Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

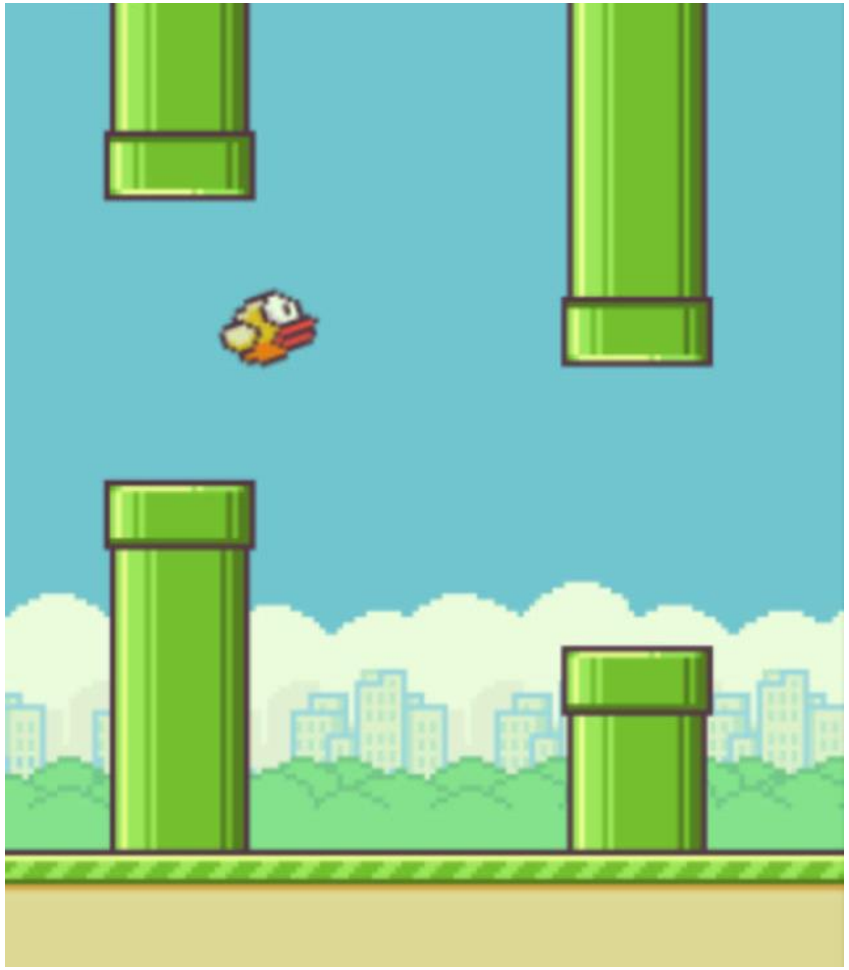
Model free Flappy Bird

Michele Alessi, Samuele D'Avenia, Elena Rivaroli

MSc Data Science and Scientific Computing

Reinforcement Learning A.Y. 2022-2023

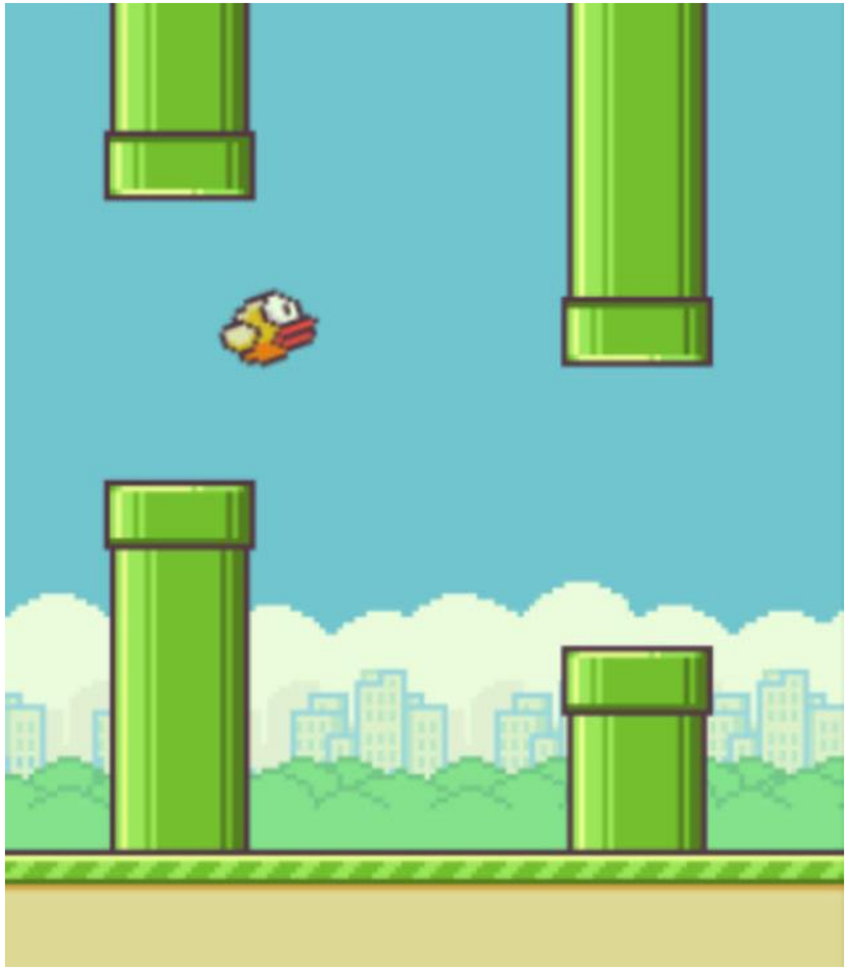
Flappy bird



- Tap -> The bird flies up
- Do not tap -> The bird moves down

Aim: keep the bird alive as long as possible

Flappy bird



- Tap -> The bird flies up
- Do not tap -> The bird moves down

Aim: keep the bird alive as long as possible

Player	Best score
Michele	5
Samuele	29
Elena	6

Text-flappy-bird environment [1]

