Reinforcement Learning

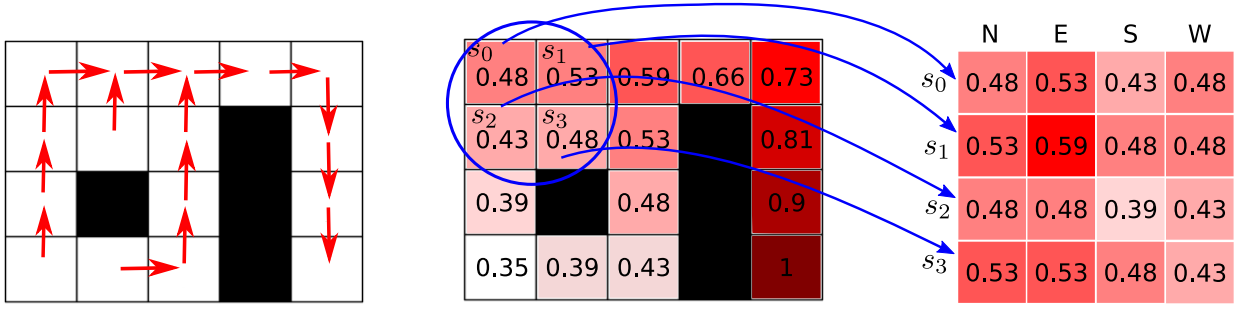
4. Model-free reinforcement Learning

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

RL algorithms

Value function and Action Value function



- The value function $V^\pi : S \to \mathbb{R}$ records the agregation of reward on the long run for each state (following policy π). It is a vector with one entry per state
- The action value function $Q^\pi : S \times A \to \mathbb{R}$ records the agregation of reward on the long run for doing each action in each state (and then following policy π). It is a matrix with one entry per state and per action

SARSA

- Reminder (TD): $V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$
- SARSA: For each observed $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$:
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
- Policy: perform exploration (e.g. ϵ-*greedy*)
- One must know the action a_{t+1}, thus constrains exploration
- On-policy method: more complex convergence proof

Singh, S. P., Jaakkola, T., Littman, M. L., & Szepesvari, C. (2000). Convergence Results for Single-Step On-Policy Reinforcement Learning Algorithms. *Machine Learning*, 38(3):287–308.

SARSA: the algorithm

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

- Taken from Sutton & Barto, 2018

Q-LEARNING

- For each observed $(s_t, a_t, r_{t+1}, s_{t+1})$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)]$$

- $\max_{a \in A} Q(s_{t+1}, a)$ instead of $Q(s_{t+1}, a_{t+1})$
- Off-policy method: no more need to know a_{t+1}
- Policy: perform exploration (e.g. ϵ-*greedy*)
- Convergence proven given infinite exploration


Watkins, C. J. C. H. (1989). *Learning with Delayed Rewards*. PhD thesis, Psychology Department, University of Cambridge, England.


Watkins, C. J. C. H. & Dayan, P. (1992) Q-learning. *Machine Learning*, 8:279–292

Q-LEARNING: the algorithm

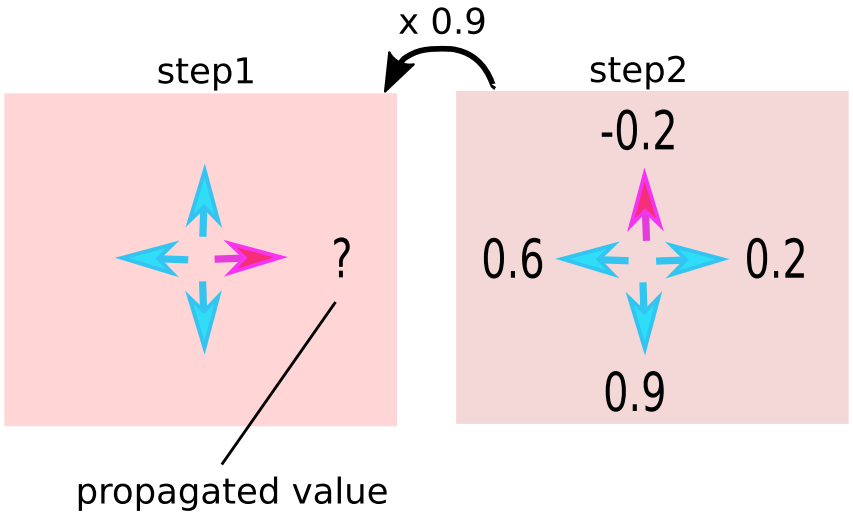
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

