CS6482 Deep RL

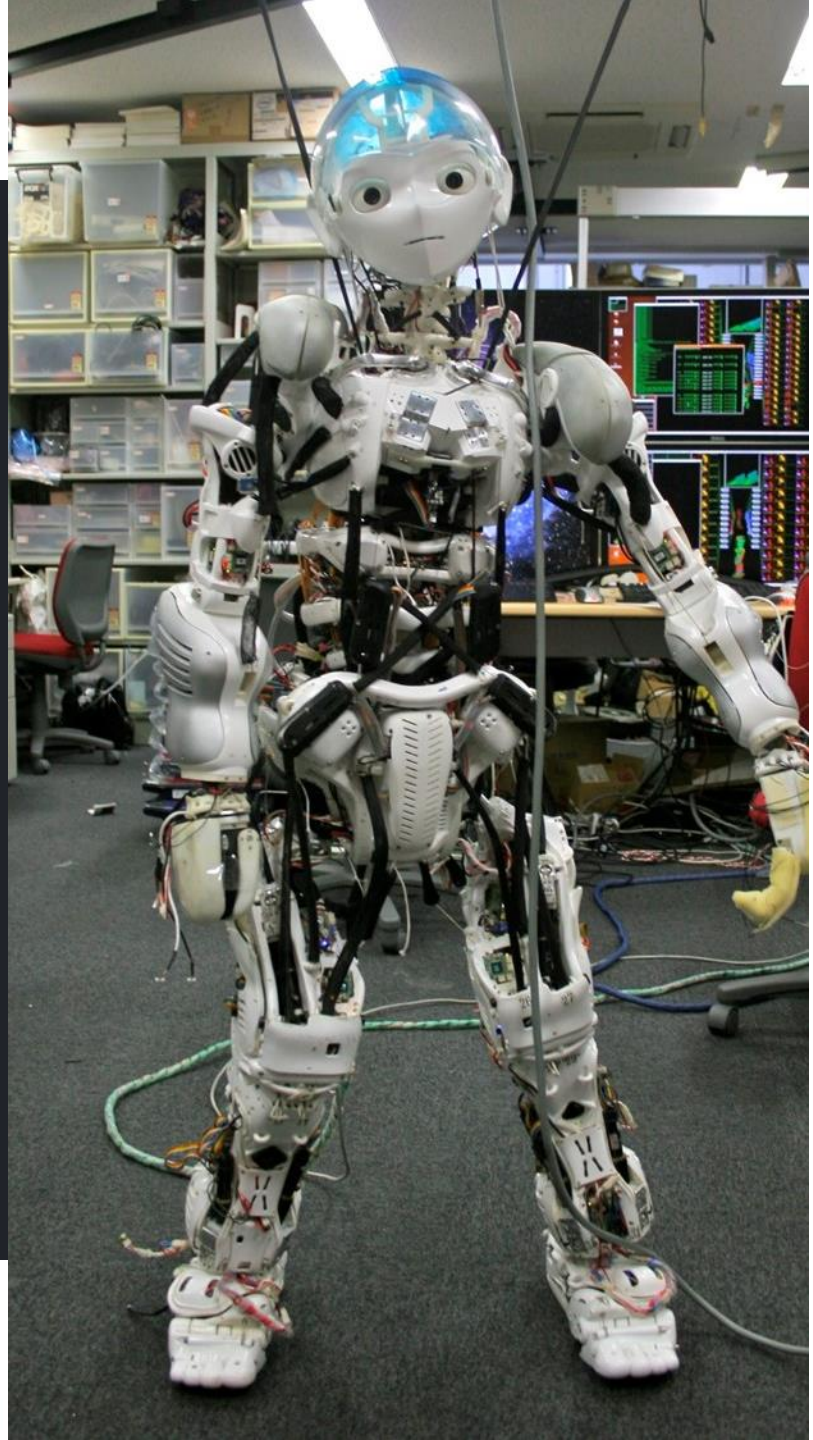
F: Reinforcement Learning: TD LEarning

J.J. Collins

Dept. of CSIS

University of Limerick

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH



Objectives

- Understand the Temporal Difference (TD) update
- Compare TD with Monte Carlo
- Differentiate between Sarsa and Q Learning
- Analyse the results from the cliff walking example

Summary of Chapters 6 in Sutton and Barto. Reinforcement Learning: an Introduction, 2nd Edition. The MIT Press. 2018.

