UPPSALA
UNIVERSITET

Reinforcement Learning

Lecture 5 - Model-free control

Per Mattsson

2022

Department of Information Technology

Repetition

Important concepts

UPPSALA UNIVERSITET

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.
- **Policy:** $\pi(a|s)$ (For deterministic policy also $a = \pi(s)$.)
- **The return:** $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
- **State-value function:**

  Expected return when starting in s and following policy π,

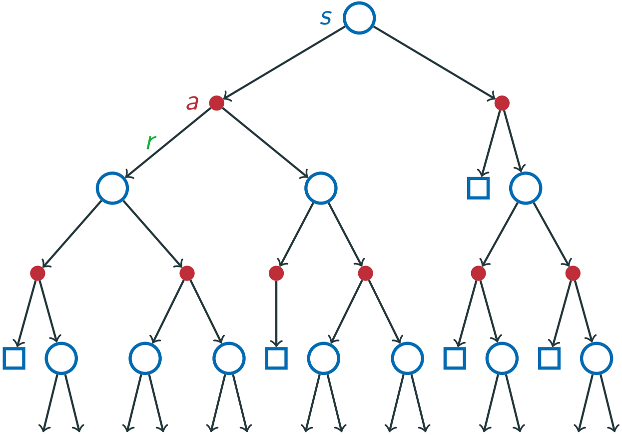
$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right].$$

- **Action-value function:**

  Expected return when starting in s, taking action a and *then* follow π,

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

Markov Decision Process Tree



- **Bellman equation for state-values:**

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right]$$

- **Bellman equation for action-values:**

$$q_\pi(s, a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right]$$

- **Bellman optimality equation:**

$$v_*(s) = \max_a q_*(s, a) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]$$

- **Optimal policy:** Act greedily w.r.t $v_*(s)$

$$\pi_*(s) = \arg\max_a q_*(s, a) = \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]$$

- **Policy evaluation:** Evaluate $v_\pi(s)$ for all s, using e.g., iterative policy evaluation:

$$V(s) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) | S_t = s\right].$$

- **Policy improvement:** Find greedy policy w.r.t $v_\pi(s)$,

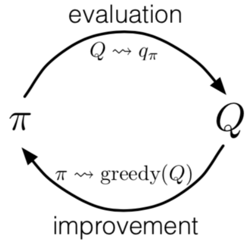
$$\pi'(s) = \arg\max_a q_\pi(s, a)$$

where $q_\pi(s, a) = \sum_{r,s'} p(s', r | s, a) \left[r + \gamma v_\pi(s')\right]$

- **Value iteration:** Evaluate $v_*(s)$ using

$$V(s) \leftarrow \max_a \mathbb{E}\left[R_{t+1} + \gamma V(S_{t+1}) | S_t = s, A_t = a\right]$$

Policy Iteration:



Repetition: Model-free prediction

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

- Policy evaluation (prediction) using only experience.
- **Monte-Carlo (MC):** (for episodic tasks)

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t)).$$

- **Temporal differences (TD):**

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

- Can we do policy improvement using only experience?

Model-free control

How to do policy improvement?

- Greedy policy improvement w.r.t $v_\pi(s)$:

$$\pi'(s) = \arg\max_a \sum_{r,s'} p(s', r | s, a)[r + \gamma v_\pi(s')].$$

  This requires the model $p(s', r | s, a)$.

- Greedy policy improvement w.r.t $q_\pi(s, a)$:

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

  We do not need the model!

- **Idea 1:** Estimate $q_\pi(s, a)$ instead of $v_\pi(s)$.

Example: Can we learn enough from greedy actions?

Which door gives most reward?

- **Initial:** $Q(\text{left}) = Q(\text{right}) = 0$.
- You open left and get reward 2:

$$Q(\text{left}) = 2, \quad Q(\text{right}) = 0$$

- You open left and get reward 0:

$$Q(\text{left}) = 1, \quad Q(\text{right}) = 0$$

- You open left and get reward 4:

$$Q(\text{left}) = 3, \quad Q(\text{right}) = 0$$

We never learn about what happens if we open the right door!

Idea 2: Make sure that we continue to explore different options!

- Trade-off between exploiting current knowledge and exploring new options!
- **Possible solution:** Ensure that all actions have a non-zero probability.
- **ε-soft policy:** If $\pi(a|s) \geq \frac{\varepsilon}{|\mathcal{A}|}$ for all a and s.

