LECTURER: TAI LE QUY

INTRODUCTION TO REINFORCEMENT LEARNING

Introduction to Reinforcement Learning	1
Sequential Decision Process	2
Dynamic Programming	3
Reinforcement Learning Algorithms and their Properties	4
Deep Reinforcement Learning	5
Summary: Introduction to Reinforcement Learning	6

UNIT 4

REINFORCEMENT LEARNING ALGORITHMS AND THEIR PROPERTIES

STUDY GOALS

- Describe the mechanisms of temporal difference algorithms and their importance in reinforcement leaning (RL)
- Compare and contrast exploration and exploitation strategies used by RL agents to balance their behaviors
- Understand how SARSA and Q-learning algorithms behave in different RL scenarios

EXPLAIN SIMPLY

1. Explain how RL algorithms achieve optimal policies through mapping between states and actions

2. Describe the differences between model-based and model-free RL algorithms

3. Compare and contrast SARSA and Q-learning algorithms discussing their strengths and weaknesses

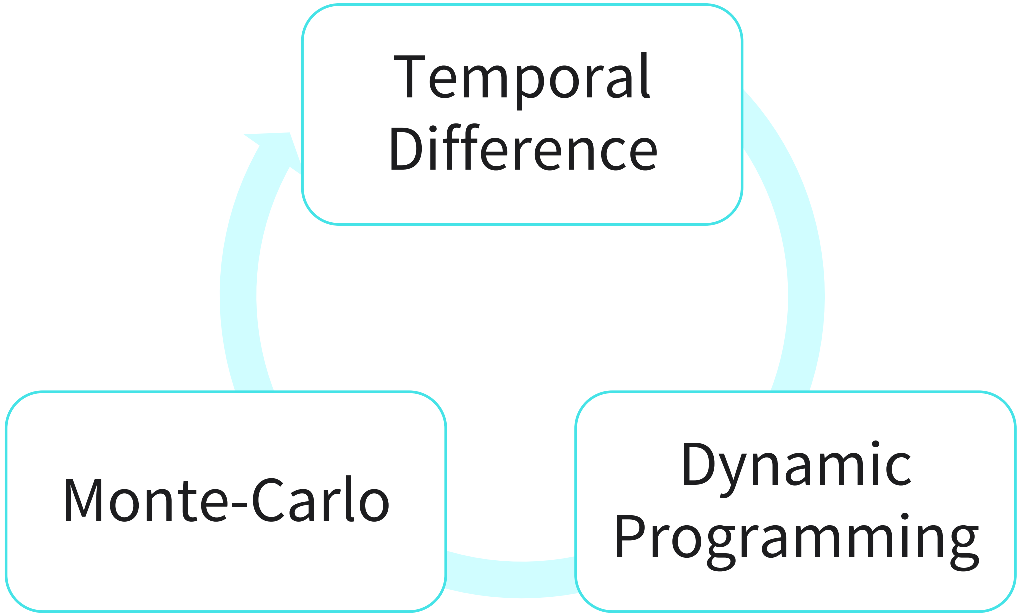
CATEGORIZATIONS OF REINFORCEMENT LEARNING

MODEL-BASED AND MODEL FREE

- Categorized based on whether the algorithm has **access** to the internal dynamics of the environment that represents the problem being solved or not.
- Model-based learn optimal policies using **transition functions** (could be true models of the environment, if available, or they could be learned approximations)
  - Dynamic programming (DP)
- Model-free algorithms learn optimal policies directly from the raw experiences they gather through their interactions with the environment
  - Monte-Carlo (MC) method
  - Temporal difference (TD) learning (build on DP and MC)
    - Q-learning
    - State-action-reward-state-action (SARSA)

Predict a quantity based on future values of a signal

- Dynamic programming assumes perfect world model
- Monte-Carlo methods are model-free and learn from raw experiences
- Temporal difference methods combine the best of both worlds



STATE VALUE FUNCTIONS (V), ACTION VALUE FUNCTIONS (Q).