Recap

RL → trial and error learning
[Optimal] mapping from states to actions to maximise long term reward
Take action a in state s, transition to state s', and receive reward r

Value Functions

State value function V (prediction)
State-Action value function Q (control)

How to learn $v_*, q_*, and\ \pi_*$

Concerns

Exploration versus exploitation
Credit Assignment Problem

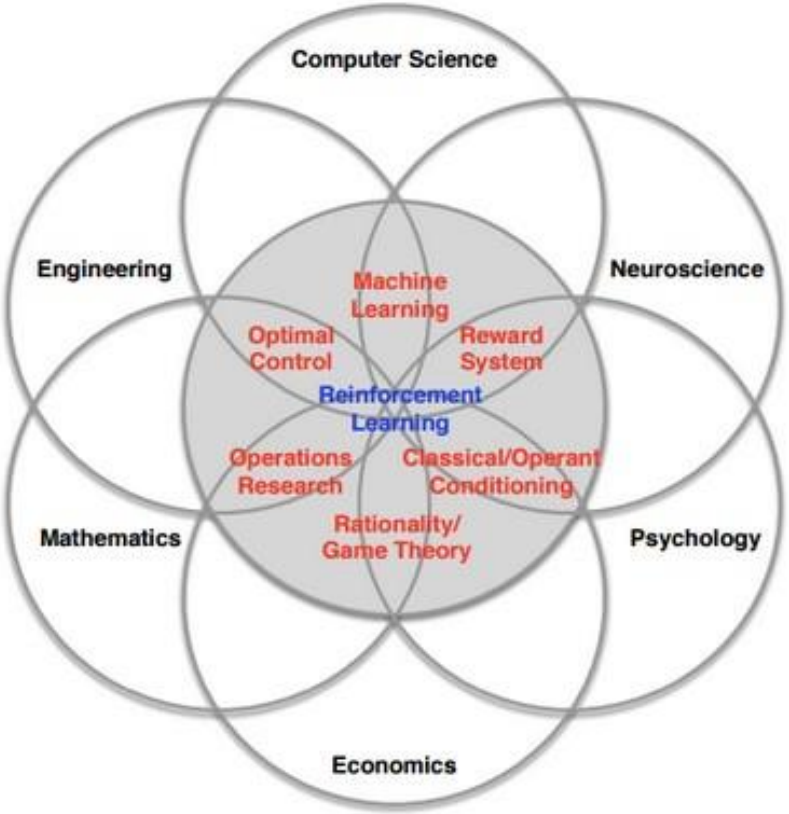
Recap cont.