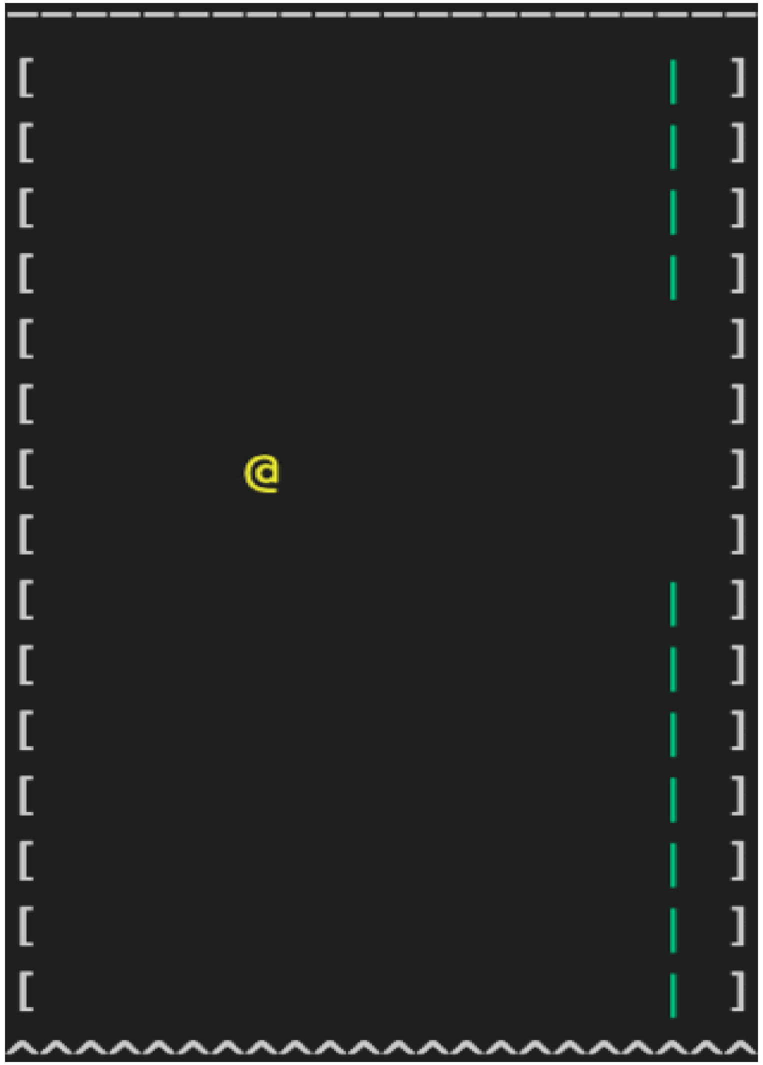
Rules:

- If you flap you move up by 1
- If you don't, you move down

Reward: +1 for every step until the game ends

Game ends:

- It touches the pipe
- It touches the floor



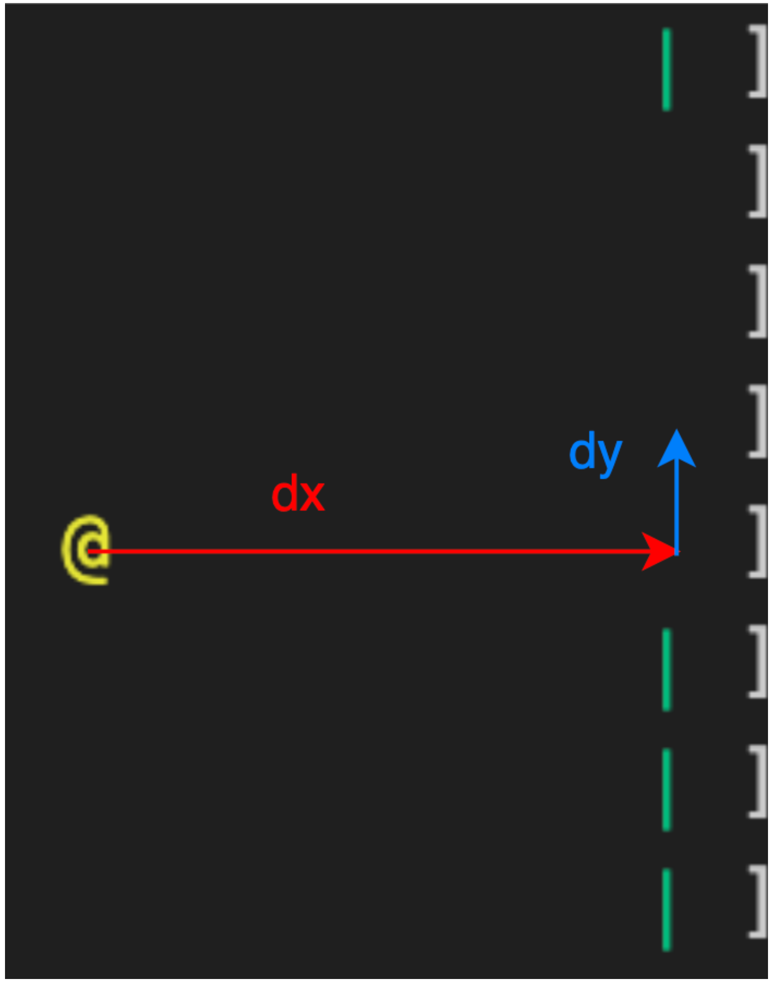
Model-free approach

States (dx, dy):

- 14 possible dx
- 22 possible dy

Actions

- 0 Remain Idle
- 1 Flap



General setup

General setup

- Techniques tried out
  - Monte Carlo Control*
  - Sarsa*
  - Expected Sarsa*
  - Q-learning**

*On-policy technique
** Off-policy technique

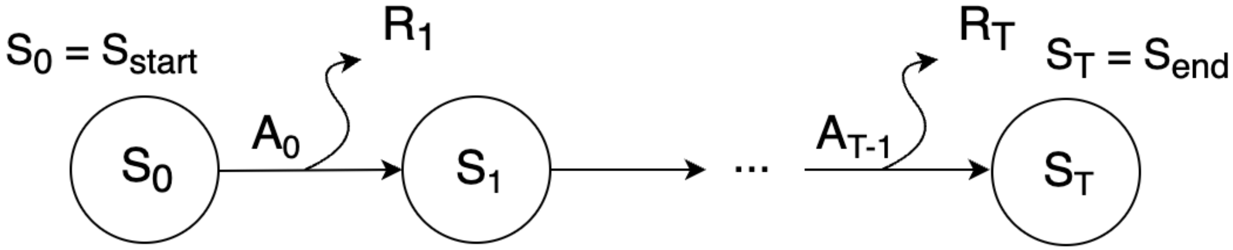
General setup

- Techniques tried out
  - Monte Carlo Control*
  - Sarsa*
  - Expected Sarsa*
  - Q-learning**
- Study on behaviour of hyper-parameters
  - λ on Sarsa and Q-learning
  - k_α on Sarsa, Q-learning and Expected Sarsa
  - k_ε on all the techniques

