Lecture 5: Temporal-Difference Learning

Friday, November 26, 2021

Reinforcement Learning, Winter Term 2021/22

Joschka Boedecker, Gabriel Kalweit, Jasper Hoffmann

Neurorobotics Lab
University of Freiburg

Lecture Overview

Lecture Overview

Recap: Monte-Carlo Reinforcement Learning

Last lecture: Estimate/ Optimize the value function of an *unknown* MDP using Monte-Carlo Prediction.

- MC methods learn from episodes of *experiences*
- MC is model-free: no knowledge required about MDP dynamics
- MC learns from complete episodes based on averaging sample returns
- First-visit MC method: Estimates $v_\pi(s)$ as the average of the returns following first visits to s.
- Every-visit MC method: Estimates $v_\pi(s)$ as the average of the returns following all visits to s.

Example: Blackjack

Temporal Difference Learning

This lecture: Estimate/ optimize the value function of an *unknown* MDP using Temporal Difference Learning.

- TD is a combination of Monte Carlo and dynamic programming ideas
- Similar to MC methods, TD methods learn directly raw experiences without a dynamic model
- TD learns from *incomplete* episodes by bootstrapping
- Bootstrapping: update estimated based on other estimates without waiting for a final outcome (update a guess towards a guess)

Lecture Overview

TD Prediction

Monte Carlo Update

Update value $V(S_t)$ towards the *actual* return G_t.

$$V(s_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

α is a step-size parameter.

Simplest temporal-difference learning algorithm: $TD(0)$

Update value $V(S_t)$ towards the *estimated* return $R_{t+1} + \gamma V(S_{t+1})$.

$$V(s_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

- $R_{t+1} + \gamma V(S_{t+1})$ is called the *TD target*
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

MC Error and TD Error

If V does not change during an episode, then the MC error can be decomposed of consecutive TD errors.

$$\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
&= \delta_t + \gamma (G_{t+1} - V(S_{t+1})) \\
&= \delta_t + \gamma \delta_{t+1} + \gamma^2 (G_{t+2} - V(S_{t+2})) \\
&= \delta_t + \gamma \delta_{t+1} + \cdots + \gamma^{T-t-1} \delta_{T-1} + \gamma^{T-t} (G_T - V(S_T)) \\
&= \delta_t + \gamma \delta_{t+1} + \cdots + \gamma^{T-t-1} \delta_{T-1} + \gamma^{T-t} (0 - 0)) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k
\end{aligned}$$

TD Prediction

Tabular TD(0) for estimating v_π

```
Input: the policy $\pi$ to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    $A \leftarrow$ action given by $\pi$ for $S$
    Take action $A$, observe $R$, $S'$
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until $S$ is terminal
```

Driving Home Example

State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

TD Prediction

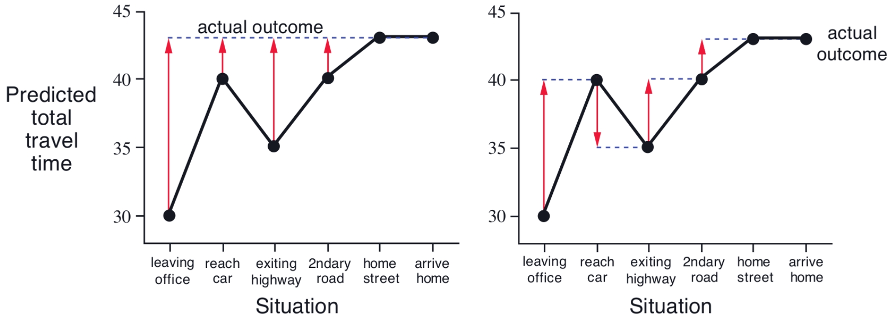


Figure 6.1: Changes recommended in the driving home example by Monte Carlo methods (left) and TD methods (right).

MC vs TD

- TD can learn online after every step, MC has to wait for the final outcome/return
- TD can even learn without ever getting a *final* outcome, which is especially important for infinite horizon tasks
- The return G_t depends on many random actions, transitions and rewards, the TD-target depends on one random action, transition and reward
- Therefore, the TD-target has lower *variance* than the return
- But the TD-target is a *biased* estimate of v_π
- This is known as the bias/variance trade-off

MC vs TD

- MC and TD converge if every state and every action are visited an infinite number of times
- What about finite experience?

Imagine two states, A and B, and the following transitions:

A,0,B,0	B,1
B,1	B,1
B,1	B,1
B,1	B,0

What are the values of A and B given this data?

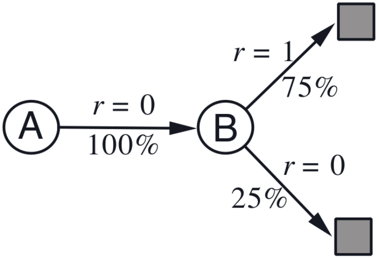
MC vs TD

- W.r.t. B, the process terminated immediately 6/8 times with a return of 1, 0 otherwise
- Thus, it is reasonable to assume a value of 0.75
- What about A?

MC vs TD

- W.r.t. B, the process terminated immediately 6/8 times with a return of 1, 0 otherwise
- Thus, it is reasonable to assume a value of 0.75
- What about A?

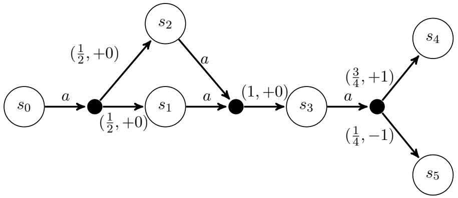
A led in all cases to B. Thus, A could have a value of 0.75 as well. This answer is based on first modelling the Markov Process and then computing the values given the model. TD is leading to this value. MC gives a value of 0 – which is also the solution with 0 MSE on the given data. One can assume, however, that the former gives lower error on *future* data.



MC vs TD

- Batch MC converges to the solution with minimum MSE on the observed returns
- Batch TD converges to the solution of the maximum-likelihood Markov model
- Given this model, we can compute the estimate of the value-function that would be exactly correct, if the model were exactly correct
- This is called the *Certainty Equivalence*

MC vs TD



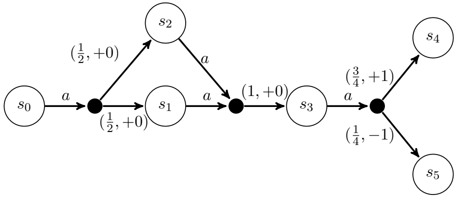
Assume that the agent encounters the following set of trajectories at every iteration i (where $i \mod 4 = 0$):

$$\begin{array}{rlllllll}
t_i: & s_0 & \rightarrow & s_1 & \rightarrow & s_3 & \rightarrow & s_4 \\
t_{i+1}: & s_0 & \rightarrow & s_1 & \rightarrow & s_3 & \rightarrow & s_4 \\
t_{i+2}: & s_0 & \rightarrow & s_1 & \rightarrow & s_3 & \rightarrow & s_4 \\
t_{i+3}: & s_0 & \rightarrow & s_2 & \rightarrow & s_3 & \rightarrow & s_5
\end{array}$$

Question (5 min)

Given these trajectories, explain why TD-learning is better fitted to estimate the value function compared to MC. Assume no discount and that the value function is initialized with zeros. To which value function is MC going to converge, given a suitable learning rate α? What about TD?

MC vs TD



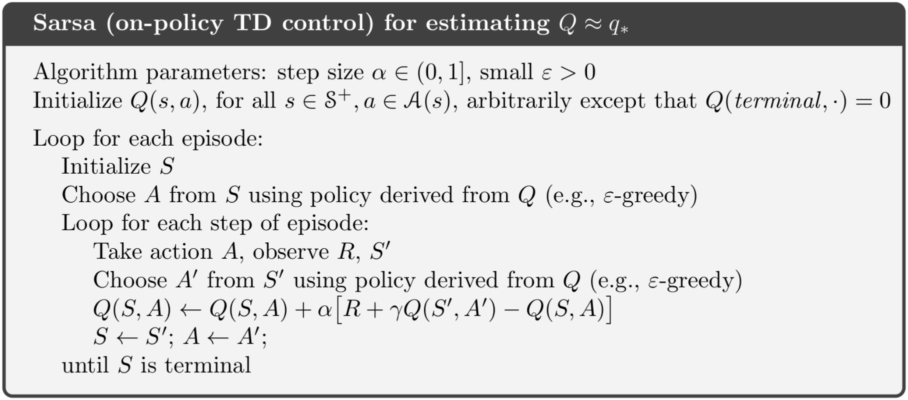
Solution

MC always takes the full return to update its values. Therefore, s_1 only updates on return $+1$, whereas s_2 only updates on return -1. TD takes this into account due to bootstrapping. MC converges to: $v(s_0) = v(s_3) = 0.5$, $v(s_1) = v(s_4) = +1$ and $v(s_2) = v(s_5) = -1$. TD converges to the true value function $v(s_0) = v(s_1) = v(s_2) = v(s_3) = 0.5$ and $v(s_4) = v(s_5) = 0$, since $\frac{3}{4}$ trajectories end with a return of $+1$ and $\frac{1}{4}$ with a return of -1 – which corresponds to the true distribution of the MDP.

Lecture Overview

SARSA

- SARSA: State, Action, Reward, State, Action
- Why is it considered an on-policy method?

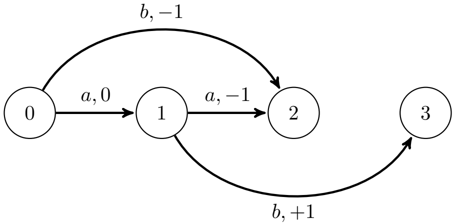


Sarsa (on-policy TD control) for estimating $Q \approx q_*$

```
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
```

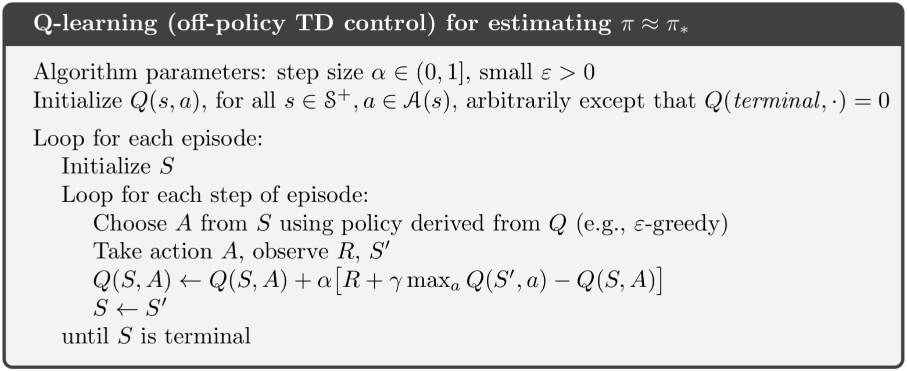
SARSA Example



$$\begin{array}{lccccc}
\mathsf{traj}_1: & 0 & \rightarrow & 1 & \rightarrow & 2 \\
\mathsf{traj}_2: & 0 & \rightarrow & 1 & \rightarrow & 3 \\
\mathsf{traj}_3: & 0 & \rightarrow & 1 & \rightarrow & 2
\end{array}$$

Q-learning

- Why is it considered an off-policy method?



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
```

Q-learning Example

$$\begin{tikzcd}
0 \arrow[r, "{a, 0}"] \arrow[rr, bend left=50, "{b, -1}"] & 1 \arrow[r, "{a, -1}"] \arrow[rr, bend right=50, "{b, +1}"'] & 2 & 3
\end{tikzcd}$$

$$\begin{array}{lccccc}
\mathsf{traj}_1: & 0 & \rightarrow & 1 & \rightarrow & 2 \\
\mathsf{traj}_2: & 0 & \rightarrow & 1 & \rightarrow & 3 \\
\mathsf{traj}_3: & 0 & \rightarrow & 1 & \rightarrow & 2
\end{array}$$

Expected SARSA

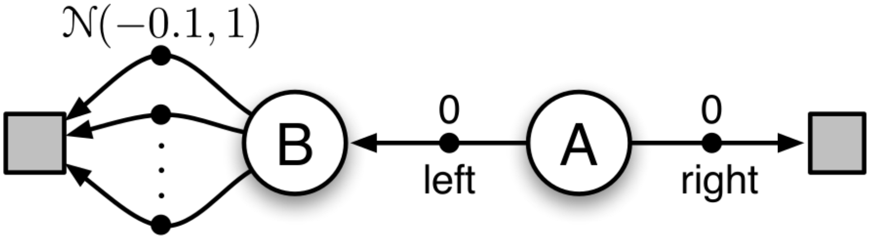
- Similar to Q-learning, but uses the expectation
- *Expected SARSA*-Update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \underbrace{\mathbb{E}_\pi[Q(S_{t+1}, A_{t+1})|S_{t+1}]}_{\sum_a \pi(a|S_{t+1})Q(S_{t+1}, A_{t+1})} - Q(S_t, A_t)]$$

- If π is greedy, but behaviour comes from an exploratory policy, then *Expected SARSA* is equivalent to Q-learning and is thus a generalization of this concept

Overestimation Bias

- In all control algorithms so far, the target policy is created by the *maximization* of a value-function
- We thus consider the maximum over estimated values as an estimate of the maximum value
- This can lead to the so-called *overestimation bias*



Overestimation Bias

- Recall the Q-learning target: $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- Imagine two random variables X_1 and X_2:

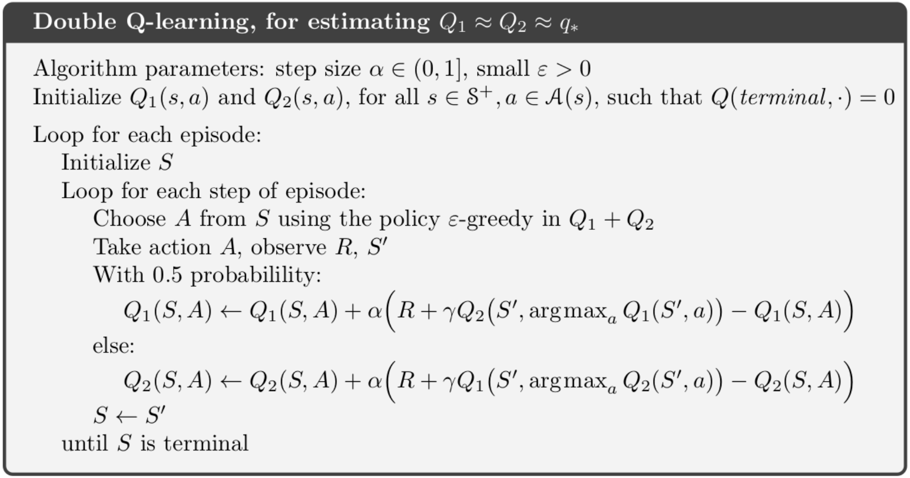
$$\mathbb{E}[\max(X_1, X_2)] \geq \max(\mathbb{E}[X_1], \mathbb{E}[X_2])$$

- $Q(S_{t+1}, a)$ is not perfect – it can be *noisy*:

$$\max_a Q(S_{t+1}, a) = \overbrace{Q(S_{t+1}, \underbrace{\arg\max_a Q(S_{t+1}, a)}_{\text{action comes from } Q})}^{\text{value comes from } Q}$$

- If the noise in these is decorrelated, the problem goes away!

Double Q-learning

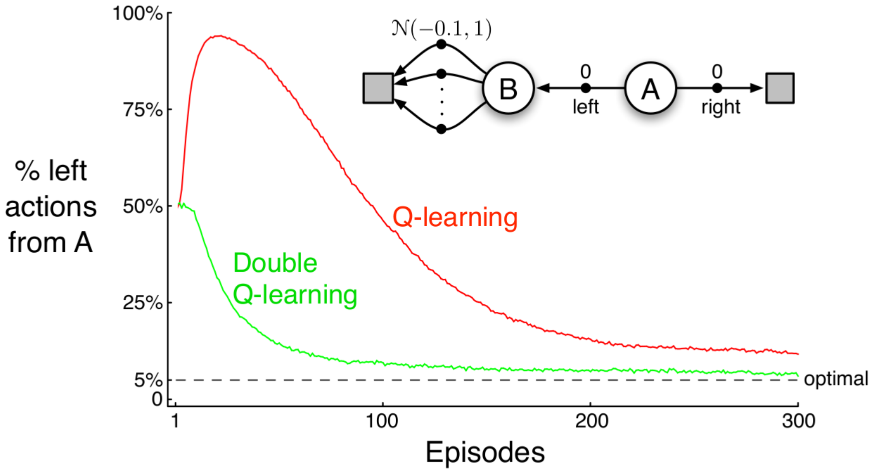


Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

```
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        Choose $A$ from $S$ using the policy $\varepsilon$-greedy in $Q_1 + Q_2$
        Take action $A$, observe $R$, $S'$
        With 0.5 probabilility:
            $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \operatorname{argmax}_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
        else:
            $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \operatorname{argmax}_a Q_2(S', a)\big) - Q_2(S, A)\Big)$
        $S \leftarrow S'$
    until $S$ is terminal
```

Double Q-learning



Lecture Overview

Summary by Learning Goals

Having heard this lecture, you can now…

- Explain the differences between MC and TD
- Use TD to predict a value function
- Apply on-policy TD-control
- Apply off-policy TD-control
- Explain the overestimation bias