Introduction to Reinforcement Learning

Na Li

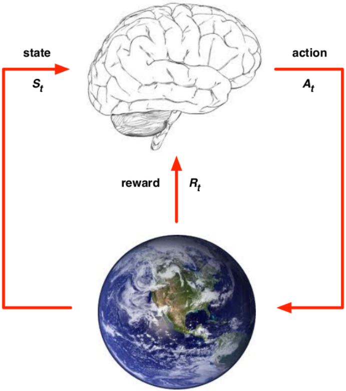
February 28, 2020

Outline

- About Reinforcement Learning
- An Example
- Limitations
- Categories
- Discrimination: Unsupervised, Supervised vs. RL
- Applications

1. About Reinforcement Learning

Introduction



- Agent:
  - Executes action A_t.
  - Receives observation S_t.
  - Receives reward R_t.
- Environment:
  - Receives action A_t.
  - Emits observation S_{t+1}.
  - Emits reward R_{t+1}.

Scenario of Reinforcement Learning

Scenario of Reinforcement Learning

Core idea

Core idea of RL:

- Interacts with the environment.
- Learns from experience.
- The target is to get the maximum expected cumulative rewards.

Reward hypothesis

Expected cumulative rewards in state S in time t:

$$V(S) = E_{\pi}[G_t|S]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots (\gamma \in [0, 1))$$

- the greater the γ, the long-term rewards are more concerned.
- MC method: using random samples to estimate expectations. –> TD: for efficiency.

Elements

policy $\pi_\theta(A|S)$

- A policy is the agent's behavior.
- It is a map from state to action.
- E.g. a function, lookup table.

Elements

reward signal $R(S, A)$

- Short-term.
- The primary basis for altering the policy.

Elements

value function $V(S)$

- A prediction of future reward, which is long-term rewards.
- Used to evaluate the goodness/badness of states.
- And therefore to select between actions.

Elements

model(optional)

- A model predicts what the environment will do next.
- $\mathcal{P}$ predicts the next state, i.e. $\mathcal{P}^{a}_{ss'} = P[S'|S,A]$
- $\mathcal{R}$ predicts the next reward, i.e. $\mathcal{R}^{a}_{s} = P[R'|S,A]$

2. An Example

An Example

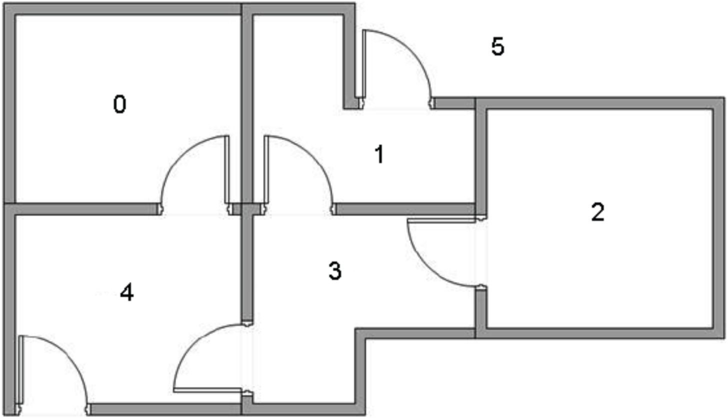
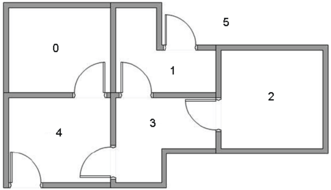
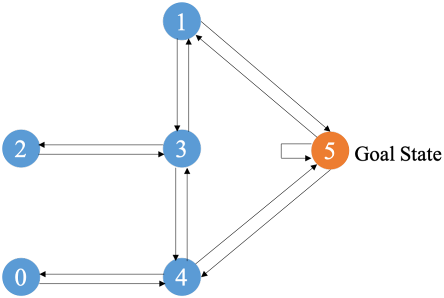


Figure: House Structure

An Example

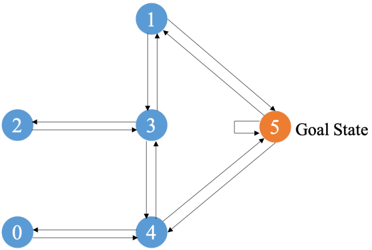




Algorithm

1. Set the λ and reward matrix R.
2. Initialize Q as zero.
3. For each episode:
   (a) select a random initial state S.
   (b) Do While the goal state hasn't been reached.
      (i) Select one possible action A from current state.
      (ii) Using A, considering going to the next state S'
      (iii) Considering the next action A'.
      (iv) Compute: $Q(S, A) = R(S, A) + \lambda max_{A'} Q(S', A')$
      (v) Set the next state as the current state.

