Tabular Reinforcement Learning

Olivier Sigaud

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

Reinforcement learning

- In Dynamic Programming (planning), T and r are given
- Reinforcement learning goal: build π^* without knowing T and r
- Model-free approach: build π^* 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

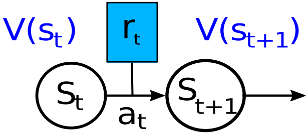
Incremental estimation

- Estimating the average immediate (stochastic) reward in a state s
- $E_k(s) = (r_1 + r_2 + ... + r_k)/k$
- $E_{k+1}(s) = (r_1 + r_2 + ... + r_k + r_{k+1})/(k+1)$
- Thus $E_{k+1}(s) = k/(k+1)E_k(s) + r_{k+1}/(k+1)$
- Or $E_{k+1}(s) = (k+1)/(k+1)E_k(s) - E_k(s)/(k+1) + r_{k+1}/(k+1)$
- Or $E_{k+1}(s) = E_k(s) + 1/(k+1)[r_{k+1} - E_k(s)]$
- Still needs to store k
- Can be approximated as

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

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

Temporal Difference error

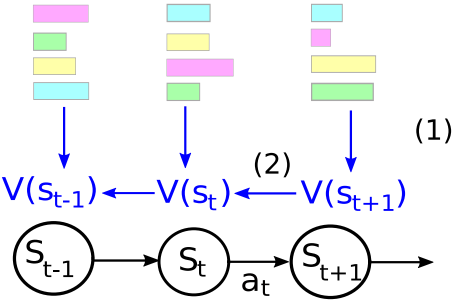


- The goal of TD methods is to estimate the value function $V(s)$
- If estimations $V(s_t)$ and $V(s_{t+1})$ were exact, we would get $V(s_t) = r_t + \gamma V(s_{t+1})$
- The approximation error is

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2}$$

- δ_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



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

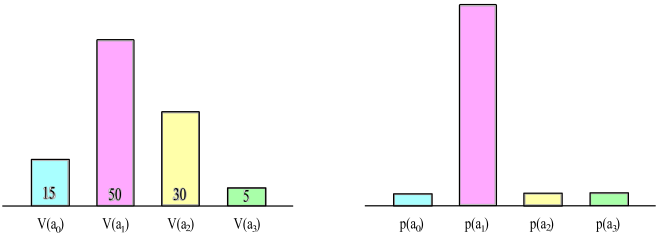
- Combines two estimation processes:
  - incremental estimation (1)
  - value propagation from $V(s_{t+1})$ to $V(s_t)$ (2)

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 ϵ-greedy exploration

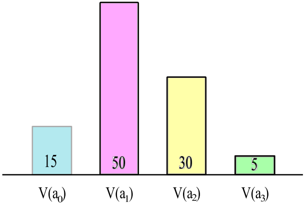
Dayan, P. & Sejnowski, T. (1994). TD(lambda) converges with probability 1. *Machine Learning*, 14(3):295–301.

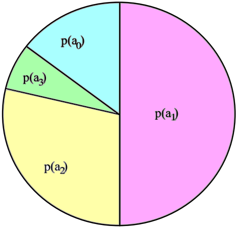
ϵ-greedy exploration



- 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

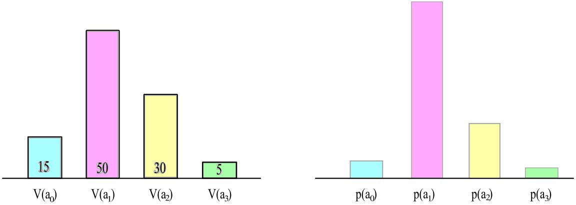




$$p(a_i) = \frac{V(a_i)}{\sum_j V(a_j)}$$

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

Softmax exploration



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

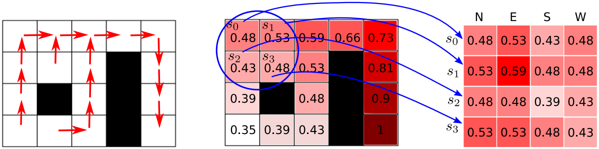
- The parameter β is called the temperature
- If $\beta \rightarrow 0$, increase contrast between values
- If $\beta \rightarrow \infty$, all actions have the same probability $\rightarrow$ random choice
- Meta-learning: tune β dynamically (exploration/exploitation)
- More used in computational neurosciences

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

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

Value function and Action Value function



- The value function $V^{\pi} : S \rightarrow \mathbb{R}$ 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 $Q^{\pi} : S \times A \rightarrow \mathbb{R}$ 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): $V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$
- 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. 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, 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. 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

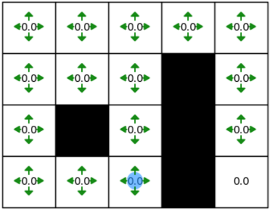
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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 $Q(\mathit{terminal}, \cdot) = 0$

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

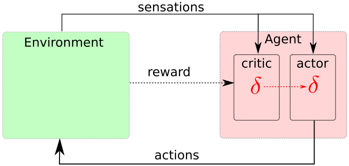
- Taken from Sutton & Barto, 2018

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: $P^{\pi}_{k+1}(a|s) \leftarrow P^{\pi}_{k}(a|s) + \alpha_k' \delta_k$
- 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