- State value function $V_\pi(s_t) = E[G_t | s_0 = s_t]$
- Action value function $Q_\pi(s_t, a_t) = E[G_t | s_0 = s_t, a_0 = a_t]$
- Discounted rewards $G_t = \sum_{t=0}^{T} \gamma^t . r_t$
- Bellman equations

$$V_\pi(s_t) = E_{s_{t+1} \sim P}[r_t + \gamma . V_\pi(s_{t+1})]$$

$$Q_\pi(s_t, a_t) = E_{s_{t+1} \sim P}\left[r_t + \gamma . E_{a_{t+1} \sim \pi}[Q_\pi(s_{t+1}, a_{t+1})]\right]$$

- The next state s_{t+1} is sampled from the transition function P and the next action a_{t+1} is sampled from the policy π

- Value functions in DP are updated as per the Bellman equation

$$V_{k+1}(s_t) = E[r_t + \gamma \, . \, V_k(s_{t+1})]$$

  - The estimate of the value function is updated to V_{k+1} based on the current estimate V_k. (**bootstrapping –** “taking a guess from a guess”)
  - Assume access to a perfect model of the world

- MC methods learn **directly** from the raw experiences that are gathered through the agent-environment interactions (model-free), **sampling**.

$$V_{k+1}(s_t) = V_k(s_t) + \alpha \,.\, [G_t - V_k(s_t)]$$

  - G_t : the true return from time step *t*, $[G_t - V_k(s_t)]$: error term identifying how far off the V -function's estimate is from the true return
  - The hyperparameter α: step size or the learning rate.
    - Controls the rate at which new information, If α is large, then the error term has a large influence on the factor by which the estimate is updated
    - Algorithms must wait until the end of the **episode** when G_t is available to determine the update that would be applied to V_k.
  - MC methods only work with episodic tasks, i.e., tasks, such as games like Chess and Go that have a clear end as marked by a terminal state.

- TD is model-free and learns by sampling, **do not wait** till the end of the episode to perform an update step.

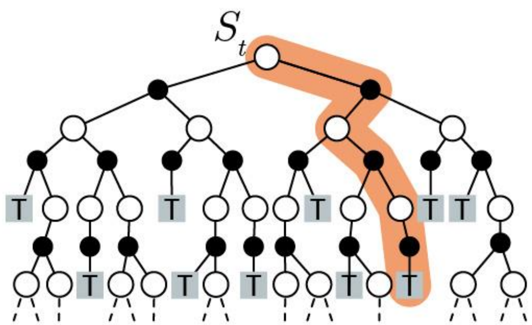
$$V_{k+1}(s_t) = V_k(s_t) + \alpha \, . \left[r_t + \gamma \, . \, V_k(s_{t+1}) - V_k(s_t) \right]$$

  - The true return G_t has been replaced by an **estimated** value
  - $V_k(s_{t+1})$ is called TD target
  - The difference between the target and the current value is an error term called TD error, denoted as δ_t