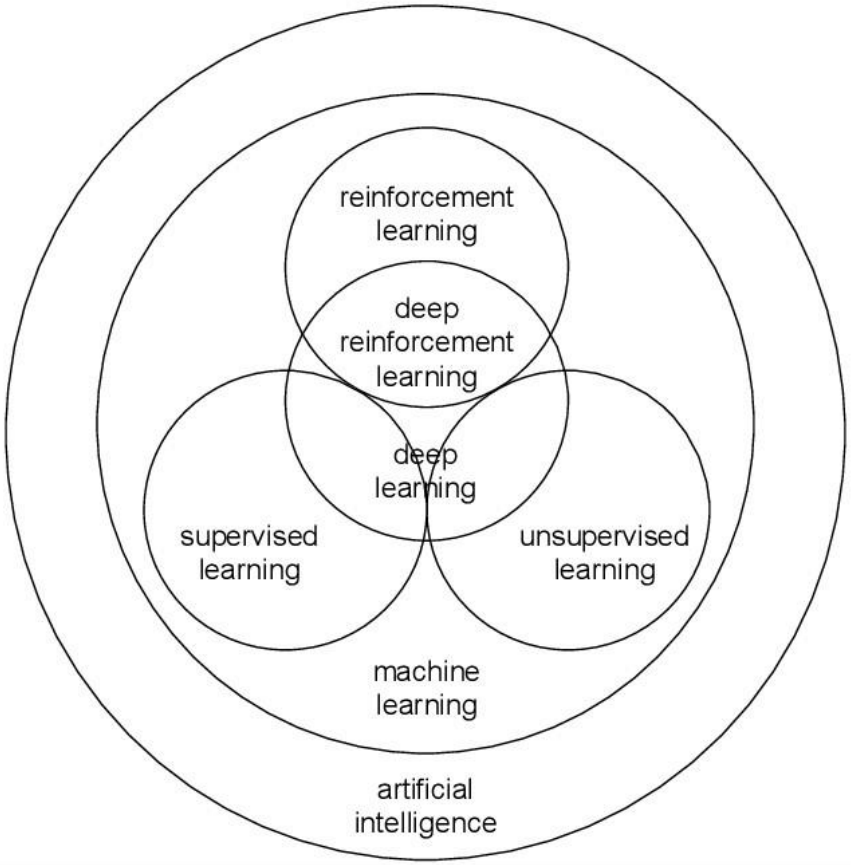
- **Markov Decision Process (MDP):**
  - Current state is fully observable and contains all the information necessary to solve the problem
  - RL is an MDP sequential decision making process
- Underpinned by Bellman's Optimality Equation

- Algorithms
  - **Dynamic Programming**
    - Requires a model
    - Solved using General Policy Iteration (GPI):
      - Policy evaluation and Policy Improvement
    - Bootstraps
      - Update is based on value of next state → guess based on a guess
  - **Monte Carlo**
    - Model free
    - Does not bootstrap,
      - **Must wait for the episode to end**
  - **Temporal Difference**

Recap

- Context





David Silver (2016)., Google DeepMind
https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ&index=1

Li et al (2018). https://www.semanticscholar.org/paper/Deep-Reinforcement-Learning-Li/f2ac2a3fd7b341f2b1be752b4dd46ed9abcf0751

Introducing Temporal Difference (TD)

- Cornerstone of RL
- Bridges Dynamic Programming (DP) and Monte Carlo (MC)
  - Like DP, TD **bootstraps**
  - Like MC, **TD does not require a model**, it learns from experience
- MC Update
  - Must wait until episode has terminated
  - $V(s_t) \leftarrow V(s_t) + \alpha[\boldsymbol{G_t} - V(s_t)]$
  - Where the target is $\mathrm{G_t}$

- TD Update
  - $V(s_t) \leftarrow V(s_t) + \alpha[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma V(s_{t+1})} - V(s_t)]$
  - Where the target is $R_{t+1} + \gamma V(s_{t+1})$
  - The error is $\delta = [R_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

Introducing TD

- Predicting journey time
- Rewards are time taken to complete a leg of the journey
- Value(S) = predicted time to reach destination from that state
- TD Continuously updates prediction
- Error is proportional to the change over time of the prediction i.e. temporal difference in predictions

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

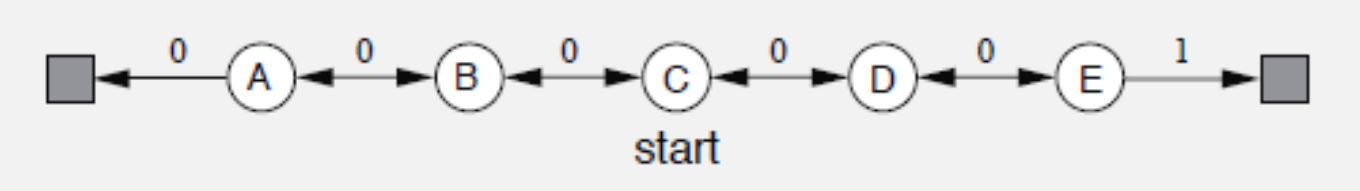
Introducing TD

Exercise 6.2.