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

Maximization in RL

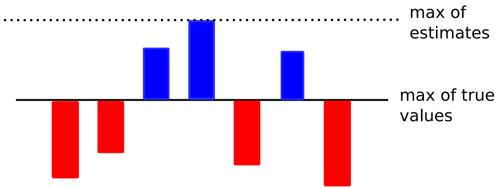
- Two maximization steps:
  - In action selection:

    $$\pi(s) \sim \operatorname*{argmax}_{a \in A} Q(s,a)$$

    might be stochastic or contain some exploration
  - In Q-LEARNING, in the value update rule

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

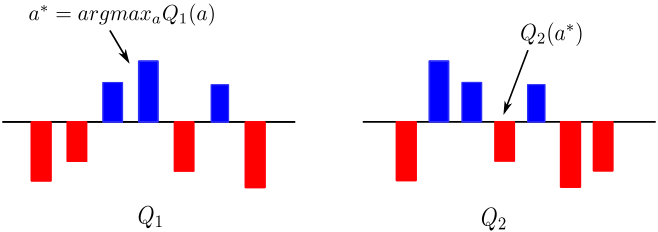
Maximization bias



- In action selection, a maximum over estimated $Q(s,a)$ is performed
- "In these algorithms, a maximum over estimated values is used implicitly as an estimate of the maximum value, which can lead to a significant positive bias."
- Example: imagine all true $Q(s,a)$ values are null

Sutton, R. S. & Barto, A. G. (2018) *Reinforcement Learning: An Introduction (Second edition)*. MIT Press

Double Q-learning



- Solution: using two Q-Tables, one for value estimation and one for action selection
- $a^* = \operatorname{argmax}_a Q_1(a)$
- $Q_2(a^*) = Q_2(\operatorname{argmax}_a Q_1(a))$ unbiased estimate of $Q(a^*)$
- $a'^* = \operatorname{argmax}_a Q_2(a)$
- $Q_1(a'^*) = Q_1(\operatorname{argmax}_a Q_2(a))$ unbiased estimate of $Q(a'^*)$
- Randomly select one of each at all steps

Van Hasselt, H. (2010) Double q-learning. *Advances in Neural Information Processing Systems*, pages 2613–2621

Double Q-LEARNING: results

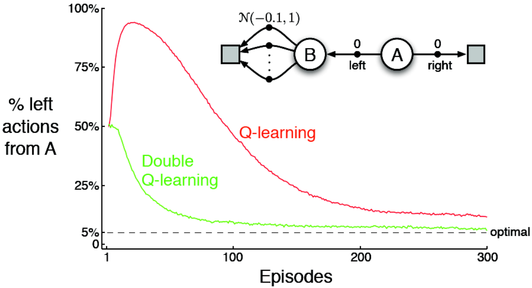
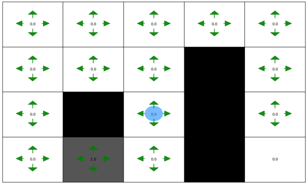


Figure 6.5: Comparison of Q-learning and Double Q-learning on a simple episodic MDP (shown inset). Q-learning initially learns to take the left action much more often than the right action, and always takes it significantly more often than the 5% minimum probability enforced by ε-greedy action selection with $\varepsilon = 0.1$. In contrast, Double Q-learning is essentially unaffected by maximization bias. These data are averaged over 10,000 runs. The initial action-value estimates were zero. Any ties in ε-greedy action selection were broken randomly.

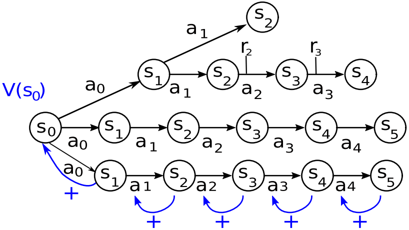
Over-estimation bias propagation



- Some initial bias cannot be prevented due to Q-Table initialization
- In Q-LEARNING, due to the max operator, the maximization bias propagates
- No propagation of under-estimation
- The same holds for DDPG without a max operator!

Fujimoto, S., van Hoof, H., & Meger, D. (2018) Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*

Monte Carlo (MC) methods



- Much used in games (Go...) to evaluate a state
- It uses the average estimation method $E_{k+1}(s) = E_k(s) + \alpha[r_{k+1} - E_k(s)]$
- Generate a lot of trajectories: $s_0, s_1, \ldots, s_N$ with observed rewards $r_0, r_1, \ldots, r_N$
- Update state values $V(s_k)$, $k = 0, \ldots, N-1$ with:

$$V(s_k) \leftarrow V(s_k) + \alpha(s_k)(r_k + r_{k+1} + \cdots + r_N - V(s_k))$$

