Machine Learning

Reinforcement Learning

Alberto Maria Metelli and Francesco Trovò

Reinforcement Learning For Prediction

Possible Options for Prediction

When we want to perform prediction and we do not know the environment dynamics or modeling the environment is too complex:

- Monte Carlo (first and every visit)

$$V(s_t) \leftarrow V(s_t) + \alpha(v_t - V(s_t))$$

- Temporal Difference

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

- $TD(\lambda)$ (eligibility traces)

$$V(s_t) \leftarrow V(s_t) + \alpha(v_t^\lambda - V(s_t))$$

with $v_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} v_t^{(n)}$

Exercise 9.5

Evaluate the value for the MDP with states $\mathcal{S} = \{A, B, C\}$ (C is terminal), actions $\mathcal{A} = \{h, r\}$ given the policy π and the following trajectories:

$$(A, h, 3) \to (B, r, 2) \to (B, h, 1) \to (C)$$
$$(A, h, 2) \to (A, h, 1) \to (C)$$
$$(B, r, 1) \to (A, h, 1) \to (C)$$

1. Can you tell without computing anything if by resorting to MC with every-visit and first-visit approach you will have different results?
2. Compute the values with the two aforementioned methods
3. Assume to consider a discount factor $\gamma = 1$ and compute the values by resorting to TD? Assume to start from zero values for each state and $\alpha = 0.1$

Exercise 9.6 (variant)

Evaluate the value for the MDP with states $\mathcal{S} = \{A, B, C\}$ (C is terminal), actions $\mathcal{A} = \{h, l\}$ given the policy π and the following trajectories:

$$(A, h, -1) \to (A, l, 4) \to (B, l, 1) \to (C)$$
$$(B, l, 4) \to (A, h, -3) \to (C)$$
$$(A, l, 1) \to (B, h, -2) \to (A, l, 1) \to (B, l, 1) \to (C)$$

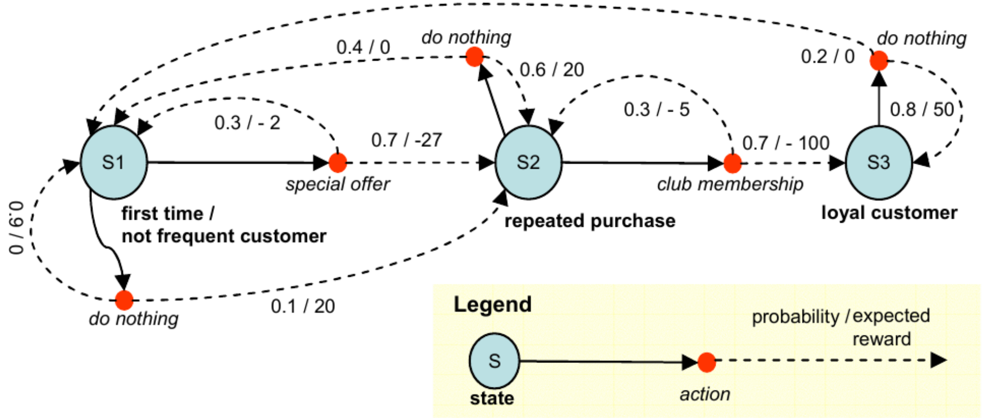
1. Compute the state-action value function $Q(A, r)$ by resorting to TD evaluation. Assume $\alpha = 0.5$, $\gamma = 1$, zero initial values.
2. Compute the state-action value function for every meaningful state-action pair by resorting to first-visit MC evaluation
3. What does the greedy policy prescribe according to the MC first-visit evaluation?
4. Assume to have performed the MC first-visit evaluation with an infinite number of trajectories from the same policy. What can we say about the optimal policy?

Reinforcement Learning For Control

Possible Options for Control

- Monte Carlo Control:
  - Policy evaluation: Monte Carlo Estimation
  - Policy improvement: ε-greedy
- SARSA:
  - Policy evaluation: Temporal Difference Estimation
  - Policy improvement: ε-greedy
- Q-learning: empirical version of Value Iteration

Example: Advertising Problem



RL Basic Elements

The elements needed to apply RL algorithms are:

- Dataset of transitions ($\{(s_n, a_n, r_n, s_{n+1})\}_{n=1}^{N}$) or model generating transitions
- Policy improvement step
- Evaluation (update) step