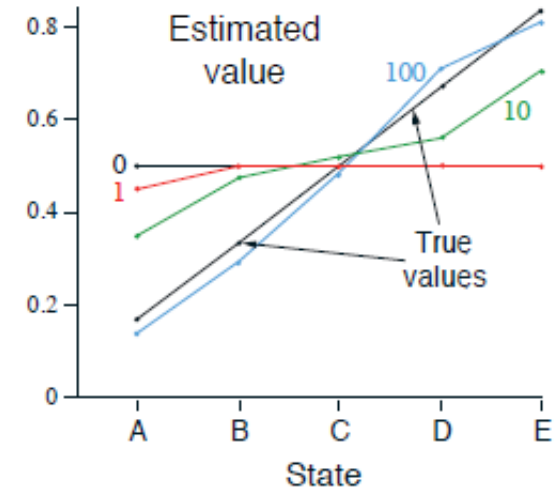
- Describe a scenario in which a TD update would be better than a Monte Carlo update?

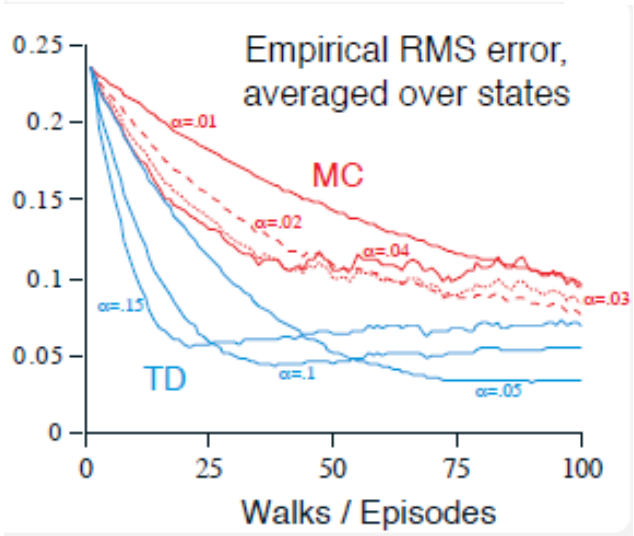
- Answer: You change employer and move to a new building at the same side of town. You still join the motorway using the same junction when driving home. Now you are starting to learn predictions for the new building. Can you see why TD updates are likely to be much better, at least initially, in this case?

MC v TD(o)



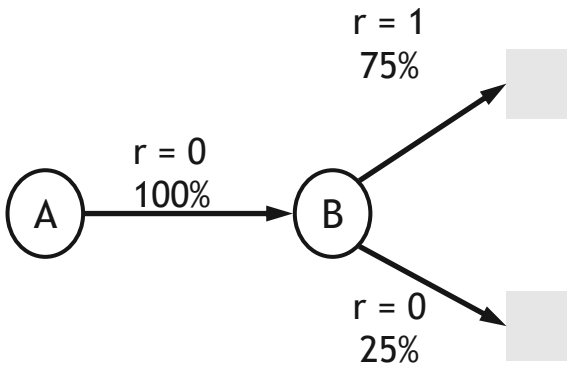
- A Random Walk
- Start in State C
- Move left or right equiprobable
- Reward of +1 if episode terminates on right
- Reward of 0 for left
- V(s) = $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$, for states A to E
- Top graph: performance of TD(0) for n episodes





UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

MC vs. TD(0)



- A different problem
- Observed the following 8 episodes:

(e1) A – 0, B – 0	(e2) B – 1	(e3) B – 1	(e4) B - 1
(e5) B – 1	(e6) B – 1	(e7) B – 1	(e8) B – 0

- MC and TD agree on V(B) = 3/4
- MC: V(A) = 0
  - converges to values that minimize the error on training data
- TD: V(A) = 3/4
  - converges to Maximum Likelihood estimate of the Markov process