ε-greedy w.r.t $q_\pi(s, a)$:

- With probability $1 - \varepsilon$ choose a greedy action $\arg\max_a q_\pi(s, a)$.
- With probability ε choose an action at random.

Policy Improvement Theorem

For any ε-soft policy π, the ε-greedy policy π' w.r.t q_π is an improvement, i.e.

$$v_{\pi'}(s) \geq v_\pi(s), \quad \text{for all } s \in \mathcal{S}$$

- **Conclusion:** Policy improvement with ε-greedy policies will converge to the best ε-soft policy.

On-policy vs off-policy learning

UPPSALA UNIVERSITET

On-policy learning

- "Learn on the job".
- Estimate $q_\pi(s, a)$ by running the policy π.

Off-policy learning

- "Look over someone's shoulder".
- Estimate $q_\pi(s, a)$ while running a different policy μ.
- For example: Learn about $q_*(s, a)$ (optimal q-function), while running a policy with more exploration.

Monte-Carlo Control

UPPSALA UNIVERSITET

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

- Use the policy to collect trajectories

$$S_0, A_0, R_1, S_1, A_1, R_2, \cdots, S_T.$$

- **Estimating state-value function:**
  - Compute the average over all returns seen from each state.
  - Incremental update:

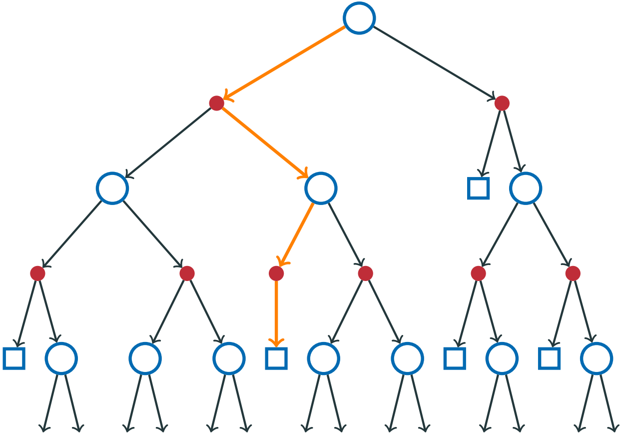
$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t)).$$

- **Estimating action-value function:**
  - Compute the average over all returns seen from each state/action-pairs.
  - Incremental update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t)).$$

Monte Carlo Backup Diagram

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t)).$$



Exploration needed!

Estimation of state-values: $V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$.

- Converges to $v_\pi(s)$ as $N(s) \to \infty$.

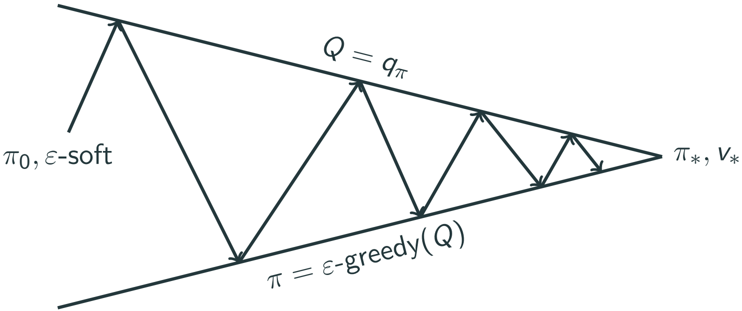
Estimation of action-values: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$.

- Converges to $q_\pi(s, a)$ as $N(s, a) \to \infty$.
- However if $\pi(a|s) = 0$ for some s and a then we will not learn this action-value!
- ε-soft policies guarantee that $\pi(a|s) > 0$ for all s and a!

So, as long as we use a ε-soft policy $Q(s, a)$ will converge to $q_\pi(s, a)$ as the number of sampled episodes goes to ∞.

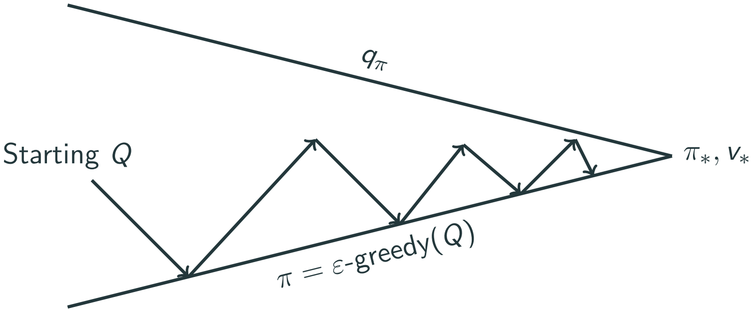
Time for policy improvement!

