Artificial Intelligence

Lab Assignment

Submitted by:

Md. Jawad Siddique (856384603) and

Suraj Rimal (856489906)

Problem:

Implement a 5 armed bandit problem with greedy and ε-greedy action selection algorithms. Compare the results of ε -greedy action selection method (ε =0.4) with the greedy one. Which one works better over 100 time-steps in 200 runs? You can choose any distribution/values for your reward function and/or other parameters.

Introduction

In this experiment, we are going to implement classical 5-armed bandit problem with two selection algorithms: greedy and **ε**-greedy action selection algorithm. Basically, we want to identify the bandit machine with the highest reward and exploit it. In greedy algorithm, it always exploits current knowledge and there will be no exploration and in **ε**-greedy algorithm it continues to explore and later after time it will perform better. We implemented the algorithm in python programming language. The 5-armed bandit problem with greedy and ε-greedy action selection algorithm shows the balance between exploration and exploitation.

Description of our approach:

For any learning method, we can measure its performance and behavior as it improves with experience over 100 time steps. This makes up one *run*. Repeating this for 200 independent runs, we obtained measures of the learning algorithm's average behavior. We compare the results in two graph between ε -greedy action selection method (ε =0.4) and the greedy method. The following is the code along with the results. For the five arm bandit problem, we choose bandit probability of 0.1, 0.4, 0.85, 0.35, 0.60. Each Q*(a) is chosen randomly from a normal distribution N(0,1). Each reward distribution is also normal chosen randomly.

Code:

```
import numpy as np
import matplotlib.pyplot as plt

#Bandit class
class Bandit:
    def __init__(self, bandit_probs):
        self.N = len(bandit_probs)  # no. of bandits
        self.prob = bandit_probs  # success probabilities for each bandit

    # success 1, failure 0
    def get_reward(self, action):
```

```
        rand = np.random.random()  # [0.0,1.0)
        reward = 1 if (rand < self.prob[action]) else 0
        return reward
#Agent class
class Agent:
    def __init__(self, bandit, epsilon):
        self.epsilon = epsilon
        self.k = np.zeros(bandit.N, dtype=np.int)
        self.est_values = np.zeros(bandit.N, dtype=np.float)

    def update_est(self,action,reward):
        self.k[action] += 1
        alpha = 1./self.k[action]
        self.est_values[action] += alpha * (reward - self.est_values[action])
    #action using e-greedy
    def get_action(self, bandit, force_explore=False):
        rand = np.random.random()  # [0.0,1.0)
        if (rand < self.epsilon) or force_explore:
            action_explore = np.random.randint(bandit.N)
            return action_explore
        else:
            action_greedy = np.random.choice(np.flatnonzero(self.est_values ==
self.est_values.max()))
            return action_greedy
#method to perform experiment
def experiment(agent, bandit, N_episodes):
    action_history = []
    reward_history = []
    for episode in range(N_episodes):
        action = agent.get_action(bandit)
        reward = bandit.get_reward(action)
        agent.update_est(action, reward)
        action_history.append(action)
        reward_history.append(reward)
    return np.array(action_history), np.array(reward_history)

if __name__ == "__main__":
    bandit_probs = [0.10, 0.40, 0.85, 0.35, 0.60]
    N_experiments = 100
    N_episodes = 200
    epsilon_greedy = 0.4
    greedy = 0.0
    N_bandits = len(bandit_probs)
    epsilon_reward_history_avg = np.zeros(N_episodes)
    epsilon_action_history_sum = np.zeros((N_episodes, N_bandits))
    greedy_reward_history_avg = np.zeros(N_episodes)
```

```
    greedy_action_history_sum = np.zeros((N_episodes, N_bandits))
    for i in range(N_experiments):
        bandit = Bandit(bandit_probs)
        # initializing  agents
        epsilon_agent = Agent(bandit, epsilon_greedy)
        greedy_agent = Agent(bandit, greedy)
        (greedy_action_history, greedy_reward_history) = experiment(greedy_agent, bandit,
N_episodes)
        (action_history, reward_history) = experiment(epsilon_agent, bandit, N_episodes)
        # Adding experiment reward
        epsilon_reward_history_avg += reward_history
        greedy_reward_history_avg += greedy_reward_history
        # Adding action history
        for j, (a) in enumerate(action_history):
            epsilon_action_history_sum[j][a] += 1
        for j, (a) in enumerate(greedy_action_history):
            greedy_action_history_sum[j][a] += 1

    epsilon_reward_history_avg /= np.float(N_experiments)
    print("epsilon reward history avg = {}".format(epsilon_reward_history_avg))
    print("")
    greedy_reward_history_avg /= np.float(N_experiments)
    print("greedy reward history avg = {}".format(greedy_reward_history_avg))

    # Plotting reward results
    plt.plot(greedy_reward_history_avg, label="greedy(ε=0)")
    plt.plot(epsilon_reward_history_avg, label="ε-greedy(ε=0.4)")
    plt.xlabel("Episode number")
    plt.ylabel("Average Rewards ".format(N_experiments))
    plt.title("Bandit reward history averaged over {} experiments for (epsilon = {} and
{})".format(N_experiments, greedy, epsilon_greedy))
    leg = plt.legend(loc='upper left', shadow=True, fontsize=12)
    plt.xlim([1, N_episodes])
    for legobj in leg.legendHandles:
        legobj.set_linewidth(8.0)
    plt.show()

    # Plotting action results
    plt.figure(figsize=(18, 12))
    i = bandit_probs.index(max(bandit_probs))
    epsilon_action_history_sum_plot = 100 * epsilon_action_history_sum[:, i] / N_experiments
    plt.plot(list(np.array(range(len(epsilon_action_history_sum_plot)))+1),
             epsilon_action_history_sum_plot,
             linewidth=5.0,
             label="ε = {}".format(epsilon_greedy))
    greedy_action_history_sum_plot = 100 * greedy_action_history_sum[:, i] / N_experiments
```

```
    plt.plot(list(np.array(range(len(greedy_action_history_sum_plot))) + 1),
             greedy_action_history_sum_plot,
             linewidth=5.0,
             label="ε = {}".format(greedy))
    plt.title("Optimal bandit action history averaged over {} experiments".format(N_experiments),
fontsize=26)
    plt.xlabel("Episode Number", fontsize=26)
    plt.ylabel("Optimal Action Choices (%)", fontsize=26)
    leg = plt.legend(loc='upper left', shadow=True, fontsize=26)
    plt.xlim([1, N_episodes])
    plt.ylim([0, 100])
    plt.xticks(fontsize=24)
    plt.yticks(fontsize=24)
    for legobj in leg.legendHandles:
        legobj.set_linewidth(16.0)
    plt.show()
```