TD Learning: Sarsa

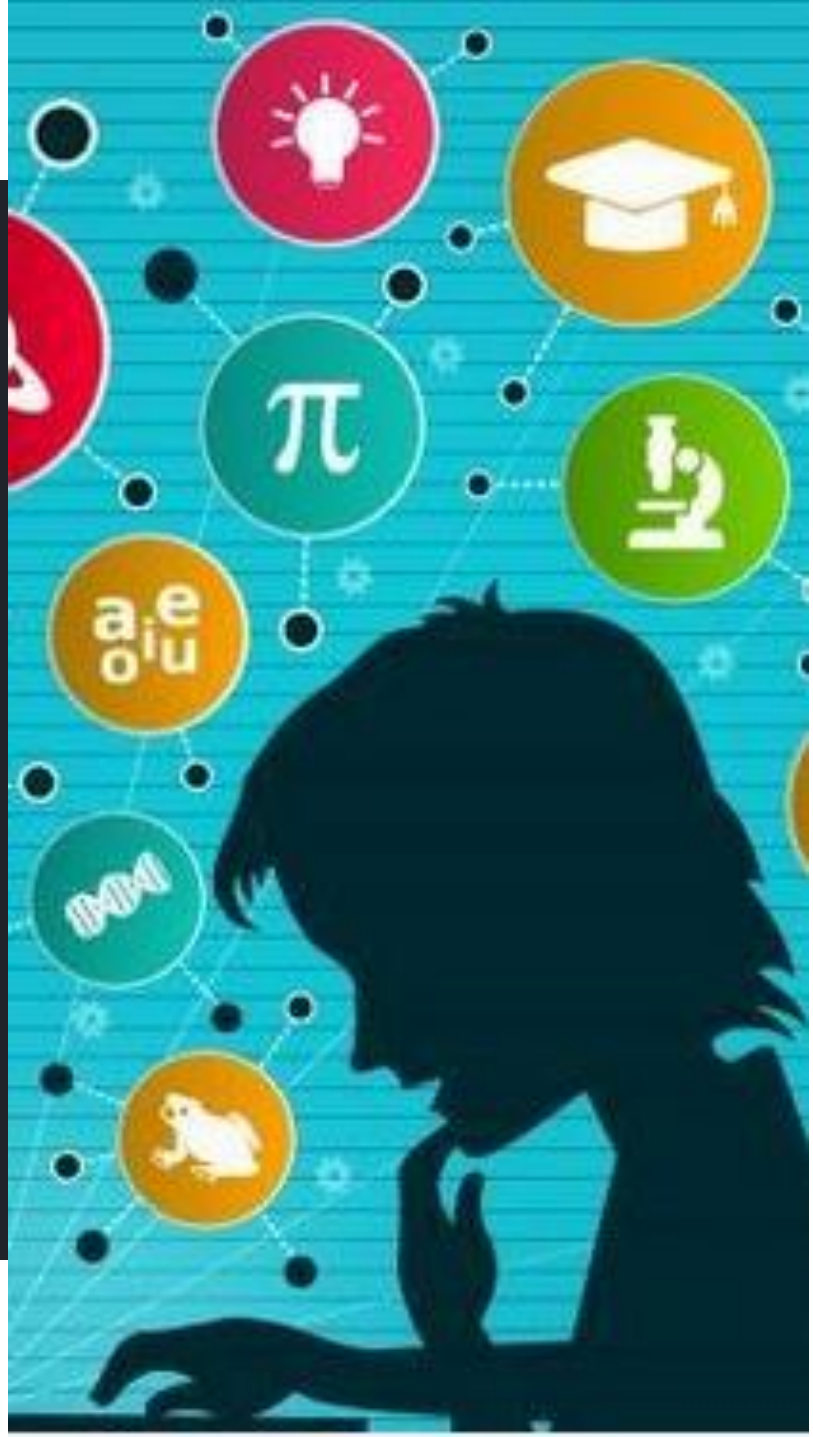
- State action reward state action

Sarsa on-policy control for Q $\approx q_*$

1. Parameters: $\alpha, \varepsilon, \gamma$

2. Set Q(T,*) = 0 where T = terminal state
3. Initialise all other Q(s,a) arbitrarily

4. Loop
5. select action a from starting state s using ε-greedy policy derived from Q
6. For each step of the episode
7. take action a, observe reward r, and transition to successor state s'
8. select action a' from state s' using ε-greedy policy derived from Q
9. $Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
10. Let s = s', a = a'



On Policy versus Off Policy

- On-policy methods improve the policy that is used to make decisions, whereas off-policy methods improve a policy different from that used to generate the data.
- The Monte Carlo ES method seen in previous lecture is an example of an on-policy method.
- Q: what about the TD(0) update seen at the start of this lesson?

