Deep Learning
Lab16:
Reinforcement Learning

Datalab

Outline

- MDP(value iteration & policy iteration)
- Q-Learning & SARSA
- Homework

Outline

- MDP(value iteration & policy iteration)
- Q-Learning & SARSA
- Homework

MDP

- Value iteration & Policy iteration
  - Example: Grid world
  - Please trace the code in the notebook to help you understand these two algorithms

Outline

- MDP(value iteration & policy iteration)
- Q-Learning & SARSA
- Homework

Q-Learning

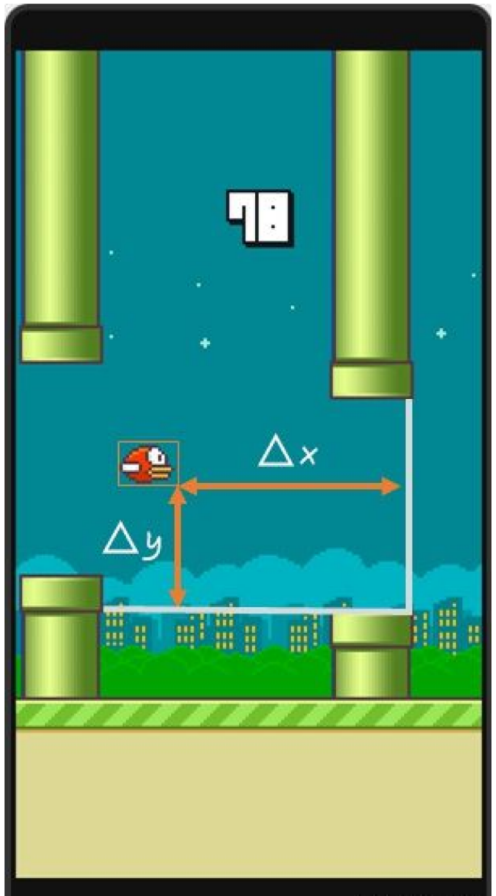
- Algorithm

Q-learning: An off-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-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

- An example
  - Flappy bird



Q-Learning

- An example
  - States: $(\Delta x, \Delta y)$
  - Actions: fly, none
  - Reward:
    - +1: pass through a pipe
    - -5: die

Q-Learning

- An example
  - Q-table(finite):

状态	飞	不飞
$(\Delta x_1, \Delta y_1)$	1	20
$(\Delta x_1, \Delta y_2)$	20	-100
…	…	…
$(\Delta x_m, \Delta y_{n-1})$	-100	2
$(\Delta x_m, \Delta y_n)$	50	-200

  - Update rule: $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

Q-Learning

- Reference
  - https://www.zhihu.com/question/26408259

SARSA

- Algorithm

Sarsa: An on-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-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

- An example
  - Q-table(finite):

状态	飞	不飞
$(\Delta x_1, \Delta y_1)$	1	20
$(\Delta x_1, \Delta y_2)$	20	-100
…	…	…
$(\Delta x_m, \Delta y_{n-1})$	-100	2
$(\Delta x_m, \Delta y_n)$	50	-200