*On-policy technique

** Off-policy technique

Monte Carlo Control

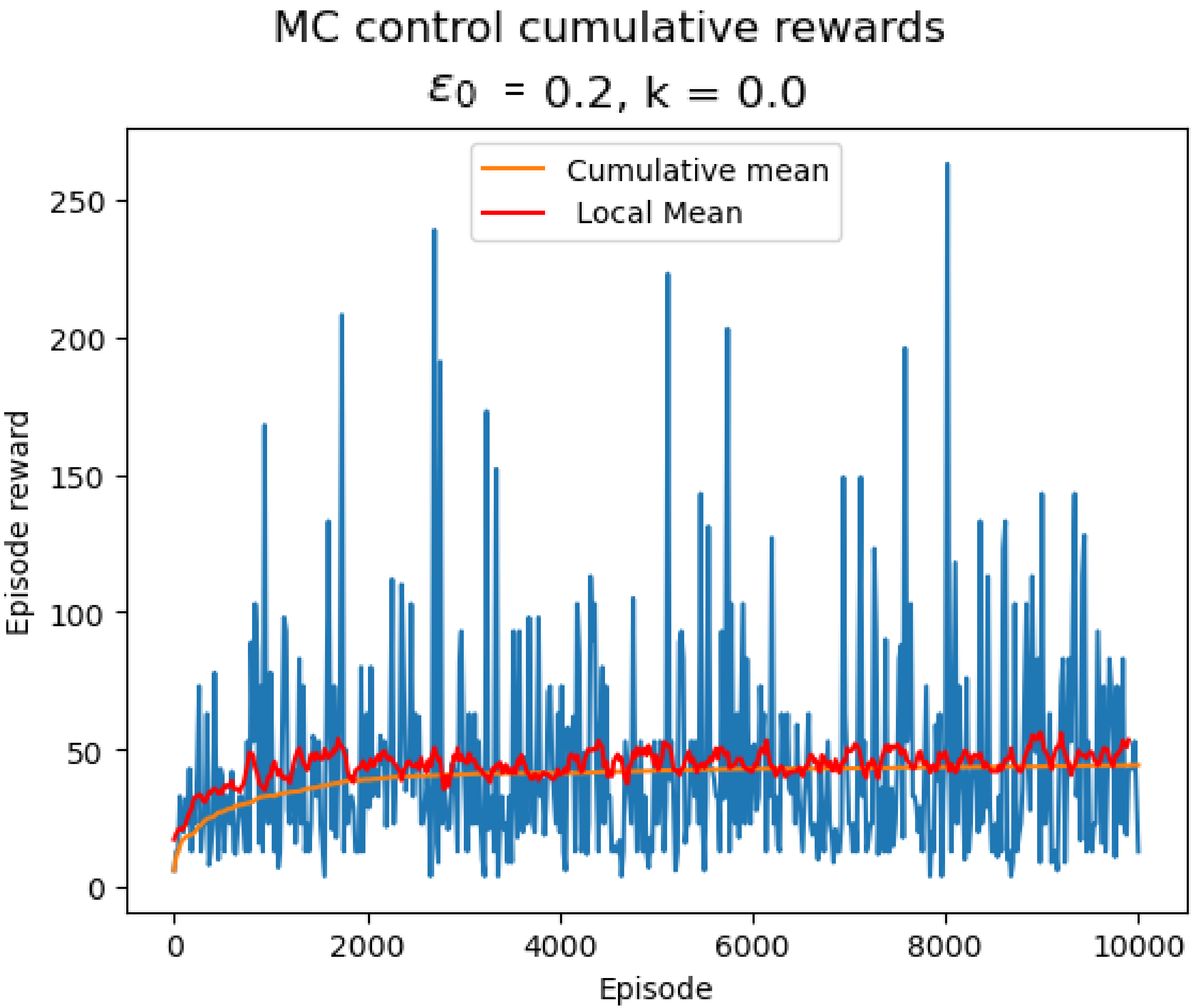


This is a on-policy control method based on first-visit MC

Update the current estimate of action-value function:

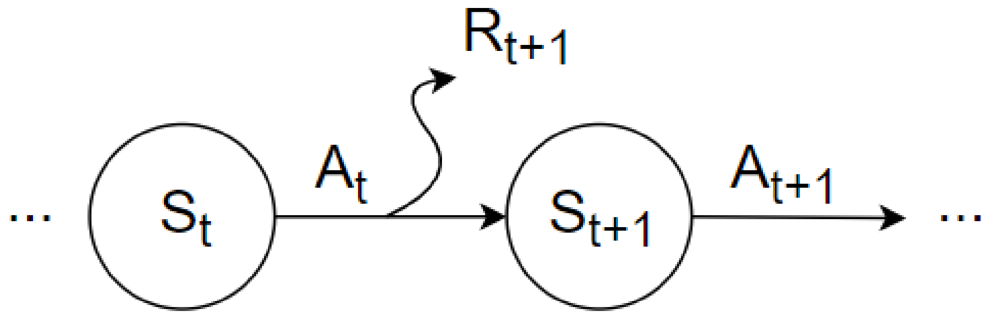
$$Q(S_t, A_t) \leftarrow \mathrm{average}(Returns(S_t))$$

$Returns(S_t, A_t)$ stores the first-visit return of (S_t, A_t) for all the episodes generated so far.



MC control cumulative rewards
ε_0 = 0.2, k = 0.0
Cumulative mean
Local Mean
Episode reward
Episode
0
50
100
150
200
250
0
2000
4000
6000
8000
10000

SARSA: On-Policy TD Control



It consists of two main ideas:

- Update the current estimate of action-value function

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Construct an ϵ-greedy policy $\pi_t^{\epsilon}(s)$ given the current Q-value

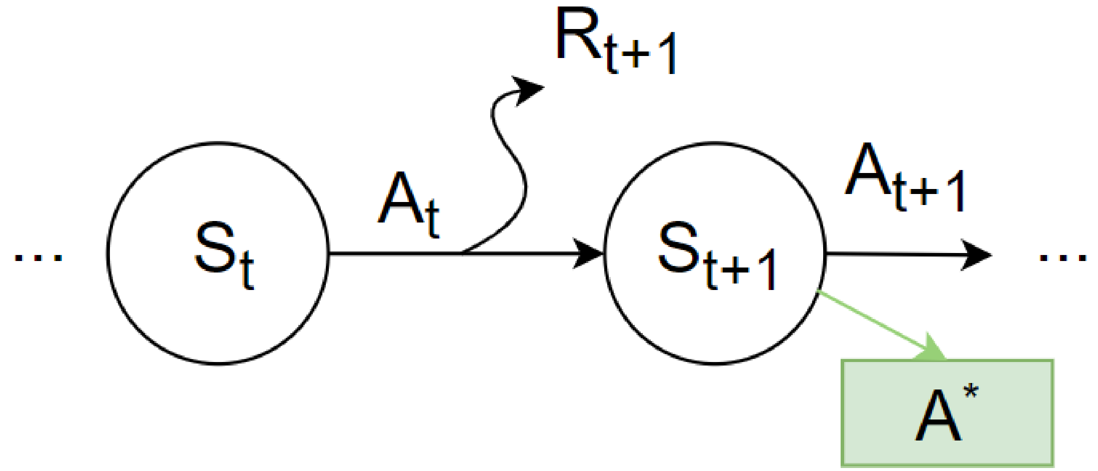
Expected SARSA

Very simple modification to SARSA method.

