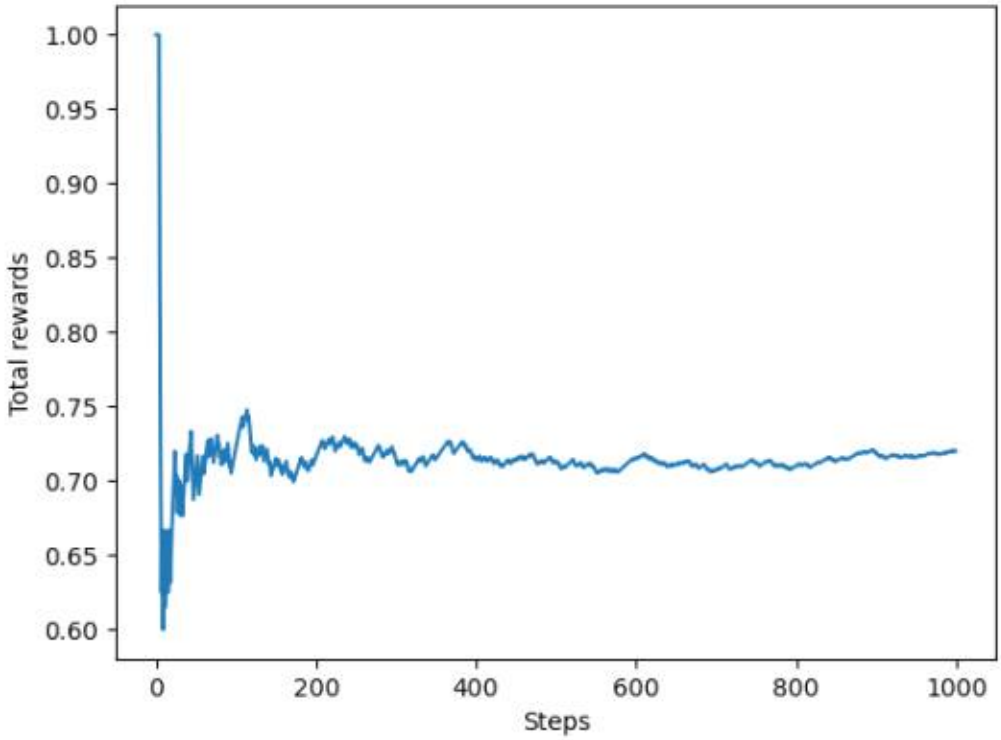
```python
if __name__ == '__main__':
    steps = 1000
    epsilon = 0.1
    bandit = Bandit()
    agent = Agent(epsilon)
    total_reward = 0
    total_rewards = []
    rate = []
    for step in range(steps):
        action = agent.get_action()
        reward = bandit.play(action)
        agent.update(action, reward)
        total_reward += reward
        total_rewards.append(total_reward/(step +1))
        print(total_reward)

    plt.ylabel('Total rewards')
    plt.xlabel('Steps')
    plt.plot(total_rewards)
    plt.show()
    plt.ylabel('Rates')
    plt.xlabel('Steps')
    plt.plot(rate)
    plt.show()
```

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


class Bandit:  1 usage
    def __init__(self, arms = 10):
        self.rates = np.random.rand(arms)


    def play(self, arm):  1 usage
        rate = self.rates[arm]
        if rate > np.random.rand():
            return 1
        else:
            return 0


class Agent:  1 usage
    def __init__(self, epsilon, action_size=10):
        self.epsilon = epsilon
        self.Qs = np.zeros(action_size)
        self.ns = np.zeros(action_size)

    def update(self, action, reward):  1 usage
        self.ns[action] += 1
        self.Qs[action] += (reward - self.Qs[action] / self.ns[action])
```

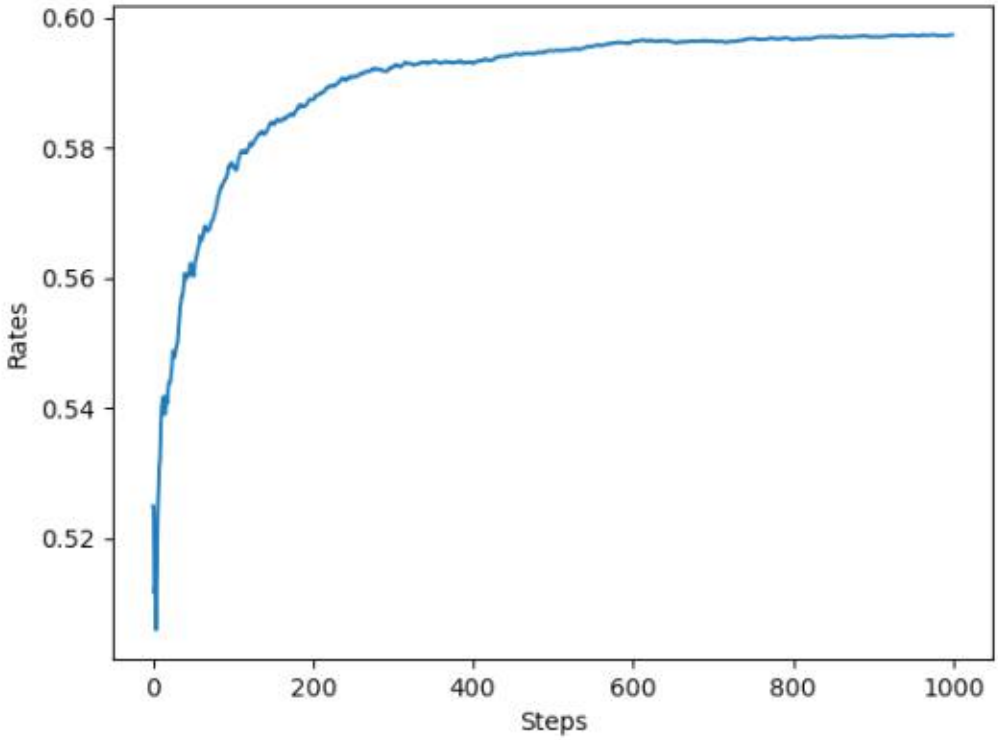
```python
import numpy as np
import matplotlib.pyplot as plt