Monte-Carlo Policy Iteration



- **Policy Evaluation:** Monte-Carlo evaluation to get $Q = q_\pi$.
- **Policy Improvement:** Let new π be ε-greedy w.r.t q_π.
- Will converge to the best ε-soft policy.
- We need infinitely many episodes to guarantee $Q = q_\pi$ – not possible in practice.

Monte-Carlo Control



At every episode:

- **Policy evaluation:** Use MC to update Q.
- **Policy improvement:** Let new π be ε-greedy w.r.t Q.
- **On-policy:** We always update Q towards q_π for the current policy.

Monte Carlo Control

1. Initialize Q (e.g. $Q(s, a) = 0$ for all s and a) and let $\pi = \varepsilon$-greedy(Q).
2. Sample episode using π: $S_0, A_0, R_1, \cdots, S_T$.
3. For each state S_t and action A_t in the episode

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

4. Improve policy: $\pi \leftarrow \varepsilon$-greedy$(Q)$.
5. Go to step 2.

Exploration

- (If we) converge we get the best policy among the ε-soft policies.
- Can gradually reduce ε (but not to fast) towards zero, in order to converge to optimal policy.
- After training we can remove exploration by setting $\varepsilon = 0$ (thus using the greedy policy w.r.t estimated Q).

SARSA

MC vs TD

TD-prediction has several advantages over MC-prediction:

- Lower variance.
- Can run online (without waiting to end of episode)
- Can use incomplete sequences.

TD-control aka SARSA:

- Apply TD to $q_\pi(s, a)$.
- Use ε-greedy policy improvements.
- Can now update every time-step!

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

Estimating state-values:

Given $\{S_t, R_{t+1}, S_{t+1}\} \sim \pi$ the update is

$$V(S_t) \leftarrow V(S_t) + \alpha \left(\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{Target}} - V(S_t) \right).$$

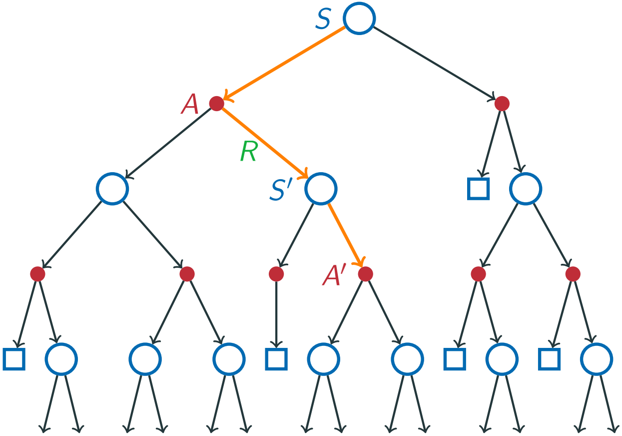
Estimating action-values (SARSA):

Given $\{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}\} \sim \pi$ the update is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(\underbrace{R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})}_{\text{Target}} - Q(S_t, A_t) \right)$$

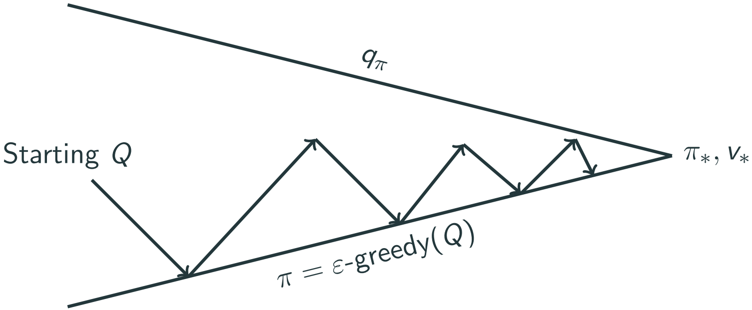
SARSA backup diagram

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



As for MC we can use ε-greedy policies to ensure that all actions are explored.

SARSA Control



At every time-step:

- **Policy evaluation:** Use SARSA to update Q.
- **Policy improvement:** Let new π be ε-greedy w.r.t Q.
- **On-policy:** We always update Q towards q_π for current policy.
- For fixed ε, tends to the best ε-soft policy.