  - Update rule: $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$

Q-Learning VS. SARSA

- Difference

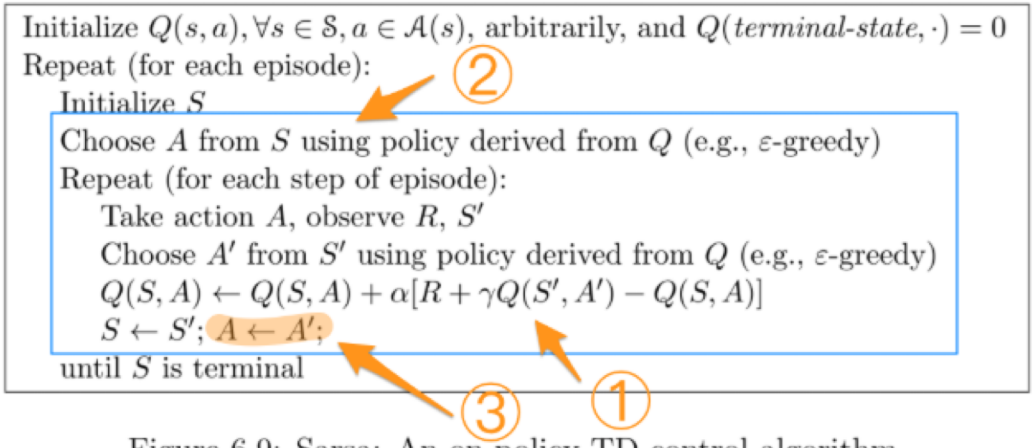


Figure 6.9: Sarsa: An on-policy TD control algorithm.

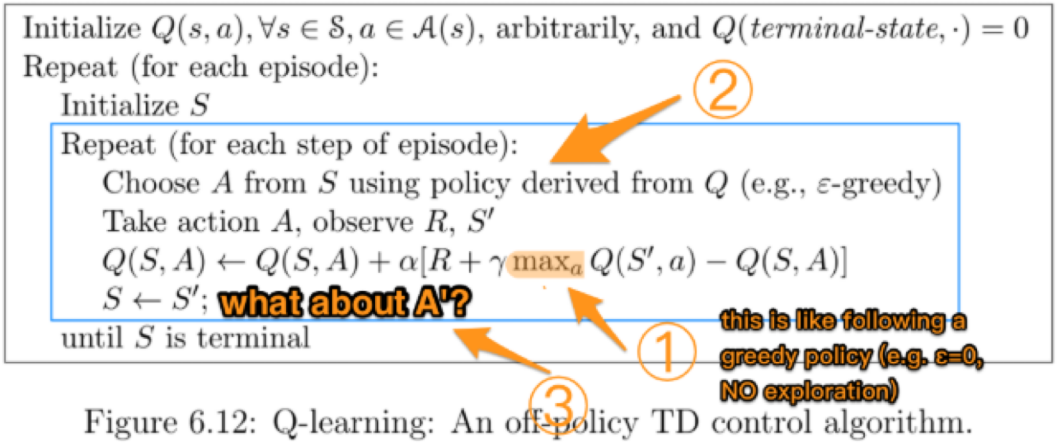


Figure 6.12: Q-learning: An off-policy TD control algorithm.

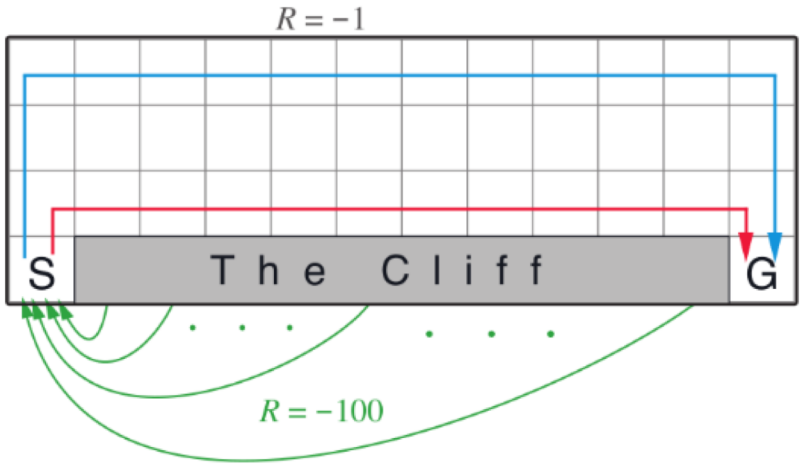
Q-Learning VS. SARSA

- Difference
  - Q-learning is using different policies for choosing next action A' and updating Q. In other words, it is trying to evaluate π while following another policy μ, so it's an off-policy algorithm.
  - In contrast, SARSA uses π all the time, hence it is an on-policy algorithm.