On Policy versus Off Policy

- All learning control methods seek to learn action values conditional on subsequent optimal behaviour,
- But they need to behave non-optimally in order to **explore**.
- **How can an agent learn about the optimal policy while behaving according to an exploratory policy?**
- The on-policy approach is a compromise—it learns action values not for the optimal policy, but for a near-optimal policy that still explores.
- An alternative approach is to use an **off-policy** approach,
  - The policy being learned (optimised) is called the **target policy**, and
  - The policy used to generate behaviour is called the **behaviour policy**.
  - In this case we say that learning is from data “off” the target policy.

TD: Q Learning

- Watkins (1989)

Q Learning off-policy control for $\pi \approx \pi_*$

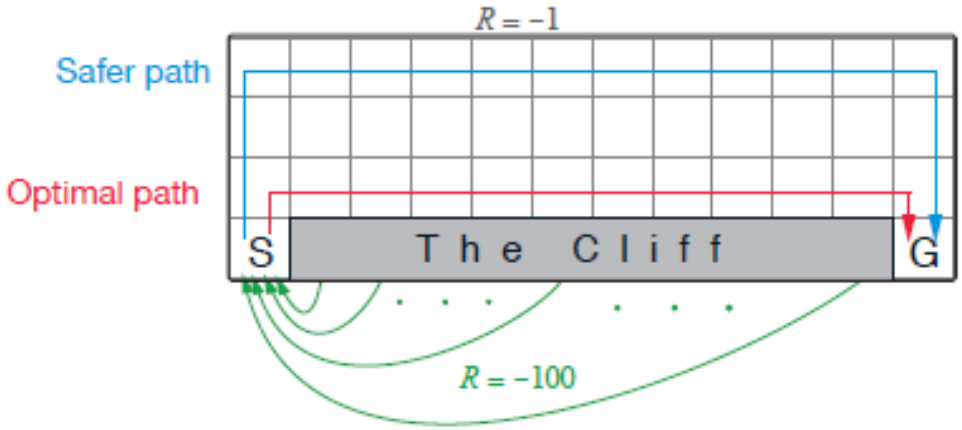
1. Parameters: $\alpha, \varepsilon, \gamma$

2. Set Q(T,*) = 0 where T = terminal state
3. Initialise all other Q(s,a) arbitrarily

4. Loop
5. choose starting state s
6. For each step of the episode
7. select action a from state s using ε-greedy policy derived from Q
8. take action a, observe reward r, and transition to state s'
9. $Q(s,a) = Q(s,a) + \alpha\left[r + \gamma max_{a'}\left(Q(s',a')\right) - Q(s,a)\right]$
10. Let s = s'

Cliff Walking Example

- Traverse grid from start to goal
- Reward of -1 on all transitions except for
- Reward of -100 if one falls off the cliff
- Which path is Sarsa and Q likely to compute?
- Why does Q have a greater negative sum of rewards?