TD vs MC

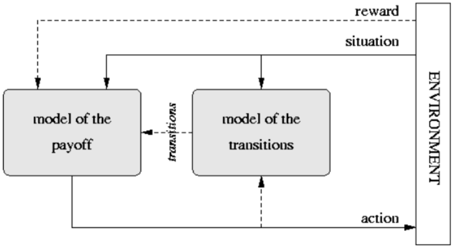
- Temporal Difference (TD) methods combine the properties of DP methods and Monte Carlo methods:
- In Monte Carlo, T and r are unknown, but the value update is global along full trajectories
- In DP, T and r are known, but the value update is local
- TD: as in DP, $V(s_t)$ is updated locally given an estimate of $V(s_{t+1})$ and T and r are unknown
- Note: Monte Carlo can be reformulated incrementally using the temporal difference δ_k update

Eligibility traces

- Goal: improve over Q-LEARNING
- Naive approach: store all (s, a) pair and back-propagate values
- Limited to finite horizon trajectories
- Speed/memory trade-off
- TD(λ), SARSA (λ) and Q(λ): more sophisticated approach to deal with infinite horizon trajectories
- A variable $e(s)$ is decayed with a factor λ after s was visited and reinitialized each time s is visited again
- TD(λ): $V(s) \leftarrow V(s) + \alpha\delta e(s)$, (similar for SARSA ($\lambda$) and Q($\lambda$)),
- If $\lambda = 0$, $e(s)$ goes to 0 immediately, thus we get TD(0), SARSA or Q-LEARNING
- TD(1) $=$ Monte Carlo...
- Eligibility traces can be seen as a combination of N-step returns for all N

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015b) High-dimensional continuous control using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*

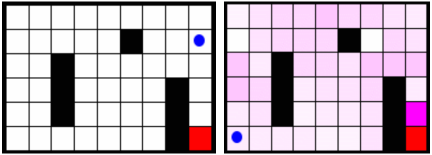
Model-based Reinforcement Learning



- General idea: planning with a learnt model of T and r is performing back-ups "in the agent's head" ([Sutton, 1990, Sutton, 1991])
- Learning T and r is an incremental self-supervised learning problem
- Several approaches:
  - Draw random transition in the model and apply TD back-ups
  - DYNA-PI, DYNA-Q, DYNA-AC
  - Better propagation: Prioritized Sweeping

Moore, A. W. & Atkeson, C. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130.

Dyna architecture and generalization

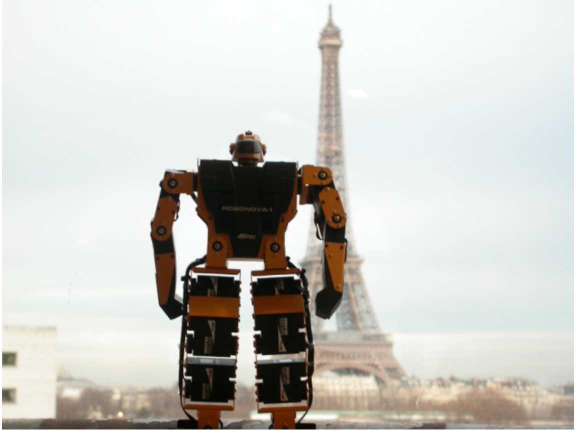


- Thanks to the model of transitions, DYNA can propagate values more often
- Problem: in the stochastic case, the model of transitions is in $card(S) \times card(S) \times card(A)$
- Usefulness of compact models
- MACS: DYNA with generalisation (Learning Classifier Systems)
- SPITI: DYNA with generalisation (Factored MDPs)

Gérard, P., Meyer, J.-A., & Sigaud, O. (2005) Combining latent learning with dynamic programming in MACS. *European Journal of Operational Research*, 160:614–637.

Degris, T., Sigaud, O., & Wuillemin, P.-H. (2006) Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems. *Proceedings of the 23rd International Conference on Machine Learning (ICML'2006)*, pages 257–264

Any question?



Send mail to: Olivier.Sigaud@upmc.fr

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

Degris, T., Sigaud, O., and Wuillemin, P.-H. (2006).
Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems.
In *Proceedings of the 23rd International Conference on Machine Learning*, pages 257–264, CMU, Pennsylvania.

Fujimoto, S., van Hoof, H., and Meger, D. (2018).
Addressing function approximation error in actor-critic methods.
In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR.

Gérard, P., Meyer, J.-A., and Sigaud, O. (2005).
Combining latent learning with dynamic programming in MACS.
European Journal of Operational Research, 160:614–637.

Moore, A. W. and Atkeson, C. (1993).
Prioritized sweeping: Reinforcement learning with less data and less real time.
Machine Learning, 13:103–130.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. (2016).
High-dimensional continuous control using generalized advantage estimation.
In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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.

Sutton, R. S. (1990).
Integrating architectures for learning, planning, and reacting based on approximating dynamic programming.
In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, San Mateo, CA. Morgan Kaufmann.

Sutton, R. S. (1991).
DYNA, an integrated architecture for learning, planning and reacting.
SIGART Bulletin, 2:160–163.

Sutton, R. S. and Barto, A. G. (2018).
Reinforcement Learning: An Introduction (Second edition).
MIT Press.

van Hasselt, H. (2010).
Double q-learning.
In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 2613–2621. Curran Associates, Inc.

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

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.