class Bandit:  1 usage
    def __init__(self, arms = 10):
        self.rates = np.random.rand(arms)


    def play(self, arm):  1 usage
        rate = self.rates[arm]
        if rate > np.random.rand():
            return 1
        else:
            return 0


class Agent:  1 usage
    def __init__(self, epsilon, action_size=10):
        self.epsilon = epsilon
        self.Qs = np.zeros(action_size)
        self.ns = np.zeros(action_size)

    def update(self, action, reward):  1 usage
        self.ns[action] += 1
        self.Qs[action] += (reward - self.Qs[action] / self.ns[action])
```

```python
if __name__ == '__main__':
    steps = 1000
    epsilon = 0.1
    bandit = Bandit()
    agent = Agent(epsilon)
    total_reward = 0
    total_rewards = []
    rate = []
    for step in range(steps):
        action = agent.get_action()
        reward = bandit.play(action)
        agent.update(action, reward)
        total_reward += reward
        total_rewards.append(total_reward/(step +1))
        print(total_reward)

    plt.ylabel('Total rewards')
    plt.xlabel('Steps')
    plt.plot(total_rewards)
    plt.show()
    plt.ylabel('Rates')
    plt.xlabel('Steps')
    plt.plot(rate)
    plt.show()
```


1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
Total rewards
0
200
400
600
800
1000
Steps

실습 2

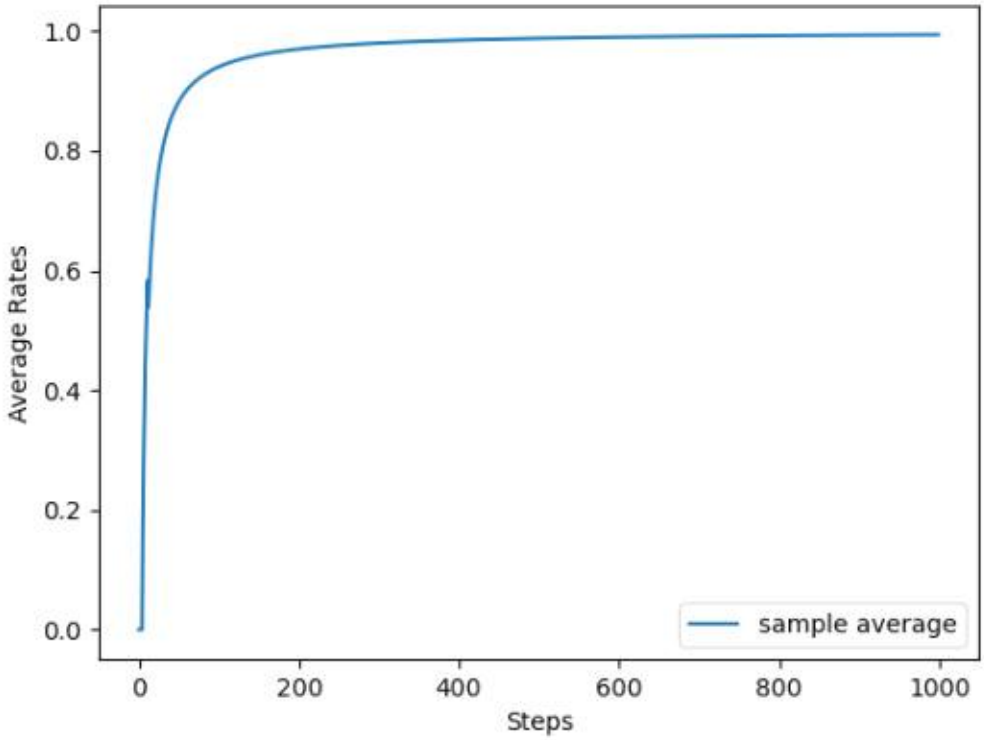
```python
runs = 200
steps = 1000
epsilon = 0.1
all_rates = np.zeros((runs, steps))


