Dynamic Programming and Reinforcement Learning

Week 5a: Temporal Difference Algorithms & Q-Learning 2

Rainer Schlosser und Alexander Kastius

Enterprise Platform and Integration Concepts

16.05.22

Recap

BI, VI, PI, ADP

- **Backward Induction (BI)**: For finite horizon MDPs, make use of the knowledge about the horizon.
- **Value Iteration (VI), Policy Iteration (PI):** For infinite horizon MDPs, make use of full knowledge about the process. Events, state transitions, reward function etc. are known to the developer.
- **Approximate Dynamic Programming (ADP):** Still assumes full knowledge, but prioritizes states by their occurrence in the simulation.

Finite Horizon vs. Infinite Horizon

- **Finite Horizon MDPs:** Have a time *T* after which the process ends. Knowledge about this can drastically improve solution time by using backward induction.
- **Infinite Horizon MDPs:** Have no fixed length, which makes BI impossible. Require either VI, PI or ADP to be solved successfully.

SARSA

1. Observe s_t, choose a_t according to the current policy
2. Observe r_t, s_{t+1}, choose a_{t+1} according to the current policy
3. Update the Q-value estimate:

$$Q_t(s_t, a_t) \leftarrow \eta_t(r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)) + Q_t(s_t, a_t)$$

or

$$Q_t(s_t, a_t) \leftarrow \eta_t(r_t + \gamma Q_t(s_{t+1}, a_{t+1})) + (1 - \eta_t) Q_t(s_t, a_t)$$

4. Repeat from 1. with each new transition, reduce η_t over time (for example $\eta_t = \frac{1}{t}$)

	$s^{(1)}$	$s^{(2)}$	$s^{(3)}$	$s^{(4)}$	…
$a^{(1)}$	1.2	3.2	2.2	3.2	
$a^{(2)}$	1.1	3.1	6.2	5.2	
$a^{(3)}$	1.4	4.0	1.7	6.7	
$a^{(4)}$	2.8	0.2	4.3	0.2	
…					

$$\left(s^{(1)}, a^{(2)}, 1, s^{(2)}, a^{(3)}\right), \gamma = 0.99$$

$$target \leftarrow 1 + 0.99 * 4.0$$

Q-Learning

1. Observe s_t, choose a_t according to the current policy
2. Observe r_t, s_{t+1} (a_{t+1} is not relevant here)
3. Update the Q-value estimate:

$$Q_t(s_t, a_t) \leftarrow \eta_t(r_t + \gamma \max_{\boldsymbol{a} \in \boldsymbol{A}} \boldsymbol{Q_t}(\boldsymbol{s_{t+1}}, \boldsymbol{a}) - Q_t(s_t, a_t)) + Q_t(s_t, a_t)$$

4. Repeat from 1. with each new transition, reduce η_t over time (for example $\eta_t = \frac{1}{t}$)

Policy at each point in time can be ϵ-greedy. Convergence is guaranteed if all combinations of s and a are revisited in endless time.

What is the difference of the Q-values in comparison to SARSA? Are we learning something different here?

Overestimation Bias

Problem

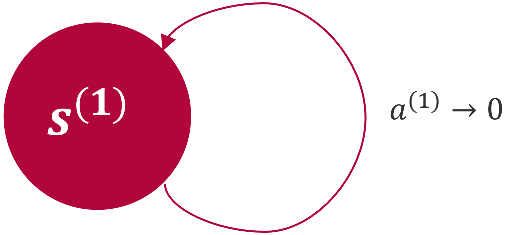
Q-learning shows the tendency to overestimate Q-values.

This is caused by the max-operator.

This can be displayed in a theoretical setup, but it is present in more complex scenarios as well.

Solution

Keep second table for the Q-values, alternate between updating both tables.



Randomly initialized estimate

	$s^{(1)}$
$a^{(1)}$	?

Actual rewards

	$s^{(1)}$
$a^{(1)}$	0

Double Q-Learning

Initialize 2 S/A-tables ($Q^{(0)}$ and $Q^{(1)}$)!

Initialize marker for current table $n = 0$

1. Observe s_t, choose a_t according to ϵ-greedy based on $Q^{(n)}$
2. Observe r_t, s_{t+1}
3. Update the Q-value estimate:

$$Q_t^{(n)}(s_t, a_t) \leftarrow \eta_t(r_t + \gamma \boldsymbol{Q}_t^{(1-n)}\left(\boldsymbol{s}_{t+1}, \underset{a \in A}{\operatorname{argmax}}\, \boldsymbol{Q}_t^{(n)}(\boldsymbol{s}_{t+1}, \boldsymbol{a})\right) - Q_t^{(n)}(s_t, a_t)) + Q_t^{(n)}(s_t, a_t)$$

4. $\mathrm{n} \leftarrow 1 - \mathrm{n}$
5. Repeat from 1. with each new transition, reduce η_t over time (for example by setting $\eta_t = \frac{1}{t}$)

Q-Learning Inefficient?