Update the current estimate of action-value function as follows:

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\
&\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big]
\end{aligned}
$$

Q-Learning: Off-Policy TD Control



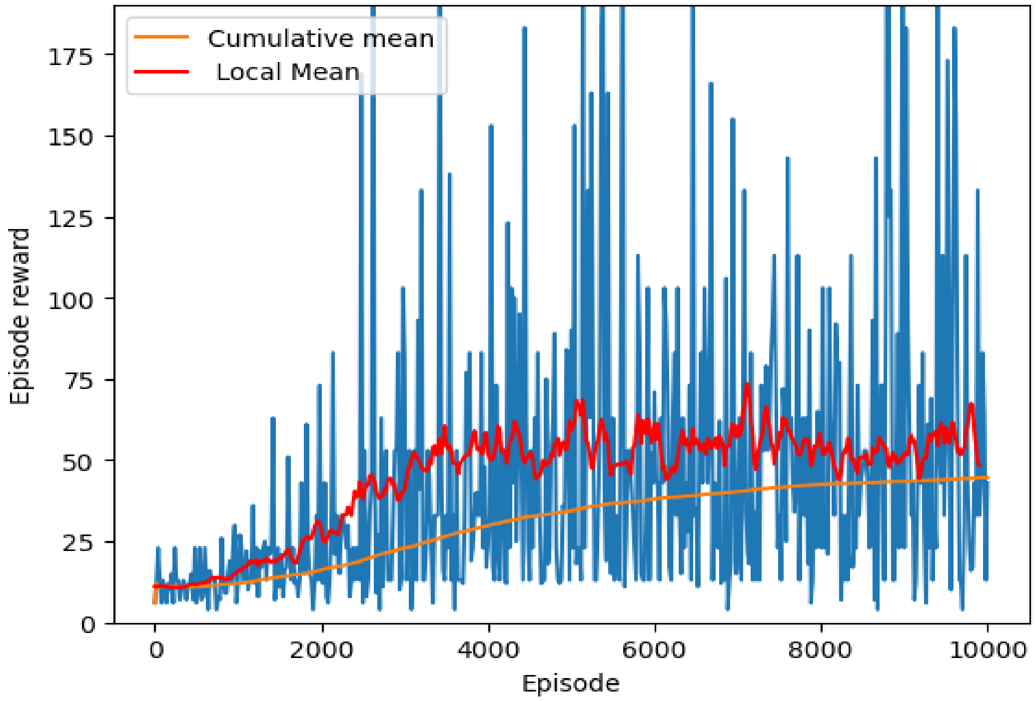
Update the current estimate of action-value function:

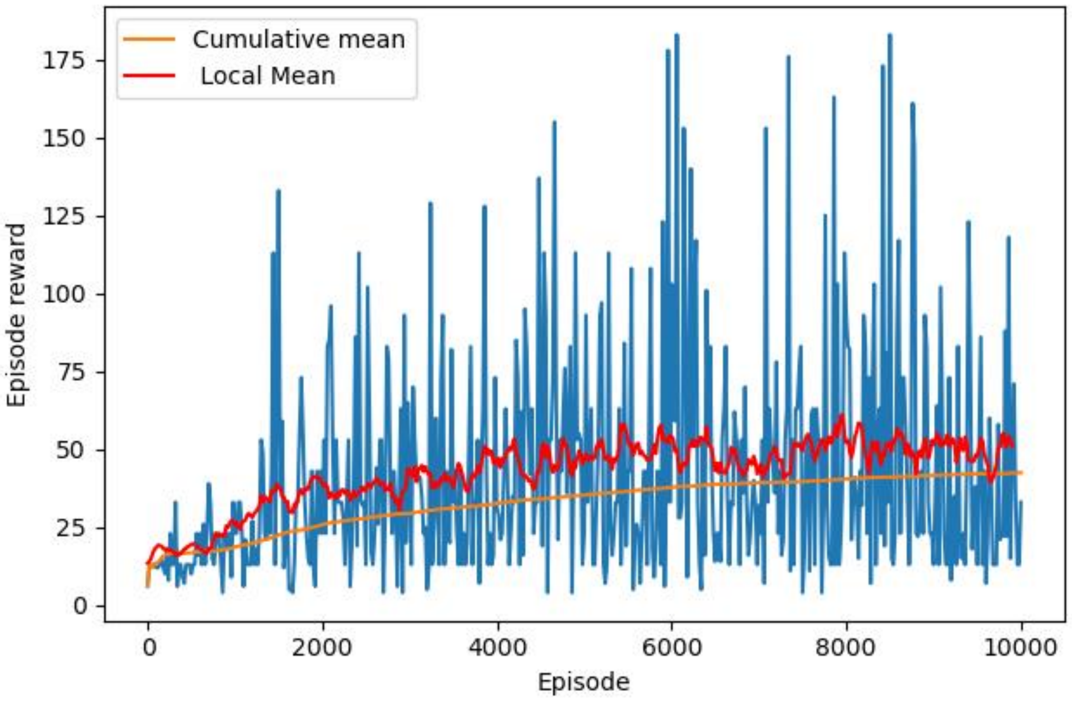
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

SARSA control cumulative rewards

Expected SARSA cumulative rewards

$$\varepsilon_0 = 0.2,\ k_\varepsilon = 0,\ \alpha_0 = 0.15,\ k_\alpha = 0$$





	SARSA	Expected SARSA
Mean	55.374	49.982
Standard Deviation	52.530	42.942

TD(λ) Learning

The eligibility trace for each state s at time t is denoted as $e_t(s)$

On each step, the eligibility traces are updated as follows:

$$e_t(s,a) \leftarrow \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t,\, a = a_t \\ \gamma\lambda e_{t-1}(s) & \text{otherwise} \end{cases}$$

- SARSA(λ) → Fairly straighforward
- Q(λ) → Some care handling eligibility traces

Hyperparameters

All the hyperparameters:

- t^*: until t^* we keep α and ε constant
- λ: determines eligibility decay
- α_0: initial value of α
- ε_0: initial value of ε
- k_α: controls how fast α goes to 0
- k_ε: controls how fast ε goes to 0