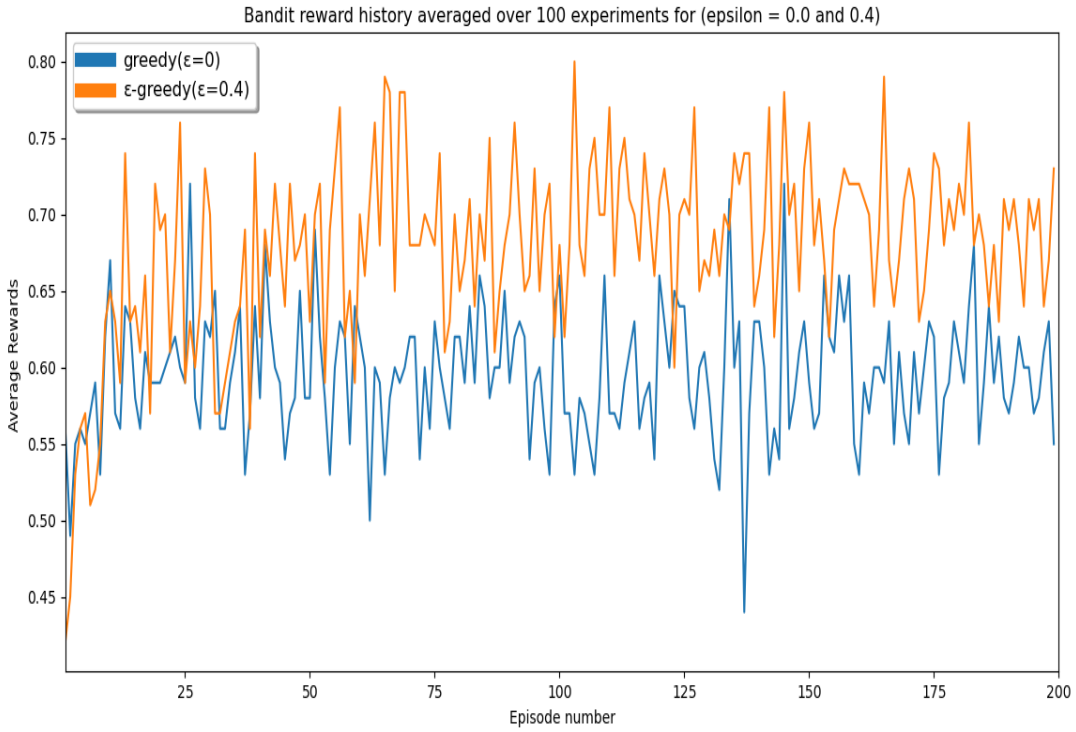

Figure 1: Comparison with greedy and ε -greedy with respect to bandit average rewards

