_{t+1} = \text{  } \,\middle|\, s_t = \text{  }, \text{no-move} \right]$$

Non-Markov dynamics

ATARI Breakout

$$s_t = \left\{ \right.$$  $$\left. \right\}$$

4 consecutive frames = single observation

Markov Decision Process: the Assumptions

Time assumption: time is discrete

$$t \to t + 1$$

Possible relaxations

- Identify the proper time granularity
- Most of MDP literature extends to continuous time

ATARI Breakout

$a_t = \text{left}$

t

ATARI Breakout

$a_t = \text{left}$

$t + 1$

Too fine-grained resolution

ATARI Breakout

$a_t = \text{left}$

t

ATARI Breakout

$a_t = \text{left}$

$t + 1$

Too coarse-grained resolution

Markov Decision Process: the Assumptions

Reward assumption: the reward is uniquely defined by a transition (or part of it)

$$r(x, a, y)$$

Possible relaxations

- Distinguish between global goal and reward function
- Move to inverse reinforcement learning (IRL) to induce the reward function from desired behaviors

Markov Decision Process: the Assumptions

Stationarity assumption: the dynamics and reward do not change over time

$$p(y|x,a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a) \qquad r(x,a,y)$$

Possible relaxations

- Identify and remove the non-stationary components (e.g., cyclo-stationary dynamics)
- Identify the time-scale of the changes
- Work on finite horizon problems

Hands-on: the Retail Store Management Problem

Question: How do you formalize this problem as an MDP? *[5min]*

At each month t, a store contains s_t *items* of a specific goods and the demand for that goods is D_t. At the end of each month the manager of the store can *order* a_t more items from the supplier. Furthermore we know that

- The *cost* of maintaining an inventory of s is $h(s)$.
- The *cost* to order a items is $C(a)$.
- The *income* for selling q items is $f(q)$.
- If the demand D is bigger than the available inventory s, customers that cannot be served leave.
- The *value of the remaining inventory* at the end of the year is $g(s)$.
- *Constraint*: the store has a maximum capacity M.
- *Goal:* maximize some measure of profit

Hands-on: the Retail Store Management Problem

Solution

- *State space*: $s \in S = \{0, 1, \ldots, M\}$.

Hands-on: the Retail Store Management Problem

Solution

- *State space*: $s \in S = \{0, 1, \ldots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \ldots, M - s\}$.

Hands-on: the Retail Store Management Problem

Solution

- *State space*: $s \in S = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \dots, M - s\}$.
- *Dynamics*: $s_{t+1} = [s_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!

Hands-on: the Retail Store Management Problem

Solution

- *State space*: $s \in S = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \dots, M - s\}$.
- *Dynamics*: $s_{t+1} = [s_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!
- The demand D_t is *stochastic and time-independent*. Formally, $D_t \overset{i.i.d.}{\sim} \mathcal{D}$.

Hands-on: the Retail Store Management Problem

Solution

- *State space*: $s \in S = \{0, 1, \dots, M\}$.
- *Action space*: it is not possible to order more items that the capacity of the store, then the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \dots, M - s\}$.
- *Dynamics*: $s_{t+1} = [s_t + a_t - D_t]^+$.
  Problem: the dynamics should be Markov and stationary!
- The demand D_t is *stochastic and time-independent*. Formally, $D_t \overset{i.i.d.}{\sim} \mathcal{D}$.
- *Reward*: $r_t = -C(a_t) - h(s_t + a_t) + f([s_t + a_t - s_{t+1}]^+)$.

Outline

Policy

Definition (Policy)

A *deterministic* stationary policy is

$$\pi : S \to A$$

Other options

- Stochastic
- History-dependent

Example: the Retail Store Management Problem

- Stationary policy composed of deterministic Markov decision rules

$$\pi(s) = \begin{cases} M - s & \text{if } s < M/4 \\ 0 & \text{otherwise} \end{cases}$$

- Stationary policy composed of stochastic history-dependent decision rules

$$\pi(s_t) = \begin{cases} \mathcal{U}(M - s, M - s + 10) & \text{if } s_t < s_{t-1}/2 \\ 0 & \text{otherwise} \end{cases}$$

State Value Function

Given a policy π

- *Infinite time horizon with discount*: the problem never terminates but rewards which are *closer* in time receive a *higher* importance.

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \,|\, s_0 = s; \pi\right],$$

  with discount factor $0 \leq \gamma < 1$:
  - *small* = short-term rewards, *big* = long-term rewards
  - for any $\gamma \in [0, 1)$ the series always converge (for bounded rewards)

State Value Function

Technical note: the expectations refer to all possible stochastic trajectories.
A (possibly non-stationary stochastic) policy π applied from state s_0 returns

$$(s_0, r_0, s_1, r_1, s_2, r_2, \dots)$$

where $r_t = r(s_t, \pi(s_t))$ and $s_t \sim p(\cdot|s_{t-1}, a_{t-1} = \pi(s_{t-1}))$ are *random* realizations.

The value function (discounted infinite horizon) is

$$V^\pi(s) = \mathbb{E}_{(s_1, s_2, \dots)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \,|\, s_0 = s; \pi\right]$$

Optimization Problem

Definition (Optimal policy and optimal value function)

The solution to an MDP is an *optimal policy* $\pi^\star$ satisfying

$$\pi^\star \in \arg\max_{\pi \in \Pi} V^\pi$$

in all the states $s \in S$, where Π is some policy set of interest.

Optimization Problem

Definition (Optimal policy and optimal value function)

The solution to an MDP is an *optimal policy* $\pi^\star$ satisfying

$$\pi^\star \in \arg\max_{\pi \in \Pi} V^\pi$$

in all the states $s \in S$, where Π is some policy set of interest.

The corresponding value function is the *optimal value function*

$$V^\star = V^{\pi^\star}$$

Hands-on: the Retail Store Management Problem

Material [link] – [10min]

Questions

- Which policy performs best? Why?
- What is the influence of the cost and reward functions in the performance of the two policies provided as example?
- Define an alternative policy and test its performance

Hands-on: the Retail Store Management Problem

Solution

Questions

- Which policy performs best? Why?

  `refill_under_threshold` works best because it avoids incurring large storage and order costs and it does not miss demand.

- What is the influence of the cost and reward functions in the performance of the two policies provided as example?

  `refill_under_threshold` works best because it avoids incurring large storage and order costs and it does not miss demand.

- Define an alternative policy and test its performance
  Who obtained the best score?

Example: ATARI Breakout

video

Thank you!

facebook
Artificial Intelligence Research