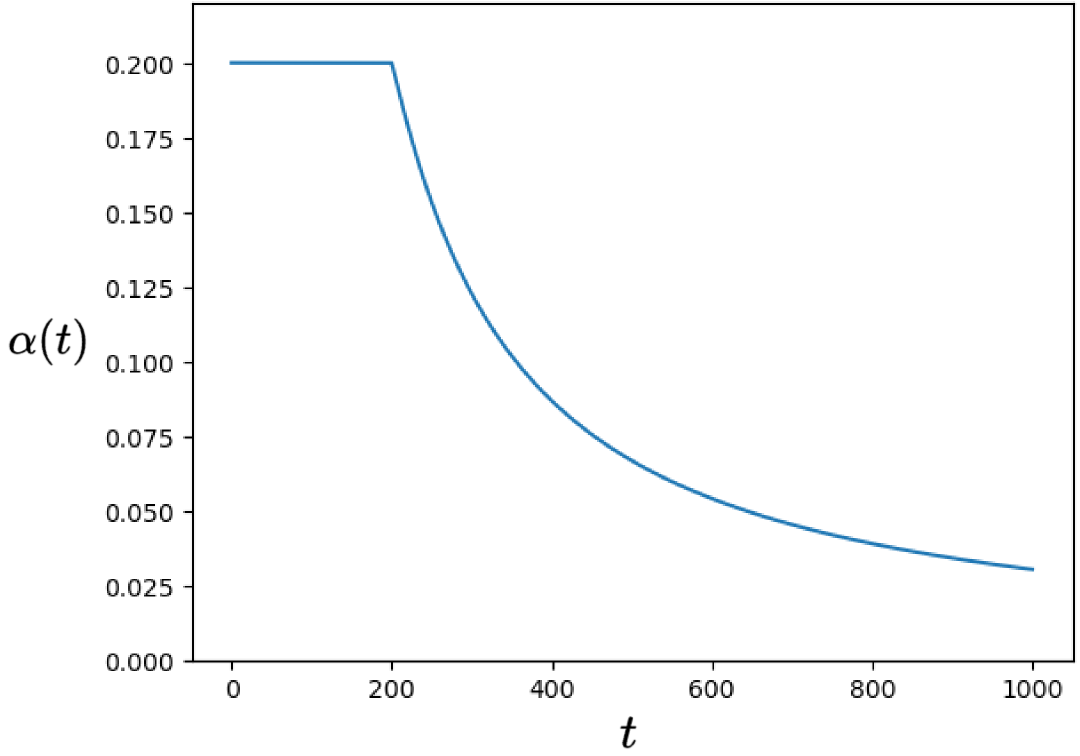
Change these

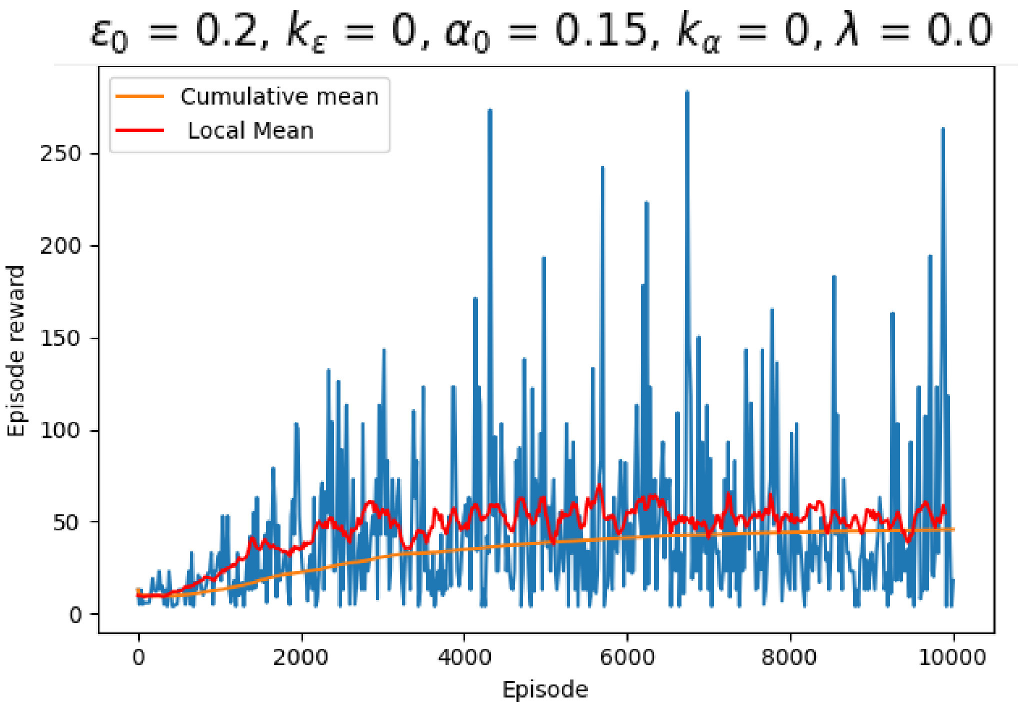
Convergence & Exploitation

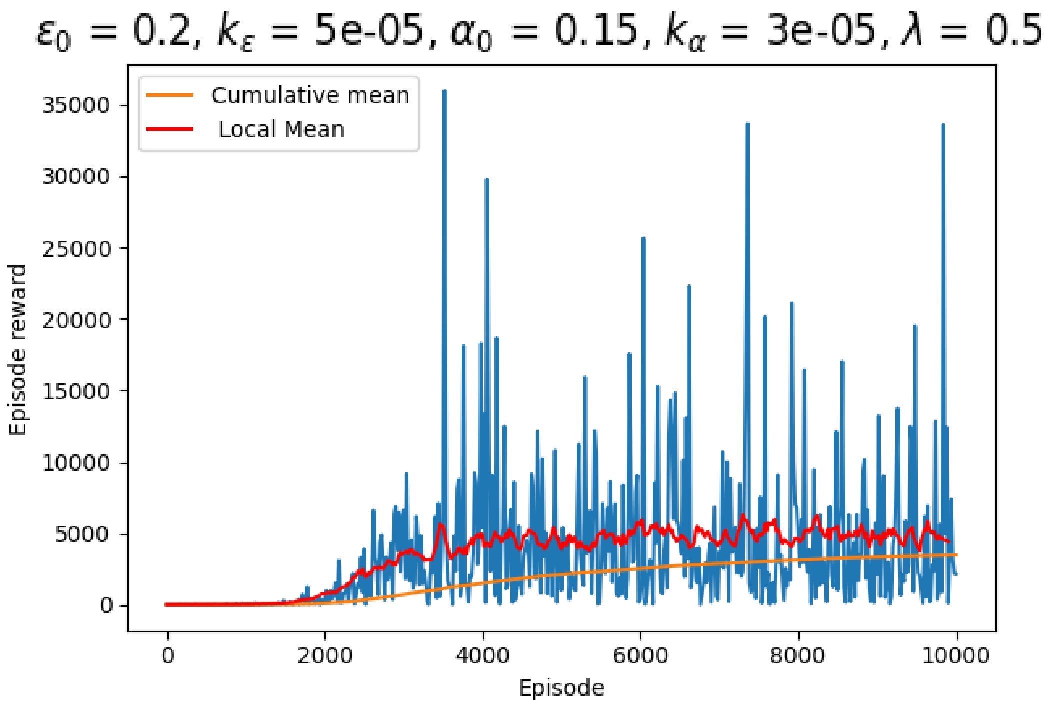
Use constant α and ϵ up to some point t^*, and then decrease them as

- $$\alpha(t) = \frac{\alpha_0}{1 + k_\alpha (t - t^\star)^{0.75}}$$

- $$\varepsilon(t) = \frac{\varepsilon_0}{1 + k_\varepsilon (t - t^\star)^{1.05}}$$