Initialization



$$R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}, Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Episode 1:

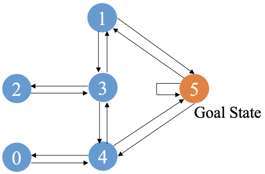
(a) Select a random initial state $S = 1$.

(b) Select one possible action $A = 5$.

(c) Next state $S' = 5$; Next action $A' = 1, 4, 5$.

(d) Compute:
$Q(1,5) = R(1,5) + 0.8 * max\{Q(5,1), Q(5,4), Q(5,5)\} = 100 + 0.8 * max(0,0,0) = 100.$

(e) $Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$



(f) $S = 5$, state '5' is the goal state and end.

Episode 2:

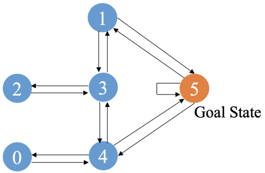
(a) Select a random initial state $S = 3$.

(b) Select one possible action $A = 1$.

(c) Next state $S' = 1$; Next action $A' = 3, 5$.

(d) Compute:
$Q(3,1) = R(3,1) + 0.8 * max\{Q(1,3), Q(1,5)\} =$
$0 + 0.8 * max(0, 100) = 80.$

(e) $Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$



(f) $S = 1$, state '1' is not the goal state and continue.

Episode 2:

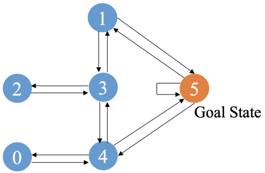
(a) Current state $S = 1$.

(b) Select one possible action $A = 5$.

(c) Next state $S' = 5$; Next action $A' = 1, 4, 5$.

(d) Compute:
$Q(1,5) = R(1,5) + 0.8 * max\{Q(5,1), Q(5,4), Q(5,5)\} = 100 + 0.8 * max(0,0,0) = 100$.

(e) $Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$



(f) state '5' is the goal state and end.

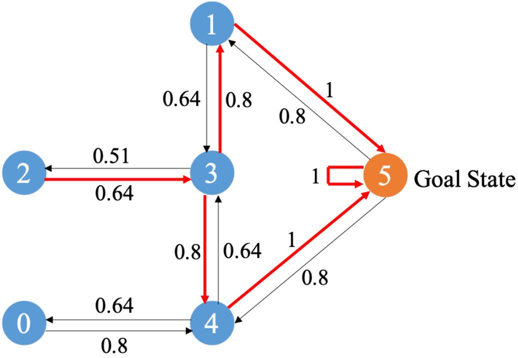
Episode N:

$$\text{Convergence result: } Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 100 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix}$$

$$\text{Normalization: } Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0.64 & 0 & 1 \\ 0 & 0 & 0 & 0.64 & 0 & 0 \\ 0 & 0.8 & 0.51 & 0 & 0.8 & 0 \\ 0.64 & 0 & 0 & 0.64 & 0 & 1 \\ 0 & 0.8 & 0 & 0 & 0.8 & 1 \end{bmatrix}$$

Result:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0.64 & 0 & 1 \\ 0 & 0 & 0 & 0.64 & 0 & 0 \\ 0 & 0.8 & 0.51 & 0 & 0.8 & 0 \\ 0.64 & 0 & 0 & 0.64 & 0 & 1 \\ 0 & 0.8 & 0 & 0 & 0.8 & 1 \end{bmatrix}$$



3. Limitations

Limitations

Reward delay.

- Playing Games.
  - Only “fire” obtains reward.
  - Although the moving before “fire” is important.
- Playing Go.
  - Gaining reward only after winning.
  - Every step is very important.

Limitations

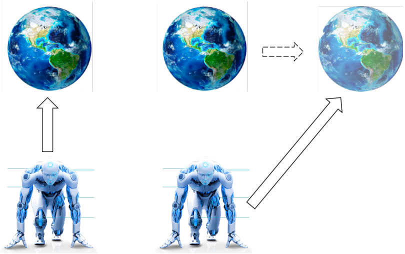
Exploration & Exploitation.