SARSA-algorithm for control

- Initialize $Q(s, a)$ (e.g. $Q(s, a) = 0$ for all s and a).
- For each episode
  1. Get initial state S.
  2. Choose A from S that is ε-greedy w.r.t Q.
  3. For each step of episode:
     - Take action A and observe R, S'.
     - Choose A' from S' that is ε-greedy w.r.t Q.
     - $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$.
     - $S \leftarrow S'$, $A \leftarrow A'$.

Off-policy control – Q-learning

Off-policy learning

- Want to learn $q_\pi(s, a)$ for a **target policy** π with experience from using the **behavior policy** μ.

When is this useful?

- Learn by observing humans or other agents.
- Re-use experience collected from old policies.
- Learn optimal $q_*(s, a)$ while following an *exploratory* policy.

Q-learning

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, \underbrace{A_{t+1}}_{\sim\pi(a|S_{t+1})})|S_t = s, A_t = a\right]$$

- Consider data collect using behavior policy μ

$$S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1} \sim \mu.$$

- Update: Let $A' \sim \pi(a|S_{t+1})$ and use the update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)).$$

Off-policy control with Q-learning

UPPSALA UNIVERSITET

- Let both behavior and target policies improve.
- **Target policy, π:** Greedy w.r.t $Q(s, a)$.
- **Behavior policy, μ:** ε-greedy w.r.t $Q(s, a)$.
- The Q-learning target is then

$$R_{t+1} + \gamma Q(S_{t+1}, A')$$

where $A' = \arg\max_a Q(S_{t+1}, a)$. Inserting this we can rewrite the target as

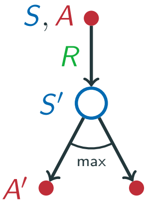
$$R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a)) =$$

$$R_{t+1} + \gamma \max_a Q(S_{t+1}, a).$$

- Compare to Bellman optimality equation:

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma \max_a q_*(S_{t+1}, a) | S_t = s, A_t = a]$$

UPPSALA UNIVERSITET



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

Theorem

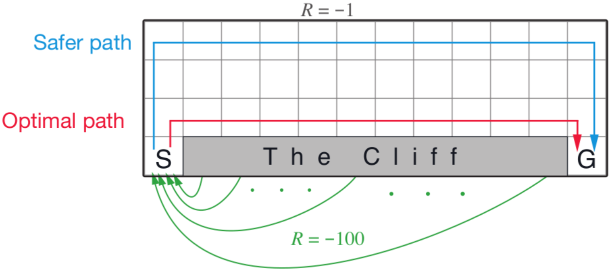
Q-learning converges to the optimal action-value function $q_*(s, a)$ as $N(s, a) \to \infty$ if the step size α decreases towards 0 with a suitable rate.

- With good estimate of $q_*(s, a)$, can find the optimal policy $\pi_* = \text{greedy}(q_*)$.
- In practice constant α often works well if it is small enough.

Q-learning for off-policy control

- Initialize $Q(s, a)$ (e.g. $Q(s, a) = 0$ for all s and a).
- For each episode
  1. Get initial state S.
  2. For each step of episode:
     - Choose A from S that is ε-greedy w.r.t to Q.
     - Take action A and observe R, S'.
     - $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$.
     - $S \leftarrow S'$
- When training is done: Get target policy as greedy w.r.t to Q.

Example 6.6: SARSA vs Q-learning



- **Optimal path:** Greedy with respect to q_* (found with Q-learning).
- **Safe path:** Best ε-soft policy (found with SARSA and fixed $\varepsilon = 0.1$).

Summary

Bellman equations vs learning algorithms

Bellman	**Sample backup**
For v_π $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$	TD-target $R_{t+1} + \gamma V(S_{t+1})$
For q_π $q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$	SARSA-target $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$
For optimal q_* $q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_a q_*(S_{t+1}, a) \mid S_t = s, A_t = a]$	Q-learning target $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$

Summary

UPPSALA UNIVERSITET

- **Idea:** Estimate $q_\pi(s, a)$.
- **Exploration:** Needed in order to learn about all possible actions.
- **ε-greedy policy:** Greedy with prob $1 - \varepsilon$, and choose random action with prob ε.
- **On-policy:** MC and SARSA. With fixed ε tends to best ε-soft policy.
- **Off-policy:** Q-learning. Converge to q_*, which can be used to find optimal policy.

Next:

- Tinkering Notebook 3 and Basic Assignment 2 (soft deadline April 24).
- Function approximation.
- For 7,5 credits: Extra lecture tomorrow.