	SARSA	Q-learning
Choosing A'	π	π
Updating Q	π	μ

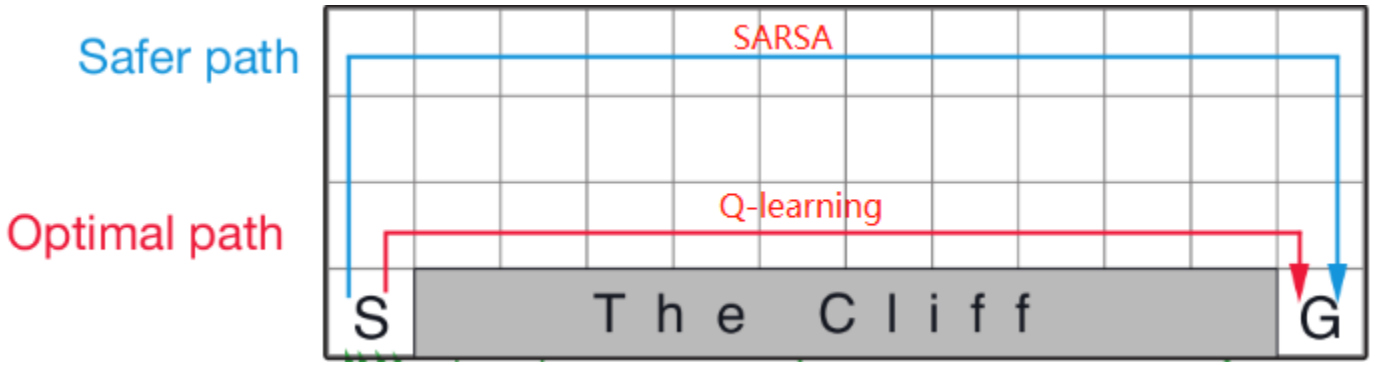
Q-Learning VS. SARSA

- An example(Cliff Walking)
  - States: 4 * 12 grids(S: start state, G: goal state, terminal state)
  - Actions: up, down, left, right
  - Reward:
    - Cliff: -100
    - Other states: -1