- Exploration finds more information about the environment.
- Exploitation exploits known information to maximise reward.
- It is usually important to explore as well as exploit.
- Restaurant Selection:
  Exploitation: Go to your favourite restaurant.
  Exploration: Try a new restaurant.
- Online Advertisements:
  Exploitation: Show the most successful advert.
  Exploration: Show a different advert.

4. Categories

Model-free vs. Model-based

- Model-free: learns the optimal policy. e.g. Q-learning, Sarsa, Policy-gradient, DQN, A3C, ...
- Model-based: models the environment, then based on that model, chooses the most appropriate policy.



Model-free vs. Model-based

- Model-free:
  - Gets the next state by interacting with the environment.
  - Learns by trial and error.
- Model-based:
  - Gets transition probability and return function directly.
  - Given current state S and action A, we can know the next state S’ and the value V directly.

Value-based vs. Policy-based

- Value-based:
  - Learns the values of actions (strategy evaluation).
  - Then selects actions based on their estimated action values (strategy optimization).
  - E.g., Q-learning, Sarsa, DQN.
- Policy-based:
  - Learns a parameterized policy that can select actions without consulting a value function.
  - E.g., Policy-gradient, A3C, DDPG.
  - *A value function may still be used to learn the policy parameter, but is not required for action selection.*

Value-based vs. Policy-based

Value-based: obtains $Q_\pi(S, A)$ firstly, then policy is:

$$\pi_* = \arg\max_\pi Q_\pi(S, A)$$

Policy-based: optimizes for the long term reward directly:

$$\begin{aligned} J(\theta) &= \sum_{S \in \mathcal{S}} d_{\pi(\theta)}(S) V_{\pi(\theta)}(S) \\ &= \sum_{S \in \mathcal{S}} (d_{\pi(\theta)} \sum_{A \in \mathcal{A}} \pi(A|S, \theta) Q_\pi(S, A)) \end{aligned}$$

Value-based vs. Policy-based

- Value-based: for low-dimensional, discrete action space cases.
- Policy-based: for continuous action space cases.

Off-policy vs. On-policy

- Behavior strategy: guides individuals to produce actual interaction with the environment.
- Target strategy: evaluates the state or action value.

Off-policy vs. On-policy

- Off-policy: Behavior strategy and target strategy is different, e.g. Q-learning, DQN.

$$
\begin{aligned}
Q(S,A) &\leftarrow Q(S,A) + \alpha[R(S,A) + \gamma \max_{a} Q(S',a) - Q(S,A)] \\
&= (1-\alpha)Q(S,A) + \alpha[R(S,A) + \gamma \max_{a} Q(S',a)]
\end{aligned}
$$

- On-policy: Behavior strategy and target strategy is same, e.g., Sarsa:

$$
\begin{aligned}
Q(S,A) &\leftarrow Q(S,A) + \alpha[R(S,A) + \gamma Q(S',A') - Q(S,A)] \\
&= (1-\alpha)Q(S,A) + \alpha[R(S,A) + \gamma Q(S',A')]
\end{aligned}
$$

Off-policy vs. On-policy

- Off-policy:
  - A daring strategy.
  - Chooses the direction of maximizing Q when updating Q.
  - Then chooses the action again in the next state.
- On-policy:
  - A conservative strategy.
  - Planing actions for the future when updating the Q value.

5. Discrimination: Unsupervised, Supervised vs. RL

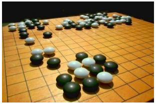
Discrimination

- Unsupervised: finds hidden structures.
- Supervised: finds mapping rules – learns from teacher(label).
- RL: finds feedback mechanisms – learns from experience.

Discrimination

- Supervised: Learns from teacher.

Next Move: “3-3”



Next Move: “2-2”

Discrimination

- RL: Learns from experience.

  First move $\Rightarrow$ ... many moves ... $\Rightarrow$ Win!

  First move $\Rightarrow$ ... many moves ... $\Rightarrow$ Lose!

6. Applications

Toolkit & Demo

- Toolkit:
  Gym: http://gym.openai.com/
  Universe:
  https://openai.com/blog/universe/
- Demo:
  Cartpolo & Flappybird.

Q & A