for run in range(runs):
    bandit = Bandit()
    agent = Agent(epsilon)
    total_reward = 0
    rates = []

    for step in range(steps):
        action = agent.get_action()
        reward = bandit.play(action)
        agent.update(action, reward)
        total_reward += reward
        rates.append(total_reward / (step+1))


    all_rates[run] = rates


avg_rates = np.average(all_rates, axis=0)
plt.ylabel('Rates')
plt.xlabel('Steps')
plt.plot(avg_rates)
plt.show()
```

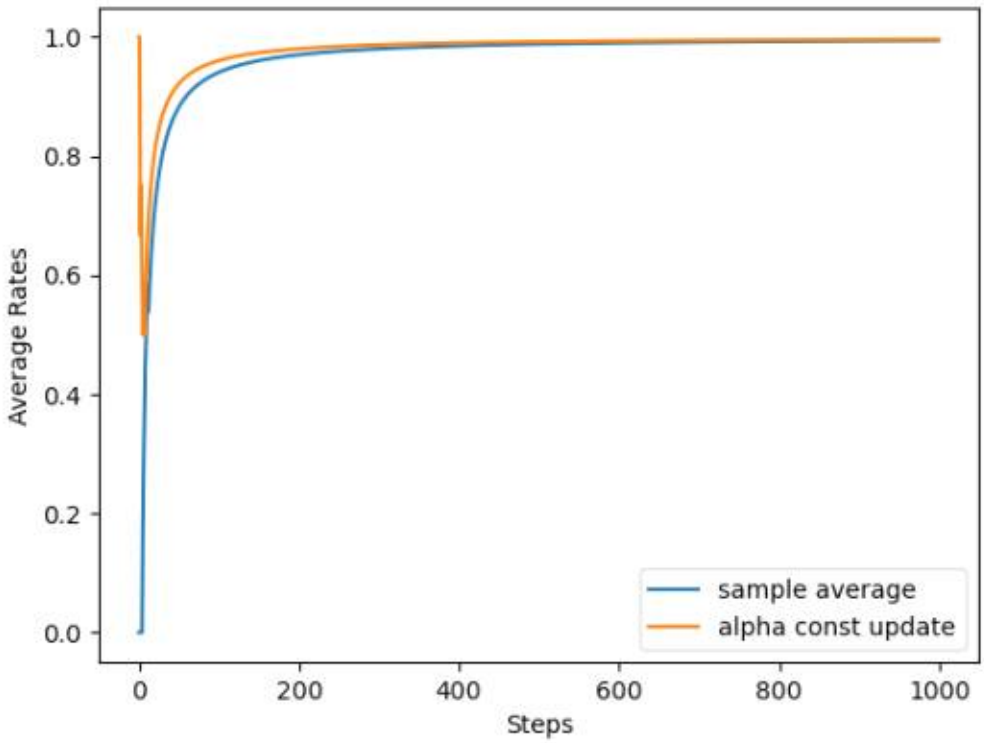


실습 3

```python
class NonStatBandit:  1 usage
    def __init__(self, arms=10):
        self.arms = arms
        self.rates = np.random.rand(arms)

    def play(self, arm):  1 usage
        rate = self.rates[arm]
        self.rates += 0.1 * np.random.rand(self.arms)

        if rate > np.random.rand():
            return 1
        else:
            return 0

class AlphaAgent:  1 usage
    def __init__(self, epsilon, alpha, action_size=10):
        self.epsilon = epsilon
        self.Qs = np.zeros(action_size)
        self.alpha = alpha

    def update(self, action, reward):  1 usage
        self.Qs[action] += (reward - self.Qs[action]) * self.alpha

    def get_action(self):  1 usage
        if np.random.rand() < self.epsilon:
            return np.random.randint( low: 0, len(self.Qs))
```

```python
if __name__ == '__main__':
    runs = 200
    steps = 1000
    epsilon = 0.1
    alpha = 0.8
    all_rates = np.zeros((runs, steps))

    agent_types = ['sample average', 'alpha const update']
    result = {}

    for agent_type in agent_types:
        for run in range(runs):
            if agent_types == 'sample average':
                agent = Agent(epsilon)
            else:
                agent = AlphaAgent(epsilon, alpha)


            bandit = NonStatBandit()
            total_reward = 0
            rates = []

            for step in range(steps):
                action = agent.get_action()
                reward = bandit.play(action)
                agent.update(action, reward)
                total_reward += reward
                rates.append(total_reward / (step+1))

            all_rates[run] = rates
            avg_rates = np.average(all_rates, axis=0)
```


1.0
0.8
0.6
0.4
0.2
0.0
Average Rates
sample average
0
200
400
600
800
1000
Steps


1.0
0.8
0.6
0.4
0.2
0.0
Average Rates
sample average
alpha const update
0
200
400
600
800
1000
Steps