TEMPORAL DIFFERENCE LEARNING AND Q-FACTORS

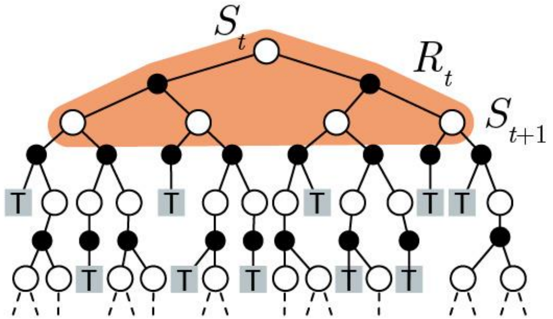
Monte Carlo

$$V(S_t) \mapsto V(S_t) + \alpha \cdot (G_t - V(S_t))$$



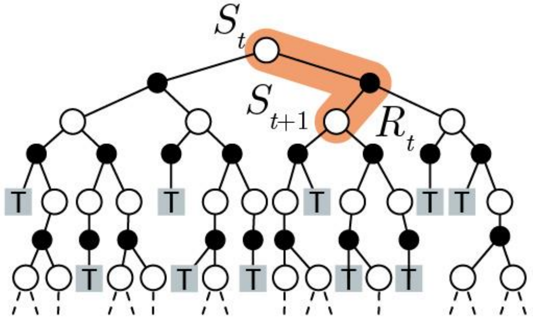
Dynamic Programming

$$V(S_t) \mapsto E_\pi \left[R_t + \gamma \cdot V(S_{t+1})\right]$$



Temporal Difference

$$V(S_t) \mapsto V(S_t) + \alpha(R_t + \gamma \cdot V(S_{t+1}) - V(S_t))$$



Source of the text: Silver, 2020.
Source of the image: Silver, 2020.

TD algorithm: efficient learning from raw experiences

Observe current state and choose action → Observe reward and new state → Calculate TD error → Update value function

$TD(0)$

Multi-step TD

$TD(\lambda)$

- TD(0): just one step into the future, performing updates after every single transition
- 2-step looks ahead

$$V_{k+1}(s_t) = V_k(s_t) + \alpha \, . \left[r_t + \gamma \, . \; r_{t+1} + \gamma^2 \, . V_k(s_{t+2}) - V_k(s_t) \right]$$

- Updated equation

$$V_{k+1}(s_t) = V_k(s_t) + \alpha \, . \left[G_t - V_k(s_t) \right]$$

- N-step TD

$$G_{t:t+n} = r_t + \gamma \, . r_{t+1} + \gamma^2 \, . r_{t+2} + \ldots + \gamma^n \, . V_k(s_{t+n})$$

- n = ∞, the TD update rule becomes equivalent to the MC update rule
- if t + n = T, where T is the length of the entire trajectory, we will get the true return G_t. → choosing n is difficult

- The λ-return is computed as geometrical weighted sum of the **n-step** returns

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} \,.\, G_{t:t+n}$$

  - The number of steps being considered ranging from 1, all the way to ∞.
  - λ ∈ [0, 1] . When λ = 0, we get the 1-step formulation of TD and increasing the value of λ increases the number of rewards to be sampled to estimate the return

BENEFITS AND SHORT-COMINGS OF TD LEARNING

BENEFITS

- Online learning
- Learn from incomplete sequences
- Episodic and non-episodic tasks
- Efficient
- Faster convergence

SHORT COMINGS

- Sensitive to start state
- Biased compared to MC
- Overestimates with function approximation
- Requires knowledge of rewards
- Limited to Markov process

Exploration vs Exploitation Dilemma

"To obtain reward, an agent must prefer effective actions that it has tried in the past. But to discover such actions, it has to try actions that it has not selected before"

Exploitation

- Pick action based on policy
- Gain large reward

Exploration

- Pick novel action
- Gain environment knowledge

Source of the text: Sutton and Barto, 2018.

Balancing exploration vs exploitation in RL

- Epsilon (ϵ) controls exploration and exploitation balance
- Ranges from completely exploitative to completely exploratory

$$a_t = \begin{cases} \pi(s_t) & with\ probability\ (1-\epsilon) \\ random\ action\ with\ probability\ \epsilon \end{cases}$$

- Epsilon value decays over time

Iteratively estimate Q-value using state-action pairs

- TD method to learn Q-values for state-action pairs
- Policy update by action selection
- The update rule for SARSA

$$Q_{k+1}(s_t,\ a_t) = Q_k(s_t,\ a_t) + \alpha\,.\left[r_t + \gamma\,.\,Q_k(s_{t+1}, a_{t+1}) - Q_k(s_t, a_t)\right]$$