Here, Figure 1 compares a greedy method with ε -greedy methods (ε = 0.4) on the 5-armed bandit problem. The graph shows the increase in expected reward with experience. The greedy method improved slightly faster than the other methods at the very beginning, but then leveled at a lower level. It achieved a reward-per-step of only about 0.72, compared with the best possible of about 0.80 in ε -greedy methods. The greedy method performed significantly worse in the long run

because it often got stuck performing suboptimal actions. In this case the ε -greedy methods works better than the greedy method over 100 time steps in 200 runs.

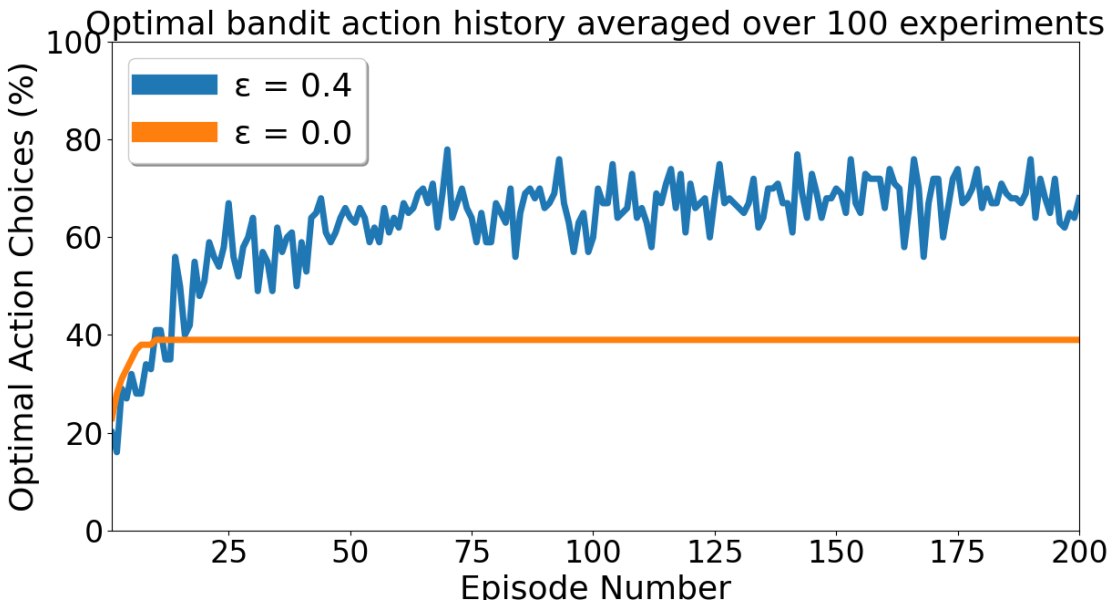

Figure 2: Comparison with greedy and ε -greedy with respect to bandit average rewards

Moreover, Figure 2 shows that the greedy method found the optimal action in only approximately two-fifth (40%) of the tasks. In the other three-fifths, its initial samples of the optimal action were disappointing, and it never returned to it. The ε -greedy methods eventually performed better because they continued to explore and to improve their chances of recognizing the optimal action. The ε = 0.4 method explored more, and usually found the optimal action earlier, but it never selected that action more than 80% of the time.

Discussion

We performed the experiment for both greedy and epsilon greedy method where ε=04. We found that greedy method improved faster at the very beginning, but it does not improve over time. The ε- greedy perform better because this method continues to explore. The weakness of greedy method is that it always exploits based on the current knowledge and it does not explore for the better rewards. The weakness of ε- greedy method is that it randomly selects an action with probability ε to explore and it does not explore more promising actions. Also, if same action has taken for many times then there is no need to explore furthermore. The ε- greedy method does not take any confidence interval.

Conclusion:

From the above explanation we can say that the ε -greedy method is better than greedy method.