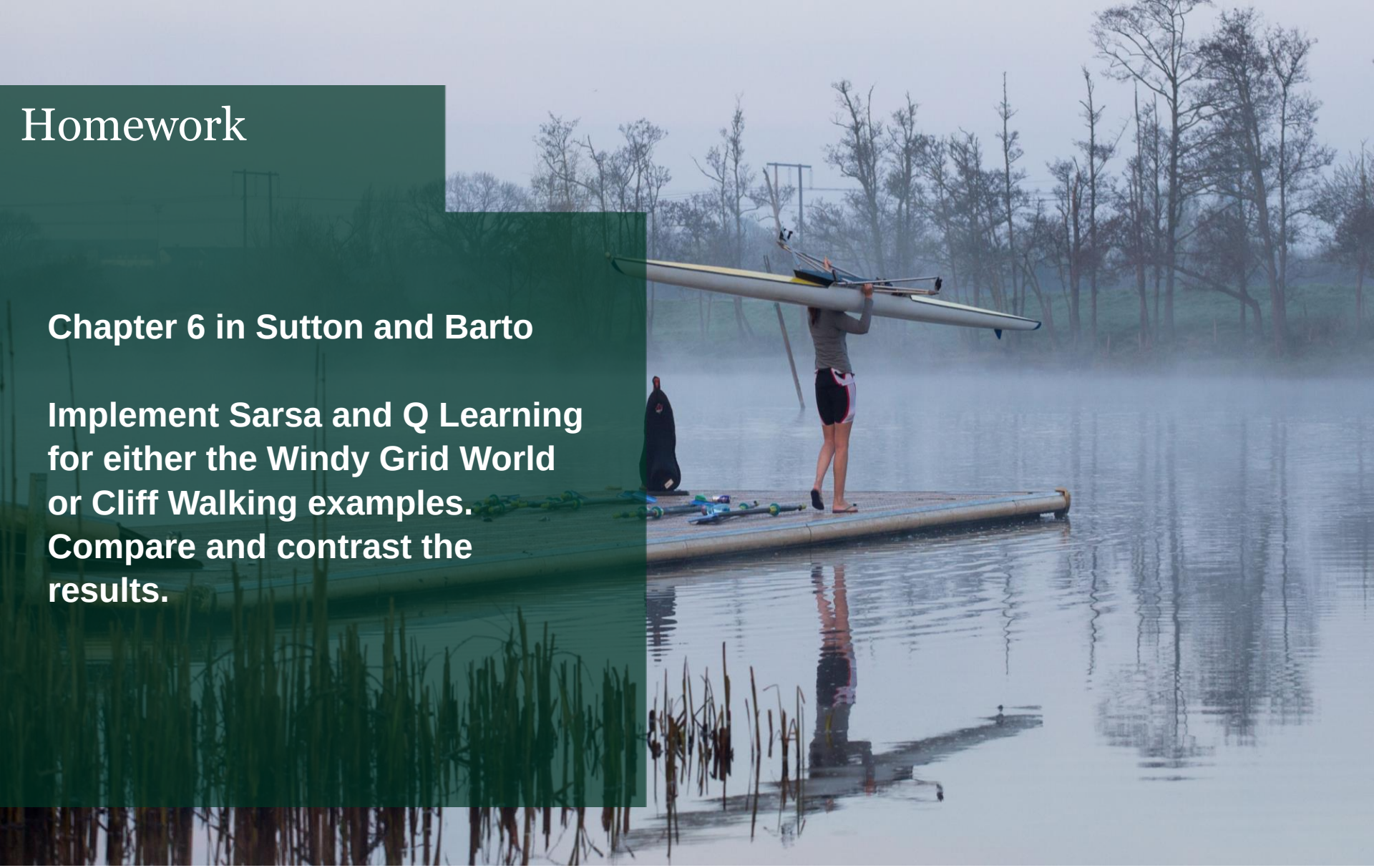
Summary

- Introduced TD(0) in the form of Sarsa and Q learning
  - Where TD(0) = TD($\lambda = 0$) = Vanilla TD
- Compared MC and TD(0)
- On-Policy versus Off-Policy

- Note: Policy Convergence Theorem for DP (Bellman 1957) applies to MC and TD(0) in tabular format
  - Did not mention this in previous lessons!

- Next: a look at Deep Q Networks aka Deep RL.

