# Reinforcement Learning

## 4. Model-free reinforcement Learning

Olivier Sigaud

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

## Reinforcement learning

we have an agent that knows everything about the tranzition function and reward function and uses this knowledge to iteratively compute an optimal value function from which it will derive an optimal policy

in RL, the agent does not know the tranzition and reward function in advance => exploration to figure out.

- In Dynamic Programming (planning), $T$ and $r$ are given
- Reinforcement learning goal: build $\pi^*$ without knowing $T$ and $r$
- Model-free approach: build $\pi^*$ without estimating $T$ nor $r$
- Actor-critic approach: special case of model-free
- Model-based approach: build a model of $T$ and $r$ and use it to improve the policy

we will focus on this in this class

## Incremental estimation

- Estimating the average immediate (stochastic) reward in a state $s$
- $E_k(s) = (r_1 + r_2 + \ldots + r_k)/k$
- $E_{k+1}(s) = (r_1 + r_2 + \ldots + r_k + r_{k+1})/(k+1)$
- Thus $E_{k+1}(s) = \underline{k/(k+1)E_k(s)} + r_{k+1}/(k+1)$

$\frac{(k+1)E_k(s)}{k+1} - \frac{1}{k+1}E_k(s)$

- Or $E_{k+1}(s) = (k+1)/(k+1)E_k(s) - E_k(s)/(k+1) + r_{k+1}/(k+1)$
- Or $E_{k+1}(s) = \underline{E_k(s) + 1/(k+1)[r_{k+1} - E_k(s)]}$
- Still needs to store $k$
- Can be approximated as

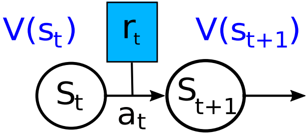
we can compute the average at the next time by just knowing the previous average and the new reward and how many times we were there (store k)

1/(k+1) alpha = learning rate

$$E_{k+1}(s) = E_k(s) + \alpha[r_{k+1} - E_k(s)] \quad (1)$$

- Converges to the true average (slower or faster depending on $\alpha$) without storing anything
- Equation (1) is everywhere in reinforcement learning

## Temporal Difference error

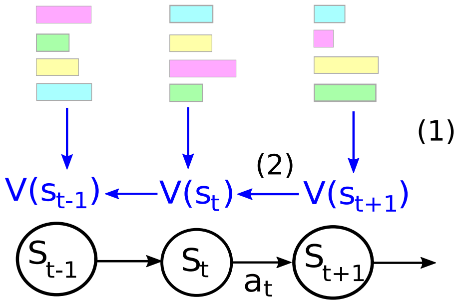



- The goal of TD methods is to estimate the value function $V(s)$
- If estimations $V(s_t)$ and $V(s_{t+1})$ were exact, we would get $V(s_t) = r_t + \gamma V(s_{t+1})$
- The approximation error is

temporal differential error $$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (2)$$

- $\delta_t$ measures the error between $V(s_t)$ and the value it should have given $r_t + \gamma V(s_{t+1})$
- If $\delta_t > 0$, $V(s_t)$ is under-evaluated, otherwise it is over-evaluated
- $V(s_t) \leftarrow V(s_t) + \alpha\delta_t$ should decrease the error (value propagation)

## Temporal Difference update rule




$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)] \quad (3)$$

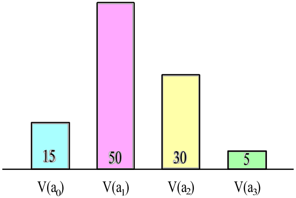
- Combines two estimation processes:
  - incremental estimation (1)
  - value propagation from $V(s_{t+1})$ to $V(s_t)$ (2)

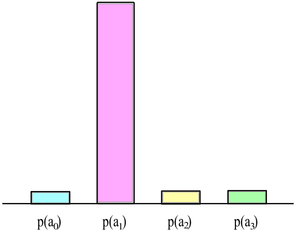
## The Policy evaluation algorithm: TD(0)

- An agent performs a sequence $s_0, a_0, r_0, \cdots, s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \cdots$
- Performs local Temporal Difference updates from $s_t$, $s_{t+1}$ and $r_t$
- Proved in 1994 provided $\epsilon$-greedy exploration

Dayan, P. & Sejnowski, T. (1994). TD(lambda) converges with probability 1. *Machine Learning*, 14(3):295–301.

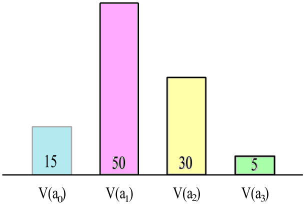
## $\epsilon$-greedy exploration

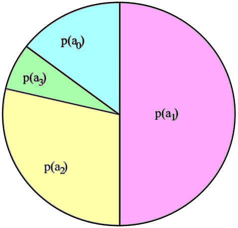



- Choose the best action with a high probability, other actions at random with low probability
- Same properties as random search
- Every state-action pair will be enough visited under an infinite horizon
- Useful for convergence proofs