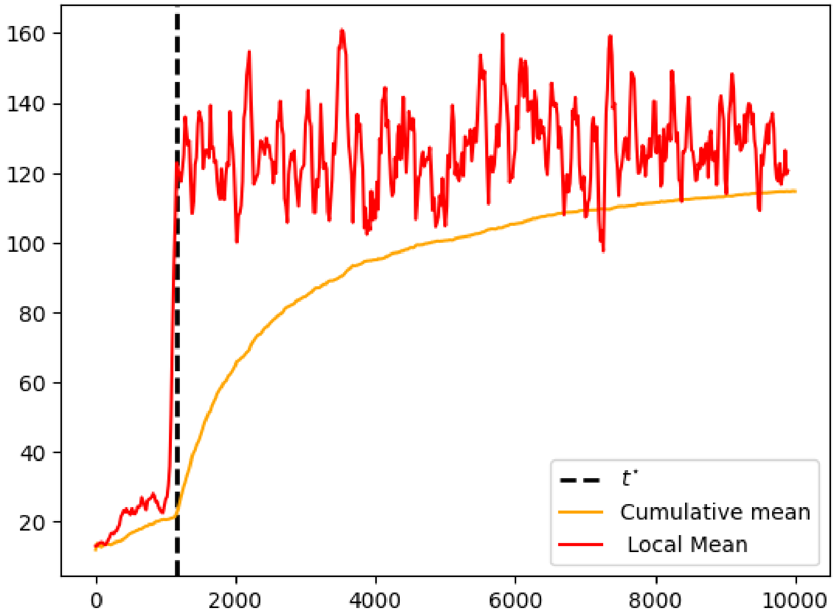






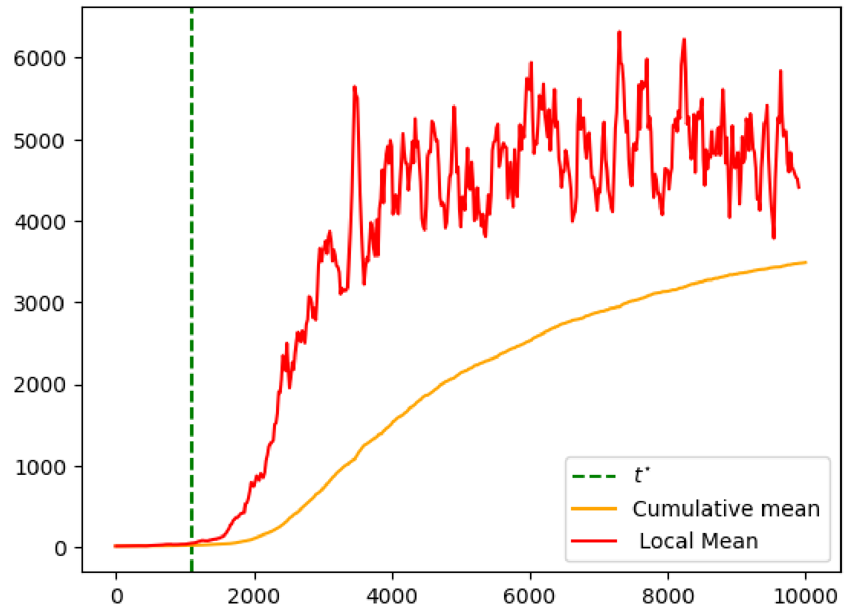
	Mean for fixed α and ε	Mean varying α and ε
Monte Carlo Control	48.74	199.46
SARSA	55.37	2633.70
Expected SARSA	49.98	127.30
Q-Learning	51.49	4783.61