Transition Model

Let us model the transition model of the advertising problem from which we will get episodes used in the RL algorithms:

$$r : S \times A \to \mathbb{R}$$
$$P : S \to S$$

Especially, we need to define the generative process:

```
class Environment(object):
    ...
    def transition_model(self, a):
        ...
        return s_prime, inst_rew
```

Policy Improvement Step

The ε-greedy policy is:

```
def eps_greedy(s, Q, eps, allowed_actions):
  if np.random.rand() <= eps:
    a = % take a random action
  else:
    Q_s = Q[s, :].copy()
    Q_s[allowed_actions == 0] = - np.inf
    a = np.argmax(Q_s)
  return a
```

NB: we need to manage also the case in which the Q-values of more than one action have the same value in a state

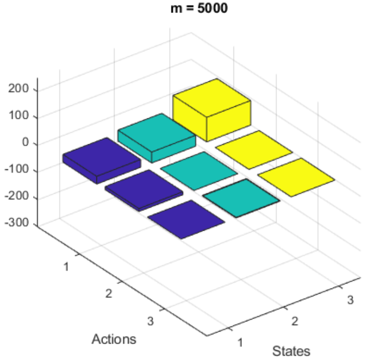
SARSA

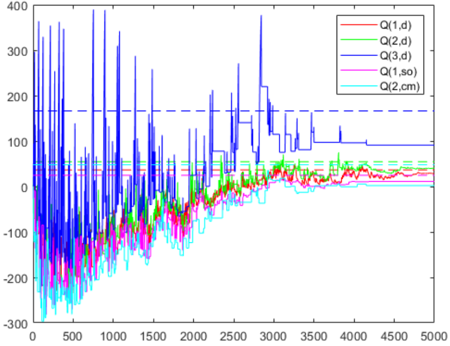
The SARSA algorithm iterates between:

- An environment step, using the transition model
- A policy improvement step, with the ϵ-greedy policy
- An evaluation step, with the TD update of the Q function:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$

SARSA - Results





Solutions Comparison

Is it a good solution?

SARSA			Exact		
40.2274	8.5816	0	36.3636	24.6818	0
67.3932	0	6.0867	54.5455	0	47.9545
79.7005	0	0	166.2338	0	0

Depending on the task we are interested in, we have a good or a poor one

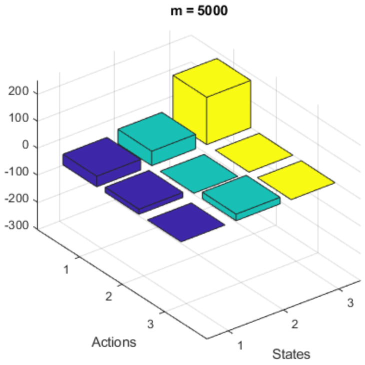
Q-learning

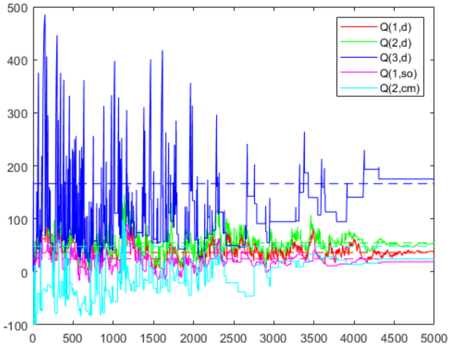
The Q-learning algorithm iterates over:

- An environment step, using the transition model
- A policy improvement step, with the ϵ-greedy policy
- An update with the Bellman optimality equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{\widetilde{a} \in \mathcal{A}} Q(s', \widetilde{a}) - Q(s,a)\big)$$

Q-learning - Results





Solutions Comparison

SARSA			Q-learning			Exact		
40.22	8.58	0	41.15	25.13	0	36.36	24.68	0
67.39	0	6.08	68.68	0	28.26	54.54	0	47.95
79.70	0	0	127.83	0	0	166.23	0	0

On-policy vs Off-policy

- With Q-learning I could have had a dataset to use for learning (off-policy learning)
- With SARSA I need to execute the ε-greedy policy at each time point (on-policy learning)

- How to use SARSA on a given transition dataset?
- Possible Solution: use importance sampling

Final Considerations

- What if I need to implement the previous two methods on a different environment?
- Replace `transition_model(s, a)` with the one corresponding to the new environment

- What other could I change in the learning process?
  - M time horizon / number of generated transitions
  - α learning rate
  - ε exploration incentive for ε-greedy