1. Observe s_t, choose a_t according to the current policy — Store **state**
2. Observe r_t, s_{t+1}, choose a_{t+1} according to the current policy — Store **reward, state, action**
3. Update the Q-value estimate: — Use them for update

$$Q_t(s_t, a_t) \leftarrow \eta_t(r_t + \gamma \max_{\boldsymbol{a} \in \boldsymbol{A}} \boldsymbol{Q_t}(\boldsymbol{s_{t+1}}, \boldsymbol{a}) - Q_t(s_t, a_t)) + Q_t(s_t, a_t)$$

4. Repeat from 1. with each new transition, reduce η_t over time (for example $\eta_t = \frac{1}{t}$) — Forget all of them again!

Policy at each point in time can be ϵ-greedy.
Convergence is guaranteed if all combinations of s and a are revisited in endless time.

We have more memory available, why shouldn't we use it?

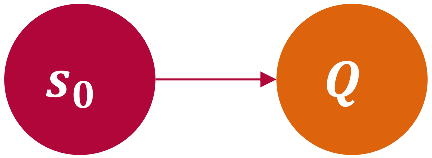
Temporal Difference Learning with (n)-Step Horizon (or simply: TD(n))

Hasso Plattner Institut

$$G_t = r_t + \gamma G_{t+1} = r_t + yQ(s_{t+1}, a_{t+1})\ \forall t$$

- The computation for now assumes that $G_{t+1} = Q(s, a)$, with a being either the action under the current policy (SARSA) or under the optimal policy (QL).
- We could easily store more state transitions and compute a sample of G_t for a long horizon.
- We assume that this converges faster, as we do rely less on possibly very wrong estimations at the beginning of the learning process.

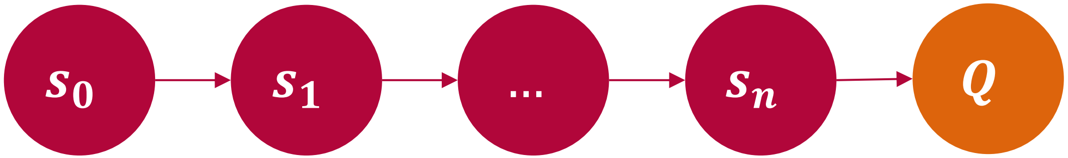
Current Backup Diagram for SARSA:



Temporal Difference Learning with (n)-Step Horizon (or simply: TD(n))

- The computation for now assumes that $G_{t+1} = Q(s,a)$, with a being either the action under the current policy (SARSA) or under the optimal policy (QL).
- We could easily store more state transitions and compute a sample of G_t for a long horizon.
- We assume that this converges faster, as we do rely less on possibly very wrong estimations at the beginning of the learning process.

New Backup Diagram for *n*-step SARSA:



TD(n) - SARSA

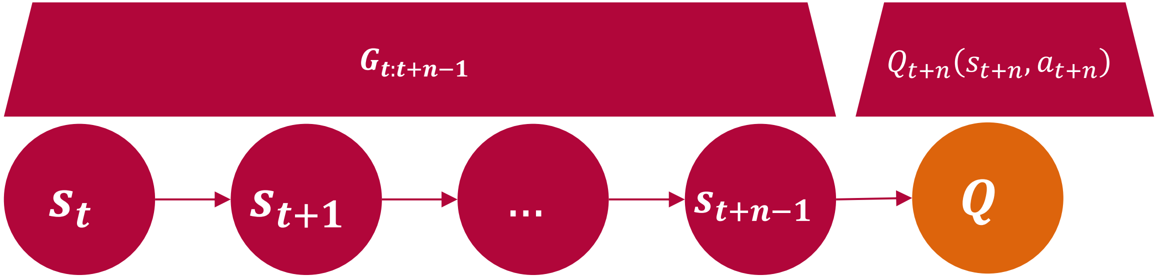
New $G_{t:n}$ def. with limited horizon:

$$G_{t:t+n} = \sum_{i=0}^{n} \gamma^i r_{t+i}$$

$G_{t:n}$ is the cumulated discounted reward under a fixed horizon n starting from step t.