Shown in the plots



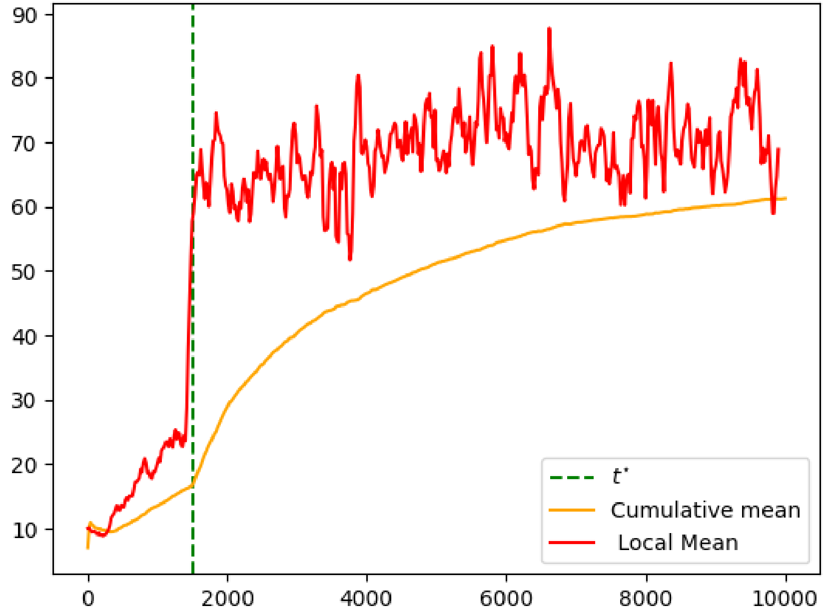
Q-Learning

$\lambda = 0.5, k_\alpha = 0.00003$



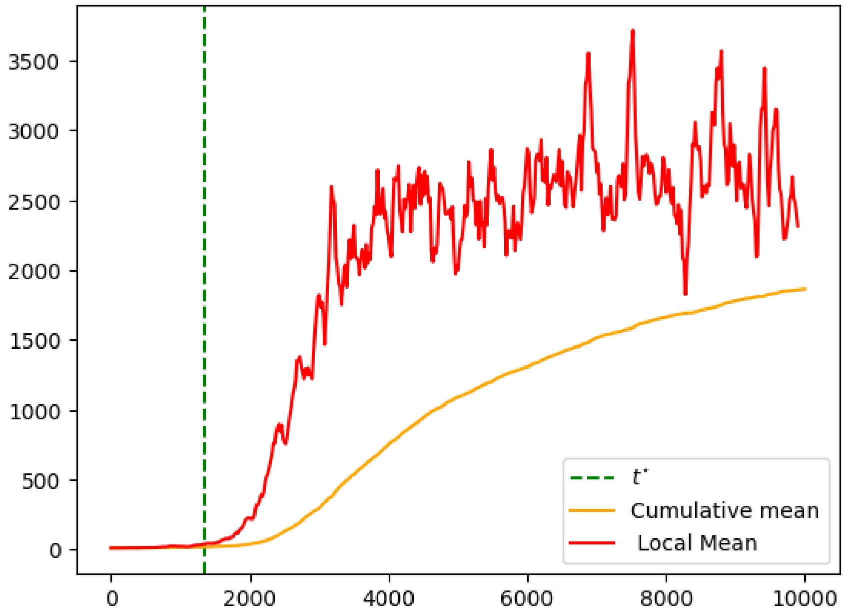
$k_\varepsilon = 0.05$

$k_\varepsilon = 0.00005$



SARSA

$\lambda = 0.2, k_\alpha = 0.00003$

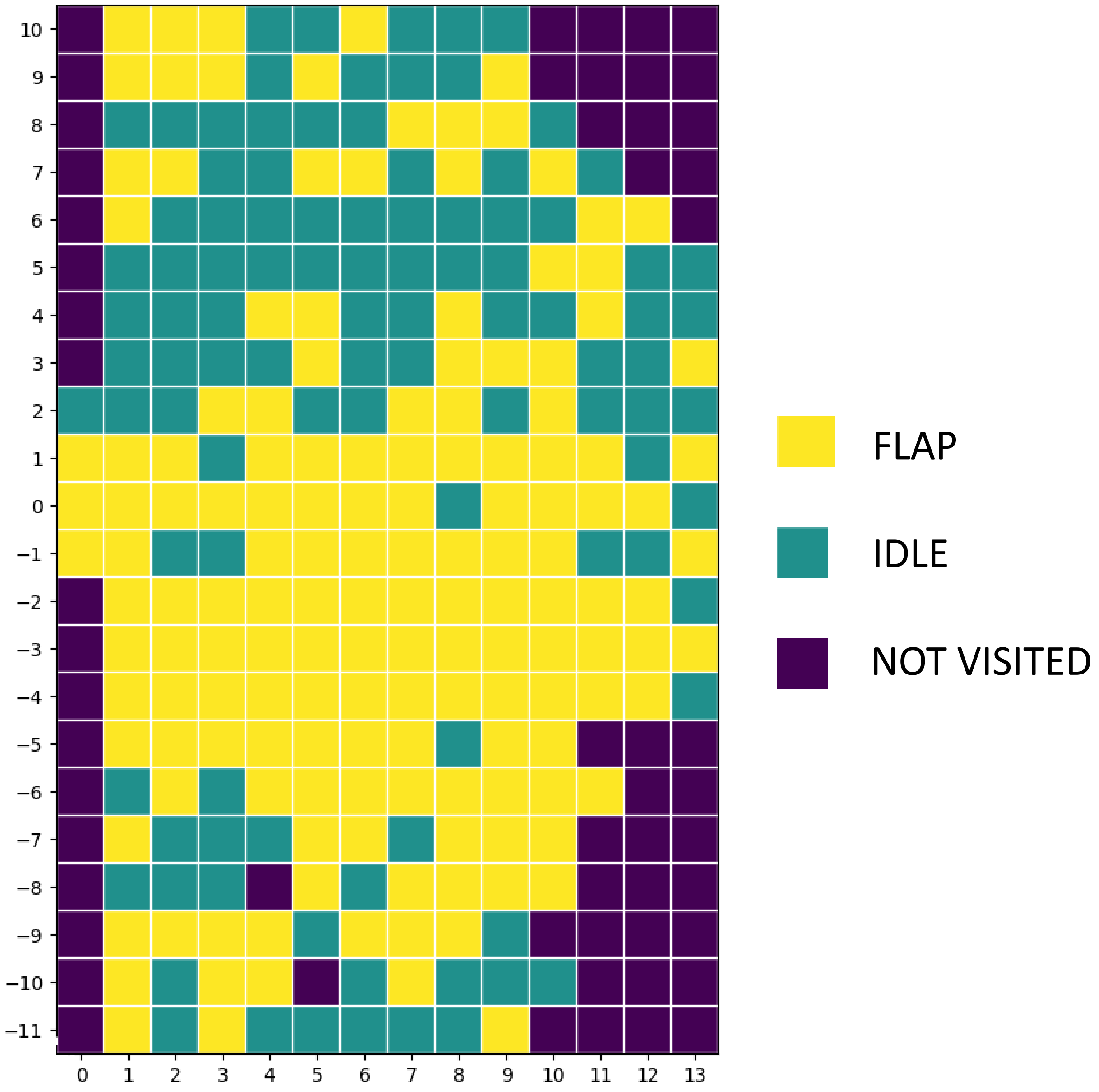


Final results

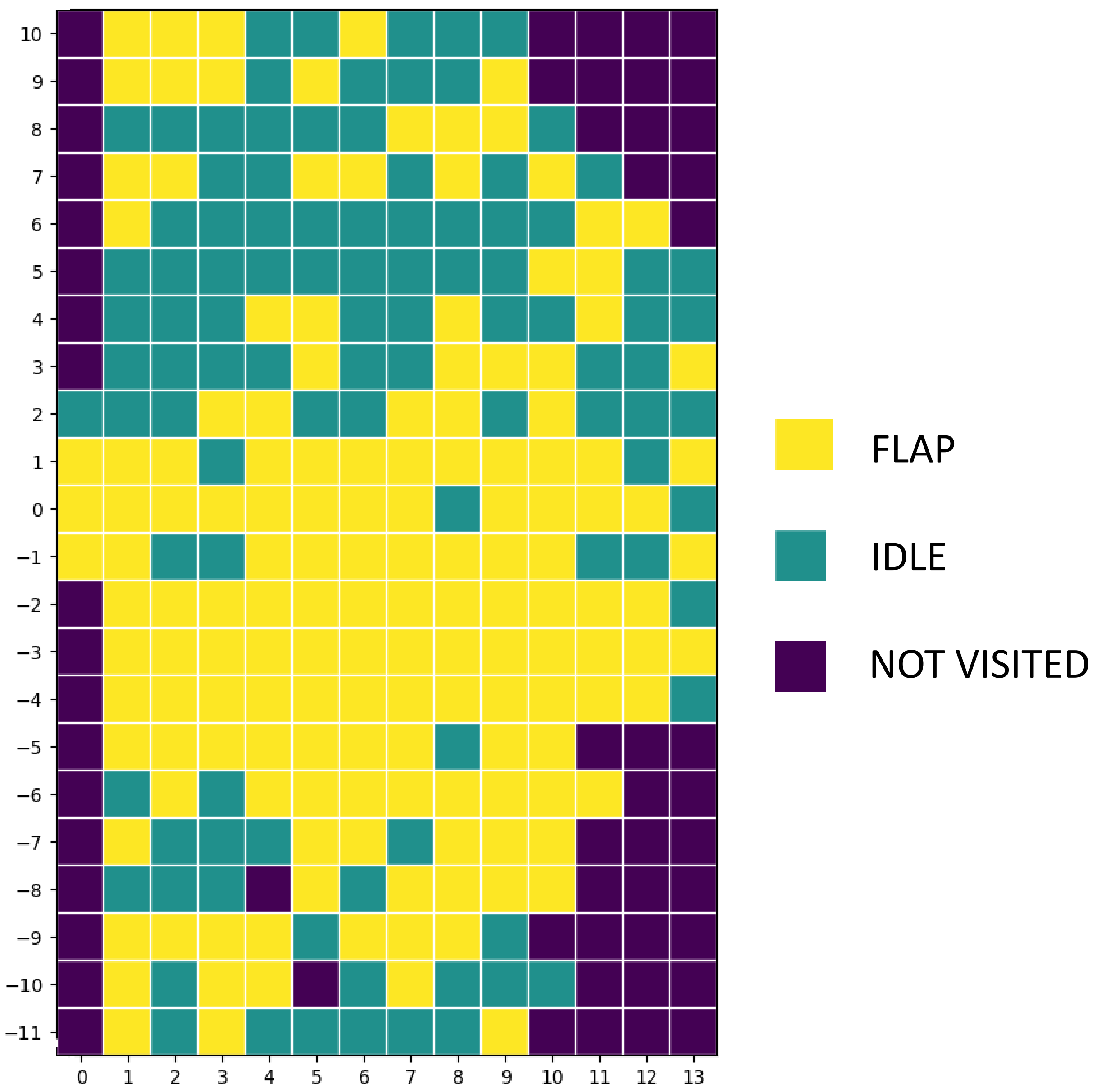
	Monte Carlo Control	SARSA	Expected SARSA	Q-Learning
λ	-	0.2	-	0.5
k_α	-	0.00003	0.0003	0.00003
k_ϵ	0.005	0.00005	0.00005	0.00005
Mean	199.46	2633.70	127.30	4783.61
Median	143	1831	93	3186
S.D.	175.70	2632.14	111.71	4952.15