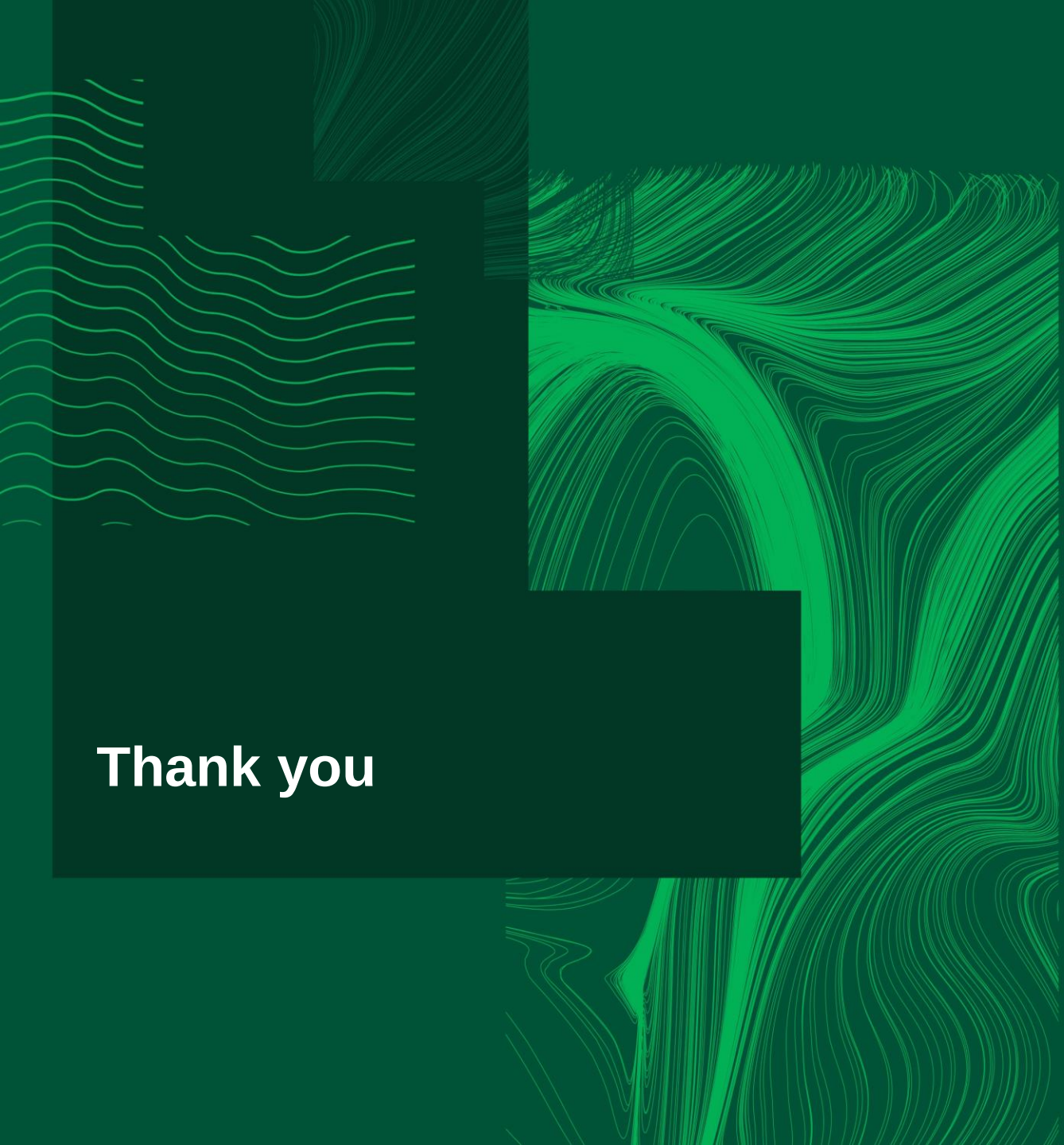
Homework



Chapter 6 in Sutton and Barto

Implement Sarsa and Q Learning for either the Windy Grid World or Cliff Walking examples. Compare and contrast the results.

Thank you



UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

University of Limerick,
Limerick, V94 T9PX,
Ireland.

Ollscoil Luimnigh,
Luimneach,
V94 T9PX, Éire.

+353 (0) 61 202020

ul.ie