Q-Learning VS. SARSA

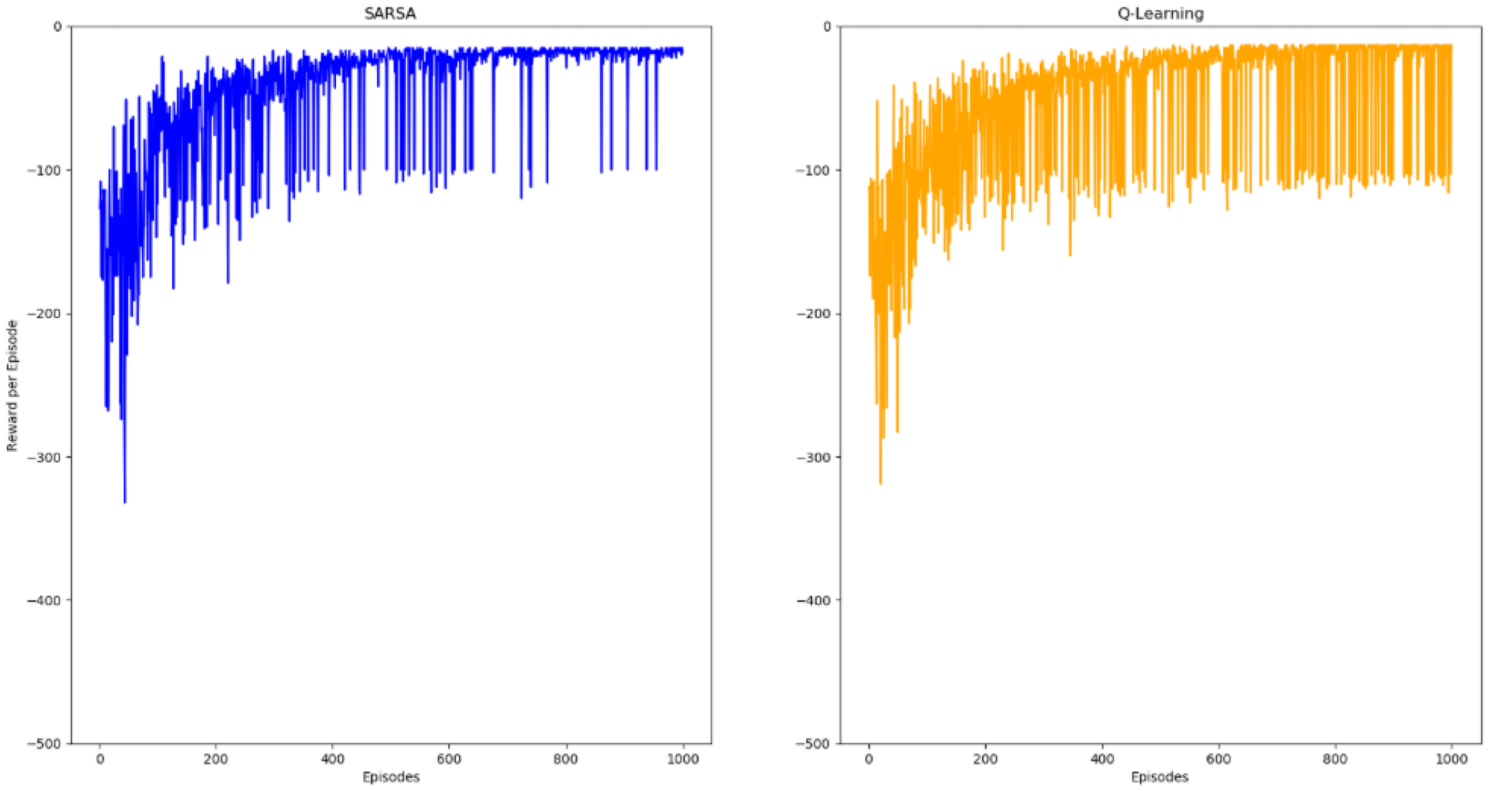
- An example(Cliff Walking)
  - Result



  - Cliff Walking

Q-Learning VS. SARSA

- An example(Cliff Walking)
  - Reward per episode

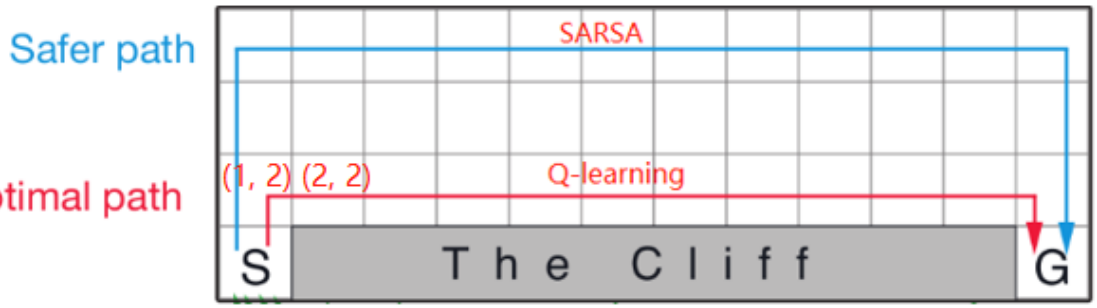


Q-Learning VS. SARSA

- An example(Cliff Walking)
  - Conclusion:
    - SARSA is a **conservative** algorithm
    - Q-learning will get a better reward
  - Why?
    - Q-learning will overestimate the q-value of the action

Q-Learning VS. SARSA

- An example(Cliff Walking)
  - Why?
    - Q-learning will overestimate the q-value of action