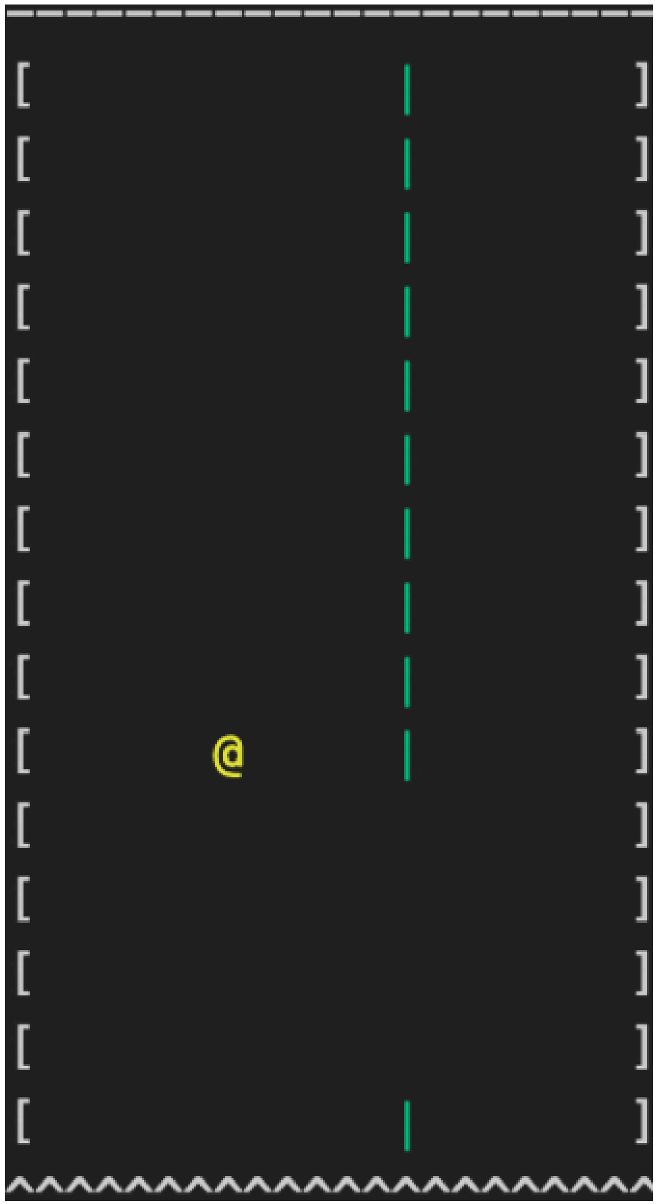
Final results

	Monte Carlo Control	SARSA	Expected SARSA	Q-Learning
λ	-	0.2	-	0.5
k_α	-	0.00003	0.0003	0.00003
k_ϵ	0.005	0.00005	0.00005	0.00005
Mean	199.46	2633.70	127.30	4783.61
Median	143	1831	93	3186
S.D.	175.70	2632.14	111.71	4952.15



10
9
8
7
6
5
4
3
2
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−9
−10
−11
0
1
2
3
4
5
6
7
8
9
10
11
12
13
FLAP
IDLE
NOT VISITED


10
9
8
7
6
5
4
3
2
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−9
−10
−11
0
1
2
3
4
5
6
7
8
9
10
11
12
13
FLAP
IDLE
NOT VISITED



Further Work

- Perform a more in depth analysis of the hyper-parameters.
- More complex environment where agent receives frames as observations.

Thanks for the attention!

References

[1] https://gitlab-research.centralesupelec.fr/stergios.christodoulidis/text-flappy-bird-gym/-/tree/master

Some notes about the code:

- For the implementation of the algorithms we started from the code provided by the course tutor, Emanuele Panizon https://www.ictp.it/member/emanuele-panizon
- Small sections of the code were implemented using ChatGPT, more details on where it was used can be found on the GitHub repository.

Appendix: Pseudocodes

Pseudocodes were taken from
http://incompleteideas.net/book/first/ebook/node1.html

MC on policy control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s, a) \leftarrow$ arbitrary
  $Returns(s, a) \leftarrow$ empty list
  $\pi \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
  (a) Generate an episode using π
  (b) For each pair s, a appearing in the episode:
    $R \leftarrow$ return following the first occurrence of s, a
    Append R to $Returns(s, a)$
    $Q(s, a) \leftarrow \text{average}(Returns(s, a))$
  (c) For each s in the episode:
    $a^* \leftarrow \arg\max_a Q(s, a)$
    For all $a \in \mathcal{A}(s)$:

$$\pi(s, a) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

SARSA(λ)

Initialize $Q(s, a)$ arbitrarily and $e(s, a) = 0$, for all s, a
Repeat (for each episode):
  Initialize s, a
  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)
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s, a) \leftarrow e(s, a) + 1$
    For all s, a:
      $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$
      $e(s, a) \leftarrow \gamma \lambda e(s, a)$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal

Q(λ)

Initialize $Q(s, a)$ arbitrarily and $e(s, a) = 0$, for all s, a
Repeat (for each episode):
  Initialize s, a
  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)
    $a^* \leftarrow \arg\max_b Q(s', b)$ (if a' ties for the max, then $a^* \leftarrow a'$)
    $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$
    $e(s, a) \leftarrow e(s, a) + 1$
    For all s, a:
      $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$
      If $a' = a^*$, then $e(s, a) \leftarrow \gamma \lambda e(s, a)$
          else $e(s, a) \leftarrow 0$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal