Reinforcement Learning
Exercise 1

Mathias Niepert, Vinh Tong

April 14, 2023

1 Multi-armed Bandits (4 Points)

a) Consider ϵ-greedy action selection for a bandit with two actions ($k = 2$) and $\epsilon = 0.5$. What is the probability that the greedy action is selected? (2P)

With probability $1 - \epsilon$ select the best action, and with probability ϵ select a random action. $\epsilon = 0$: only greedy; $\epsilon = 1$: only random.

Greedy action: $\arg\max_a Q_t(a)$

$$\pi_t(a) = \Pr\{A_t = a\} = \begin{cases} (1-\epsilon) + \frac{\epsilon}{k} & \text{if } a = \arg\max_{a'} Q_t(a') \\ \frac{\epsilon}{k} & \text{else} \end{cases}$$

$$(1 - 0.5) + \frac{0.5}{2} = 0.75$$

b) Consider a k-armed bandit problem with $k = 4$ actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ϵ-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose, you observe the following sequence of actions and rewards: $(A_1 = 1, R_2 = 1), (A_2 = 2, R_3 = 1), (A_3 = 2, R_4 = 2), (A_4 = 2, R_5 = 2), (A_5 = 3, R_6 = 0)$. On some of these time steps the ϵ case may have occurred, causing an action to be selected at random. (2P)

1. On which time steps did this definitely occur?

   $Q_1 = (0, 0, 0, 0)$, $A_1 = 1$ (greedy), $R_2 = 1$
   $Q_2 = (1, 0, 0, 0)$, $A_2 = 2$ (non-greedy), $R_3 = 1$
   $Q_3 = (1, 1, 0, 0)$, $A_3 = 2$ (greedy), $R_4 = 2$
   $Q_4 = (1, \frac{3}{2}, 0, 0)$, $A_4 = 2$ (greedy), $R_5 = 2$
   $Q_5 = (1, \frac{5}{3}, 0, 0)$, $A_5 = 3$ (non-greedy), $R_6 = 0$
   ϵ case definitely occured at time step 2 and 5.

2. On which time steps could this possibly have occurred?

   Even if the greedy action was selected, the ϵ case may have occurred.

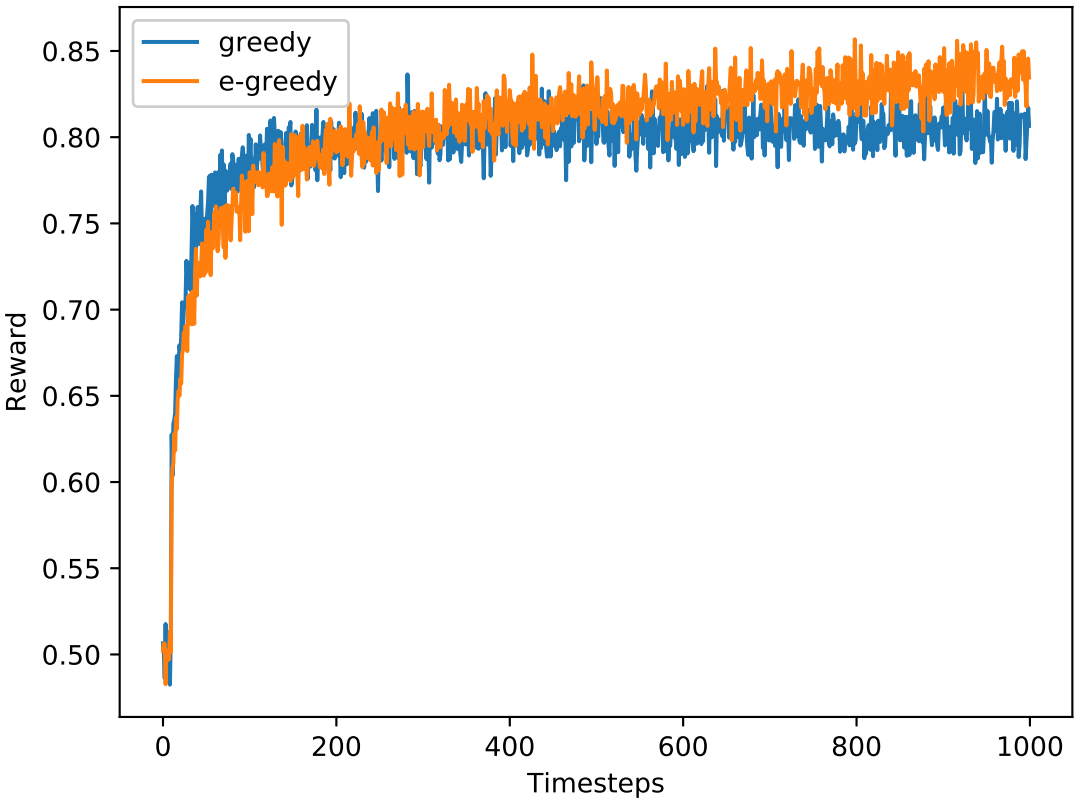
2 Action Selection Strategies (6 points)

The source code for programming exercises will be published on Ilias. The first exercise can be found as python script in *ex01-bandits/ex01-bandits.py*. The code implements a 10-armed Gaussian bandit.

a) Implement the greedy action selection strategy in the function *greedy*. Initialize the values by playing each arm once. (3P)

b) Implement the ϵ-greedy strategy in the function *epsilon_greedy*. Use $\epsilon = 0.1$. (1P)

c) In the main function set n_episodes=10000 to create a plot with less noise (this might take some time). The code template stores it as an eps file. Which of the 2 methods performs better, why? (1P)



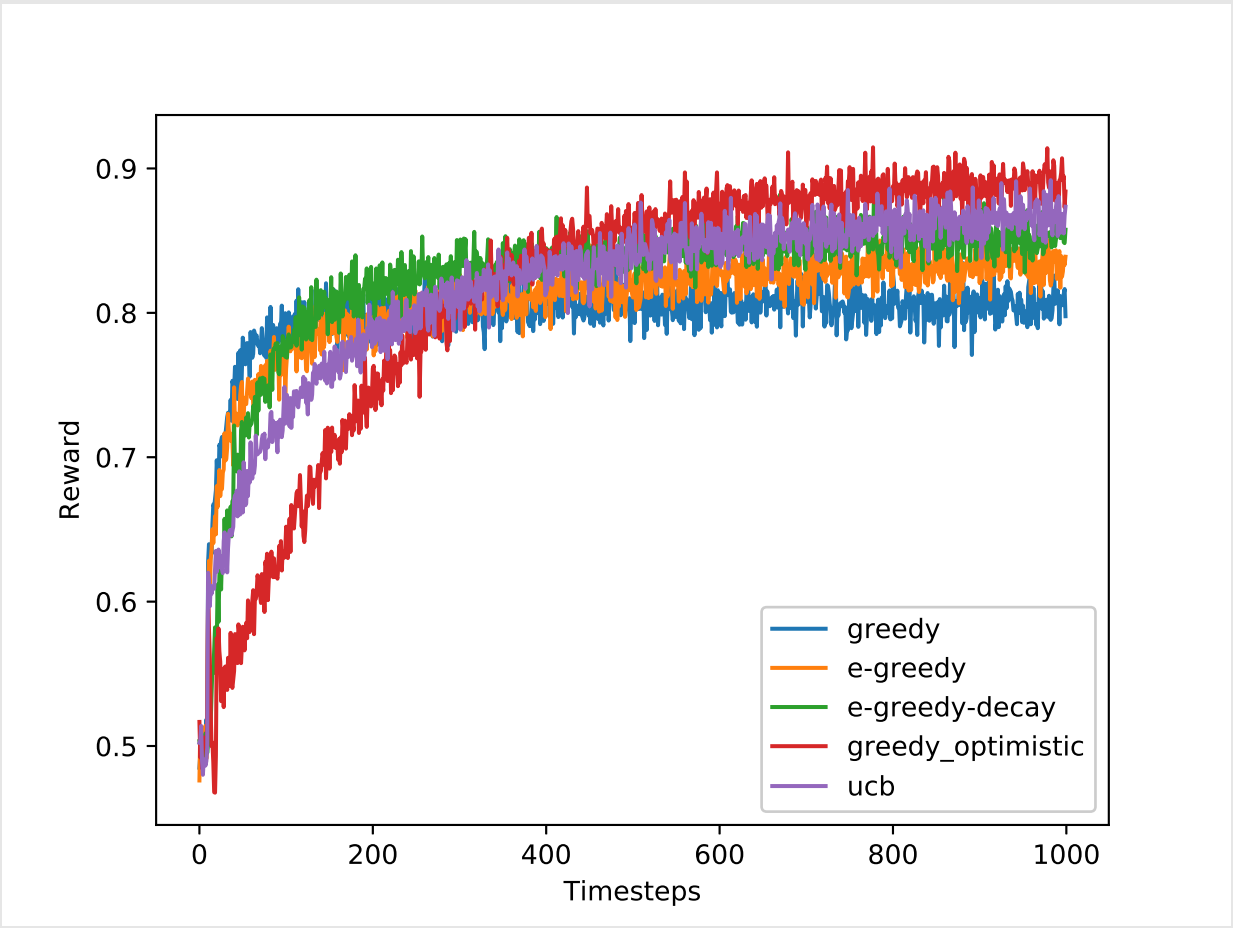
greedy is a bit better at the beginning, but often stays at suboptimal actions in the end, leading to worse performance in the limit.

e-greedy converges to the optimal Q-value and thus performs better (although it is doing suboptimal actions in the epsilon case)

d) Think about possible ways to improve the implemented methods. What changes could you make to the strategies in order to improve them? (1P)

Some possible changes:

- Decaying ϵ so that it converges to greedy
- Optimistic initialization could encourage exploration.
- One could give higher value to rarely visited states (e.g. UCB)
- Select very bad states less likely
- Use prior information over reward distribution (if available)


0.9
0.8
0.7
0.6
0.5
Reward
0
200
400
600
800
1000
Timesteps
greedy
e-greedy
e-greedy-decay
greedy_optimistic
ucb