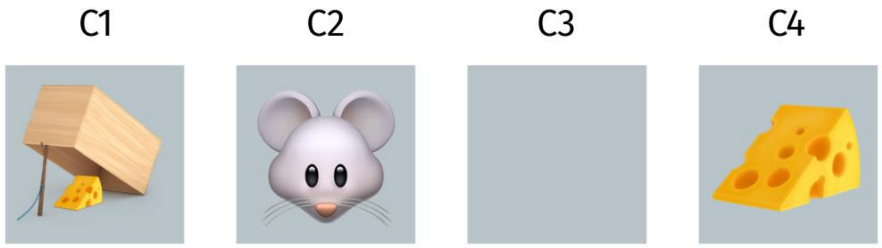
- SARSA updates its Q-value estimate based on the Q-value of the next state, s_{t+1}, and the action, a_{t+1}, suggested by the current policy
- SARSA updates its Q-value based on the policy it currently follows → on-policy

Figure 33: Pseudocode for the SARSA Algorithm

```
Initialize Q(s,a) arbitrarily
for episode = 1,2,..., M do
    Initialize state s_1
    Select action a_1 per policy derived from Q (e.g., ϵ-greedy)
    for t = 1,2,...,T do
        Perfom a_t and observe r_t, s_{t+1}
        Select action a_{t+1} per policy derived from Q(e.g., ϵ-greedy)
        Update Q(s_t, a_t) as Q(s_t, a_t) + α[r_t + γ.Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
        s_t = s_{t+1}, a_t = a_{t+1}
    end for
end for
```

Source of the image: Nair, 2023.

ON POLICY LEARNING: SARSA, EXAMPLE



The Mouse Gridworld Example

Cell	Left	Right
C1	0	0
C2	-0.9	0.7
C3	0.4	0.9
C4	0	0

Q-Table Representing Q-Values for the Mouse Gridworld Example

Estimate optimal Q-value using highest expected return

- Q-learning update: reward + maximum future Q-value

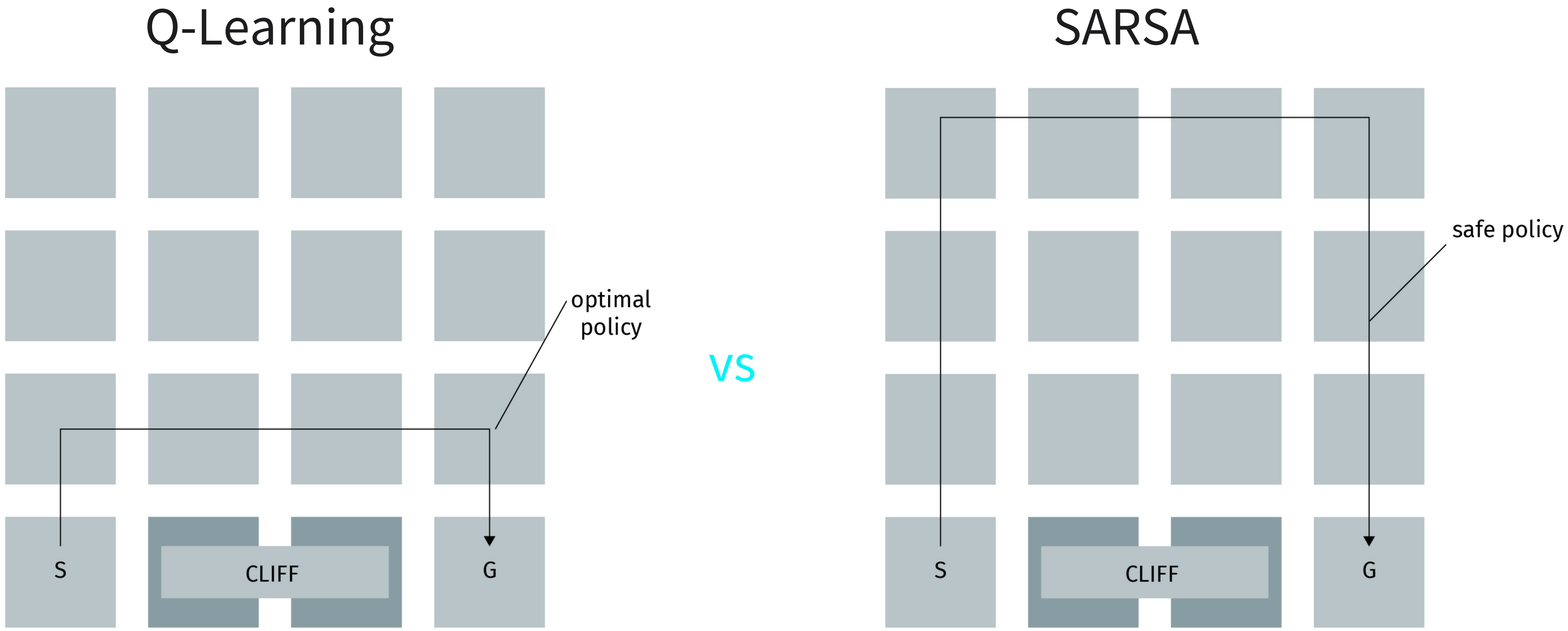
$$Q_{k+1}(s_t, a_t)$$
$$= Q_k(s_t, a_t) + \alpha \,.\left[r_t + \gamma \,.\, \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}) - Q_k(s_t, a_t)\right]$$