New Bellman-error taking G_t into account with limited horizon:

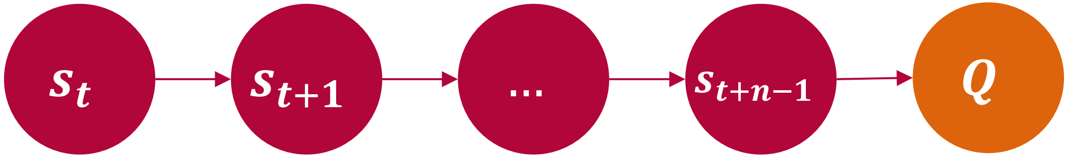
$$Q_{t+n}(s_t, a_t) \leftarrow \eta_t(G_{t:t+n-1} + \gamma^n Q_{t+n}(s_{t+n}, a_{t+n}) - Q_{t+n}(s_t, a_t)) + Q_{t+n}(s_t, a_t)$$



TD(n) – Off-Policy?

Conceptual Problems with TD(n) and Off-Policy Algorithms?

This backup here is strictly on-policy for now, as it incorporates multiple decisions performed by the policy under assessment!



TD(0) off-Policy/QL incorporated only the what-if element of the Q-value, which allowed us to exchange the future part easily. This is not the case anymore.

Solution?

Importance sampling according to the difference between the 2 policies!

Importance sampling for Off-Policy TD(n)

Idea: Find a factor that describes whether the trajectory under assessment would have been chosen by the policy we want to evaluate!

„New‟ concept: **Non-deterministic policies**

- We already use them!
- $\pi(a \mid s)$ = Probability of choosing a given s.
- In ϵ-greedy policies, this probability is $\frac{\epsilon}{|A|}$ for every non-optimal action according to our Q-table and $(1-\epsilon)+\frac{\epsilon}{|A|}$ for the optimal action.

Importance Sampling for Off-Policy TD(n)

Idea: Find a factor that describes whether the trajectory under assessment would have been chosen by the policy we want to evaluate!

For now, assume: π is the policy that we want to evaluate and b is the policy that we actually run.

In QL:

b = Policy with exploration

π = Policy for greedy exploitation

What‘s the meaning of:

$$\frac{\pi(a_t|s_t)}{b(a_t|s_t)}$$

?

Importance Sampling for Off-Policy TD(n)

Idea: Find a factor that describes whether the trajectory under assessment would have been chosen by the policy we want to evaluate!

For now, assume: π is the policy that we want to evaluate and b is the policy that we actually run.

In QL:

b = Policy with exploration

π = Policy for greedy exploitation

What‘s the meaning of:

$$\frac{\pi(a_t|s_t)}{b(a_t|s_t)}$$

This is larger than one, if the action a_t would have been chosen by the greedy policy with a larger probability!

Importance Sampling for Off-Policy TD(n)

Idea: Find a factor that describes whether the trajectory under assessment would have been chosen by the policy we want to evaluate!

Now we need such a measure for the whole n-step trajectory.

Idea: Compute this factor over all steps in the trajectory

$$\rho_{t:t+n} = \prod_{i=t}^{n} \frac{\pi(a_i|s_i)}{b(a_i|s_i)}$$

This demarks the relative importance of the trajectory from t to t+n under the assumption that a different policy should be evaluated.

And then we weight the update accordingly!

$$Q_{t+n}(s_t, a_t) \leftarrow \eta_t \boldsymbol{\rho_{t+1:t+n-1}}(G_{t:n-1} + \gamma^n Q_{t+n}(s_{t+n}, a_{t+n}) - Q_{t+n}(s_t, a_t)) + Q_{t+n}(s_t, a_t)$$

Recall

Hasso Plattner Institut

QL and SARSA Implementations

Insights in implementation of QL and SARSA.

Double Q-Learning

Easy to implement improvement to avoid overestimation bias.

On-Policy TD(n)

Makes more efficient use of the available data by computing targets with n-horizon.

Off-Policy TD(n)/QL with Horizon

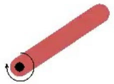
Takes the idea of On-Policy TD(n) to Q-Learning.

Yet Unsolved Issues?

Still unsolved:

- State Space Complexity
  - Many Dimensions
  - Continuous Values
- Current methods require discretization and become intractable at some point

- Continuous Control
  - Action Space might consist of continuous values as well
  - Can be discretized sometimes, which prevents us from finding the actual optimal policy



Schedule

Week	Dates	Topic
1	April 21	Introduction
2	April 25/28	Finite + Infinite Time MDPs
3	May 2/5	Approximate Dynamic Programming (ADP) + DP Exercise
4	May 12	Q-Learning (QL) (not Mon May 9)
5	May 16/19	**Q-Learning Extensions and Deep Q-Networks**
6	May 23	DQN Extensions (not Thu May 26 “Himmelfahrt”)
7	May 30/June 2	Policy Gradient Algorithms
8	June 9	Project Assignments (not Mon June 6 “Pfingstmontag”)
…		