## Roulette wheel

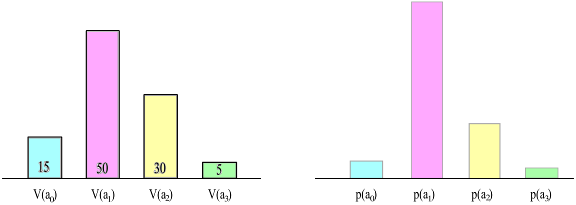



$$p(a_i) = \frac{V(a_i)}{\sum_j V(a_j)}$$

- The probability of choosing each action is proportional to its value

## Softmax exploration




$$p(a_i) = \frac{e^{\frac{V(a_i)}{\beta}}}{\sum_j e^{\frac{V(a_j)}{\beta}}}$$

- The parameter $\beta$ is called the temperature
- If $\beta \to 0$, increase contrast between values
- If $\beta \to \infty$, all actions have the same probability $\to$ random choice
- Meta-learning: tune $\beta$ dynamically (exploration/exploitation)
- More used in computational neurosciences

George Velentzas, Costas Tzafestas, and Mehdi Khamassi. (2017) Bio-inspired meta-learning for active exploration during non-stationary multi-armed bandit tasks. In *2017 Intelligent Systems Conference (IntelliSys)*, pp. 661–669. IEEE

## TD(0): limitation

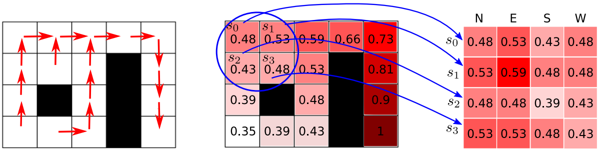
tranzition function

- TD(0) evaluates $V(s)$
- One cannot infer $\pi(s)$ from $V(s)$ without knowing $T$: one must know which $a$ leads to the best $V(s')$
- Three solutions:

  action value function

  - Q-LEARNING, SARSA: Work with $Q(s,a)$ rather than $V(s)$.
  - ACTOR-CRITIC methods: Simultaneously learn $V$ and update $\pi$
  - DYNA: Learn a model of $T$: model-based (or indirect) reinforcement learning

## 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 $\pi$). 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 $\pi$). It is a matrix with one entry per state and per action

## SARSA

temporal differential algorithm

- Reminder (TD): $V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$
- SARSA: For each observed $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$:
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
- Policy: perform exploration (e.g. $\epsilon$-*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., $\varepsilon$-greedy)
    Loop for each step of episode:
        Take action $A$, observe $R$, $S'$
        Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-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 $(s_t, a_t, r_t, s_{t+1})$:

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

- $\max_{a \in A} Q(s_{t+1}, a)$ 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. $\epsilon$-*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

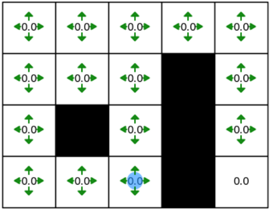
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

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., $\varepsilon$-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

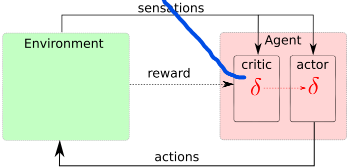
## Q-LEARNING in practice




- 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

temporal dif error

## Actor-critic: Naive design




if temporal dif error(delta)>0 => value of the state where we are is higher than we expected but it also means that the action that we just performed is better then what we thought so we should increase the probability of taking that particular action

- Discrete states and actions, stochastic policy
- An update in the critic generates a local update in the actor
- Critic: compute $\delta$ and update $V(s)$ with $V_{k+1}(s) \leftarrow V_k(s) + \alpha_k \delta_k$
- Actor: $P^{\pi}_{k+1}(a|s) \leftarrow P^{\pi}_{k}(a|s) + \alpha_k{}' \delta_k$ increase probability when delta>0, decrease when delta <0
- Link to Policy Iteration: a representation of the value (critic) and the policy (actor)
- 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)

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

we mark the max values *
=> we immediately know
where is the max

| 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 |
|---|---|
| $e_0$ | $a_1$ |
| $e_1$ | $a_2$ |
| $e_2$ | $a_2$ |
| $e_3$ | $a_2$ |
| $e_4$ | $a_1$ |
| $e_5$ | $a_1$ |

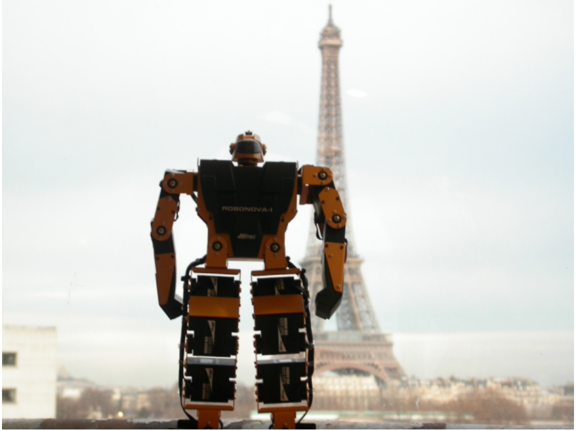
in each state, the best action

- 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

## Corresponding labs

- See https://github.com/osigaud/rl_labs_notebooks
- One notebook about model free reinforcement learning
- Implement the TD-learning algorithm, the Q-LEARNING algorithm, the SARSA algorithm and compare them
- In a separate actor-critic notebook, implement the actor-critic algorithm, using the $V$ and the $Q$ functions in the critic

## Any question?




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

Dayan, P. and Sejnowski, T.
TD(lambda) converges with probability 1.
*Machine Learning*, 14(3):295–301, 1994.

Velentzas, G., Tzafestas, C., and Khamassi, M.
Bio-inspired meta-learning for active exploration during non-stationary multi-armed bandit tasks.
In *2017 Intelligent Systems Conference (IntelliSys)*, pp. 661–669. IEEE, 2017.

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