Course 9

Model-free reinforcement learning

Christophe Eloy

Exam

- Groups of 1-2 students
- One projet on Machine learning (validated by prof.)
    - Reinforcement learning (e.g., videogames, board games)
    - You can use an existing code/tutorial if you can explain/modify/tune it
- Report to be returned by January 31, 2021
    - ~10 pages
    - How/why you choose the methods you are using?
    - How did you tune the hyper parameters?
    - How/what does it learn? Comparison with alternative methods/benchmarks

Outline

- Summary on MDPs and Bellman equations
- Model-free prediction
  - Monte Carlo
  - Temporal difference
- Model-free control
  - SARSA
  - Q-learning

Markov Decision Process

- Finite set of states $\mathcal{S}$
- Finite set of actions $\mathcal{A}$
- State transition matrix

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}\left[S_{t+1} = s' \,|\, S_t = s, A_t = a\right]$$

- Reward function

$$\mathcal{R}_{s}^{a} = \mathbb{E}\left[R_{t+1} \,|\, S_t = s, A_t = a\right]$$

- Discount factor: $\gamma \in [0,1]$

Policy and value functions

- Stochastic policy

$$\pi(a \,|\, s) = \mathbb{P}\left[A_t = a \,|\, S_t = s\right]$$

- Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$

- State-value function

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \,|\, S_t = s\right]$$

- Action-value function (Q-function)

$$q_\pi(s, a) = \mathbb{E}_\pi\left[G_t \,|\, S_t = s, A_t = a\right]$$

Bellman equations

- Bellman equation for state-value and action-value functions

$$v_\pi(s) = \sum_a \pi(a \mid s) \left[\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v_\pi(s') \right]$$

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \sum_{a'} \pi(a' \mid s') q_\pi(s', a')$$

- Bellman optimality equations ($v_*(s) = \max_a q_*(s, a)$)

$$v_*(s) = \max_a \left[\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v_*(s') \right]$$

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

Model-free prediction

- Dynamic programming
- Iterative procedure to approach state-value function, given a known policy
- Monte-Carlo algorithm (high variance, no bias)

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$

- Temporal difference algorithm (lower variance, some bias)

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$

Theorem: MC and TD algorithm converges towards $v_\pi(s)$

Model-free control

- Iterative procedure to approach Q-function, given sequences S, A, R, S', A'

- Monte Carlo algorithm (on-policy)

$$Q(S,A) \leftarrow Q(S,A) + \frac{1}{N(S,A)}\left(G - Q(S,A)\right)$$

- SARSA algorithm (on-policy)

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma Q(S',A') - Q(S,A)\right)$$

- Q-learning algorithm (off-policy)

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right)$$

Theorem: These algorithms converge towards q*

DP, MC and TD

- Dynamic programming, Monte Carlo, temporal difference

- State-value function

$$V(S) \leftarrow \mathbb{E}\left[R + \gamma V(S') \,|\, S\right]$$ DP (iterative policy evalution)

$$V(S) \leftarrow V(S) + \alpha\left(G - V(S)\right)$$ MC

$$V(S) \leftarrow V(S) + \alpha\left(R + \gamma V(S') - V(S)\right)$$ TD

- Q-function

$$Q(S, A) \leftarrow \mathbb{E}\left[R + \gamma Q(S', A') \,|\, S, A\right]$$ DP

$$Q(S, A) \leftarrow Q(S, A) + \alpha\left(G - Q(S, A)\right)$$ MC

$$Q(S, A) \leftarrow Q(S, A) + \alpha\left(R + \gamma Q(S', A') - Q(S, A)\right)$$ TD (SARSA algorithm)