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$$

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$$

	Up	Down	Left	Right
(1, 2)	0	0	0	0=>0
(2, 2)	0	-100	0	0

Q-learning

	Up	Down	Left	Right
(1, 2)	0	0	0	0=>-10
(2, 2)	0	-100	0	0

SARSA

Q-Learning VS. SARSA

- Recall the difference

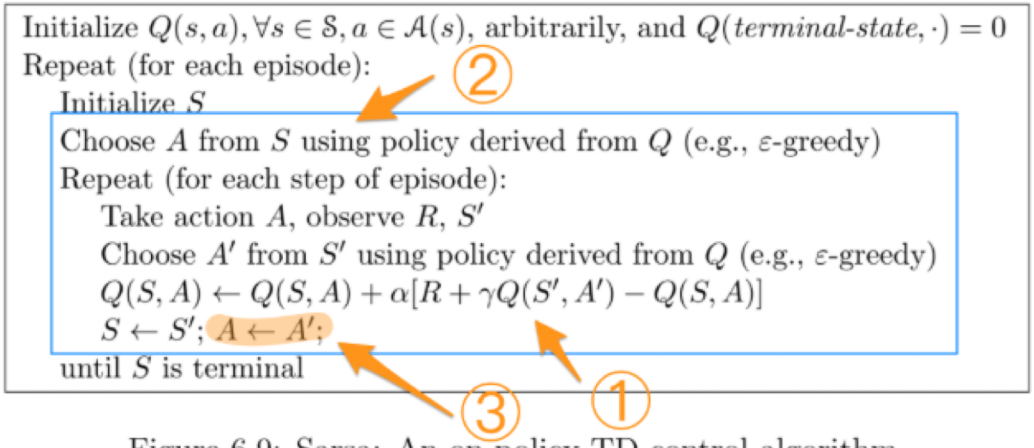


Figure 6.9: Sarsa: An on-policy TD control algorithm.

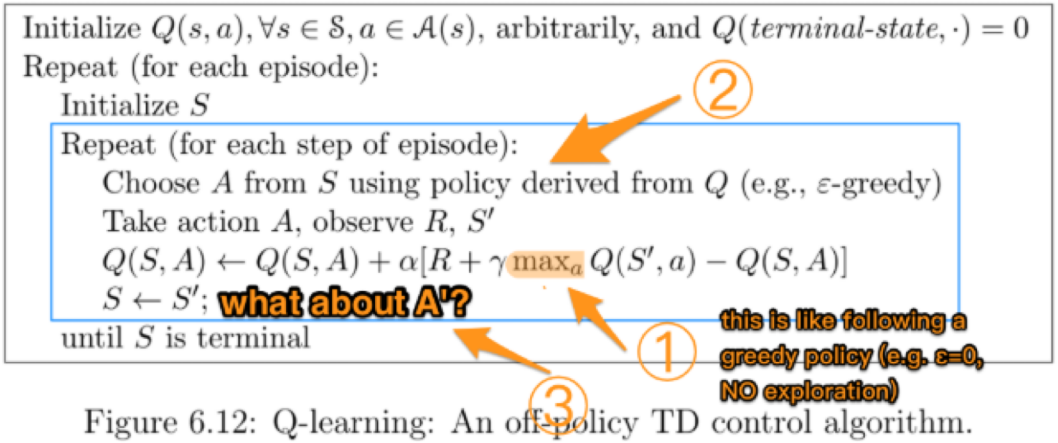


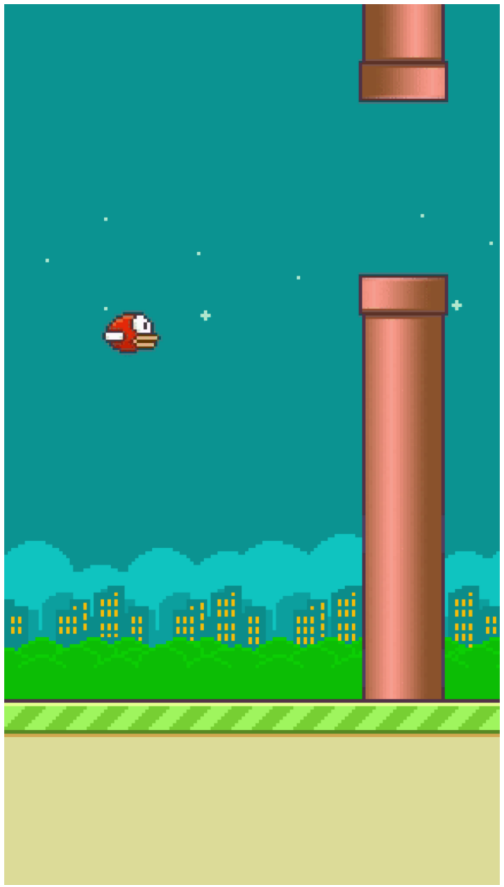
Figure 6.12: Q-learning: An off-policy TD control algorithm.