- Future Q-value computed assuming optimal policy

Figure 35: Q-Learning

```
Initialize Q(s,a) arbitrarily
for episode = 1,2,..., M do
    Initialize state s_1
    for t = 1,2,...,T do
        Select a_t per policy derived from Q (e.g., ϵ-greedy)
        Perfom a_t and observe r_t, s_{t+1}
        Update Q(s_t, a_t) as Q(s_t, a_t) + α[r_t + γ.max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
        s_t = s_{t+1}
    end for
end for
```

EXAMPLE: Q-LEARNING VERSUS SARSA IN THE CLIFF WALKING GRIDWORLD



Source of the image: Nair, 2023.

REVIEW STUDY GOALS

- Describe the mechanisms of temporal difference algorithms and their importance in reinforcement leaning (RL)

- Compare and contrast exploration and exploitation strategies used by RL agents to balance their behaviors

- Understand how SARSA and Q-learning algorithms behave in different RL scenarios

SESSION 4

TRANSFER TASK

Case study

You are tasked with developing an AI-agent to play Tic-Tac-Toe. Your goal is to develop an optimal policy that maximizes the chances of an agent to win against a human opponent.

Task

What type of reinforcement learning algorithm would be the most appropriate for this problem? What are the key steps of implementing this algorithm for game playing? Consider factors such as exploration vs exploitation, reward design, state space and action space representation

Please present your results.
The results will be discussed in plenary.

1. Temporal difference learning uses the benefits of which methods?
   a) DP and unsupervised learning
   b) MC and genetic algorithms
   c) MC and ML-models
   d) DP and MC

2. Which a popular approach for the agent to explore the environment
   a) Gradient descent
   b) Dynamic programming
   c) Epsilon greedy
   d) TD-learning

LEARNING CONTROL QUESTIONS

3. Using the Q maximizing action in Q-learning makes it a __________ method

   a) Off policy
   b) On policy
   c) Model-based
   d) Model-free

LIST OF SOURCES

Text

Sutton, R. S., & Barto A. G. (2018). *Reinforcement learning: an Introduction.* The MIT Press. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf
Silver, D. (2020). Model-Free Prediction [lecture notes]. https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf

Images

Silver, 2020.
Nair, 2023.

© 2022 IU Internationale Hochschule GmbH
This content is protected by copyright. All rights reserved.
This content may not be reproduced and/or electronically edited, duplicated, or distributed in any kind of form without written permission by the IU Internationale Hochschule GmbH.