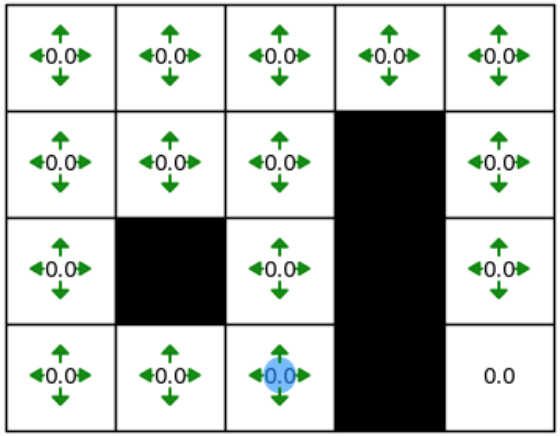
- Taken from Sutton & Barto, 2018

Difference between Q-LEARNING and SARSA



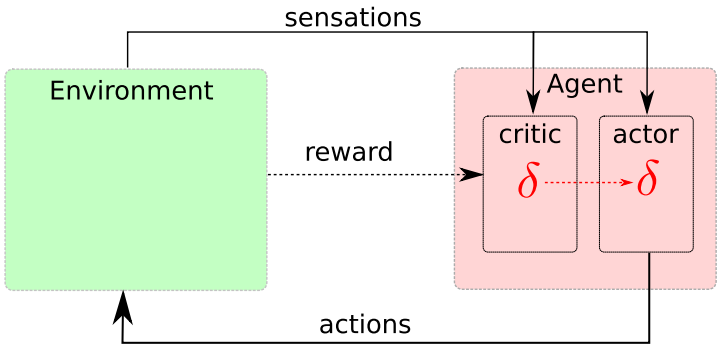
- Consider an agent taking the two pink actions
- With Q-LEARNING, the propagated value ? is $\gamma \operatorname{argmax}_a Q(s_{t+1}, a)$, thus $0.9 \times 0.9 = 0.81$
- With SARSA, it is $\gamma Q(s_{t+1}, a_{t+1})$, thus $0.9 \times -0.2 = -0.18$

Q-LEARNING in practice



- Build a states×actions table (*Q-Table*, eventually incremental)
- Initialise it (randomly or with 0 is not a good choice)
- Apply update equation after each action
- Problem: it is (very) slow

Actor-critic: Naive design

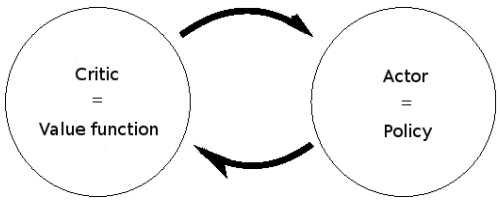


- Discrete states and actions, stochastic policy
- An update in the critic generates a local update in the actor
- Critic: compute δ and update $V(s)$ with $V_{k+1}(s) \leftarrow V_k(s) + \alpha_k \delta_k$
- Actor: $P^{\pi}_{k+1}(a|s) \leftarrow P^{\pi}_{k}(a|s) + \alpha_k \prime \delta_k$
- NB: no need for a max over actions
- NB2: one must know how to "draw" an action from a probabilistic policy (not straightforward for continuous actions)

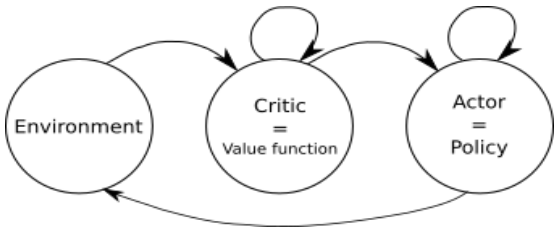
Williams, R. J. and Baird, L. (1990) A mathematical analysis of actor-critic architectures for learning optimal controls through incremental dynamic programming. In *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, pages 96–101

Dynamic Programming and Actor-Critic

Policy Iteration



Actor-Critic



- In both PI and AC, the architecture contains a representation of the value function (the critic) and the policy (the actor)
- In PI, the MDP (T and r) is known
- PI alternates two stages:
  1. Policy evaluation: update ($V(s)$) or ($Q(s,a)$) given the current policy
  2. Policy improvement: follow the value gradient
- In AC, T and r are unknown and not represented (model-free)
- Information from the environment generates updates in the critic, then in the actor

From $Q(s, a)$ to Actor-Critic

state / action	a_0	a_1	a_2	a_3
e_0	0.66	0.88*	0.81	0.73
e_1	0.73	0.63	0.9*	0.43
e_2	0.73	0.9	0.95*	0.73
e_3	0.81	0.9	1.0*	0.81
e_4	0.81	1.0*	0.81	0.9
e_5	0.9	1.0*	0.0	0.9

state	chosen action
e_0	a_1
e_1	a_2
e_2	a_2
e_3	a_2
e_4	a_1
e_5	a_1

- Given a $Q - Table$, one must determine the max at each step
- This becomes expensive if there are numerous actions
- Store the best value for each state
- Update the max by just comparing the changed value and the max
- No more maximum over actions (only in one case)
- Storing the max is equivalent to storing the policy
- Update the policy as a function of value updates

Any question?

Send mail to: Olivier.Sigaud@isir.upmc.fr

Singh, S. P., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000).

Convergence results for single-step on-policy reinforcement-learning algorithms.
Machine learning, 38(3):287–308.

Watkins, C. J. C. H. (1989).

Learning with Delayed Rewards.
PhD thesis, Psychology Department, University of Cambridge, England.

Watkins, C. J. C. H. and Dayan, P. (1992).

Q-learning.
Machine Learning, 8:279–292.

Williams, R. J. and Baird, L. (1990).

A mathematical analysis of actor-critic architectures for learning optimal controls through incremental dynamic programming.
In *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, pages 96–101. Citeseer.