Outline

- MDP(value iteration & policy iteration)
- Q-Learning & SARSA
- Homework

Homework

- Train an agent to play Flappy Bird game(SARSA)



Install PLE

- Clone the repo

```
$ git clone https://github.com/ntasfi/PyGame-Learning-Environment
Cloning into 'PyGame-Learning-Environment'...
remote: Enumerating objects: 1118, done.
remote: Total 1118 (delta 0), reused 0 (delta 0), pack-reused 1118
Receiving objects: 100% (1118/1118), 8.06 MiB | 800.00 KiB/s, done.
Resolving deltas: 100% (592/592), done.
```

- Install PLE(in the PyGame-Learning-Environment folder)

```
$ pip install -e .
Obtaining file:///E:/DL/Lab/RL/PyGame-Learning-Environment
Requirement already satisfied: numpy in c:\users\vincent\anaconda3\lib\site-pack
ages (from ple==0.0.1) (1.16.4)
Requirement already satisfied: Pillow in c:\users\vincent\anaconda3\lib\site-pac
kages (from ple==0.0.1) (6.1.0)
Installing collected packages: ple
  Found existing installation: ple 0.0.1
    Uninstalling ple-0.0.1:
      Successfully uninstalled ple-0.0.1
  Running setup.py develop for ple
Successfully installed ple
```

Install PLE

- Quick start
  - After installing the PLE, you should create the code file in the PyGame-Learning-Environment folder

In [1]:
```
from ple.games.flappybird import FlappyBird
from ple import PLE
import matplotlib.pyplot as plt
import os
import numpy as np

%matplotlib inline
os.environ["SDL_VIDEODRIVER"] = "dummy"  # this line make pop-out window not appear
game = FlappyBird()
env = PLE(game, fps=30, display_screen=False)  # environment interface to game
env.reset_game()
```

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
couldn't import doomish
Couldn't import doom
```

Homework

- What you should do:
  - Change the update rule from Q-learning to SARSA(**with the same episodes**).
  - Give a brief report to discuss the result(compare Q-learning with SARSA based on the game result).

Homework

- Precautions:
  - If you meet this problem, just stop.

```
~\Anaconda3\lib\site-packages\moviepy\video\io\html_tools.py in html_embed(clip, filetype, maxduration, rd_kwargs, center, **html_k
wargs)
    105
    106         return html_embed(filename, maxduration=maxduration, rd_kwargs=rd_kwargs,
--> 107                           center=center, **html_kwargs)
    108
    109     filename = clip

~\Anaconda3\lib\site-packages\moviepy\video\io\html_tools.py in html_embed(clip, filetype, maxduration, rd_kwargs, center, **html_k
wargs)
    140         if duration > maxduration:
    141             raise ValueError("The duration of video %s (%.1f) exceeds the 'maxduration' "%(filename, duration)+
--> 142                              "attribute. You can increase 'maxduration', by passing 'maxduration' parameter"
    143                              "to ipython_display function."
    144                              "But note that embedding large videos may take all the memory away !")

ValueError: The duration of video __temp__.mp4 (129.8) exceeds the 'maxduration' attribute. You can increase 'maxduration', by passing 'max
duration' parameterto ipython_display function.But note that embedding large videos may take all the memory away !
```

Homework

- Precautions:
  - Only need CPU resource.
  - It will take you more than 13 hours to train, please reserve enough time.

Homework

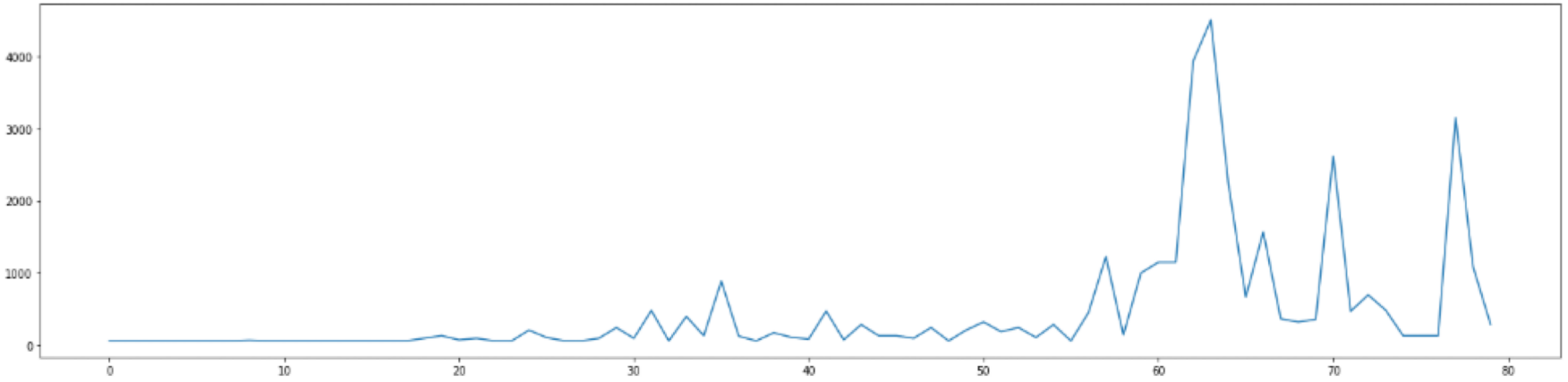
- Requirements:
  - Upload the notebook named Lab16_{strudent_id}.ipynb to demonstrate your code file and report on google drive, then submit the link to iLMS.
  - Deadline: 2019-12-26(Thur) 23:59.

Homework

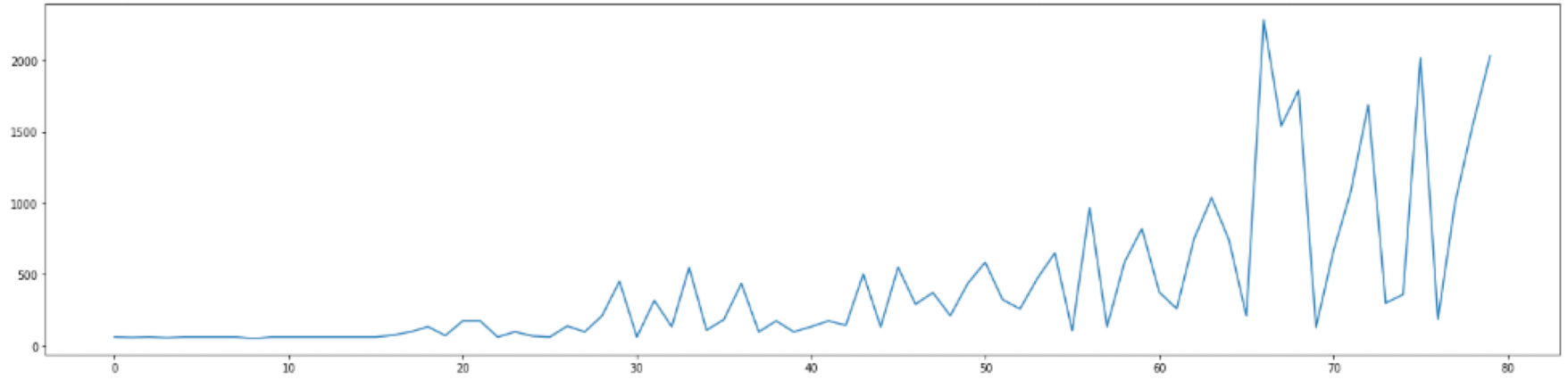
- Hints(report):
  - You can compare the plot of life time or reward against training episodes for two algorithms.

Homework

- Hints(report):



Q-learning



SARSA

Thanks!