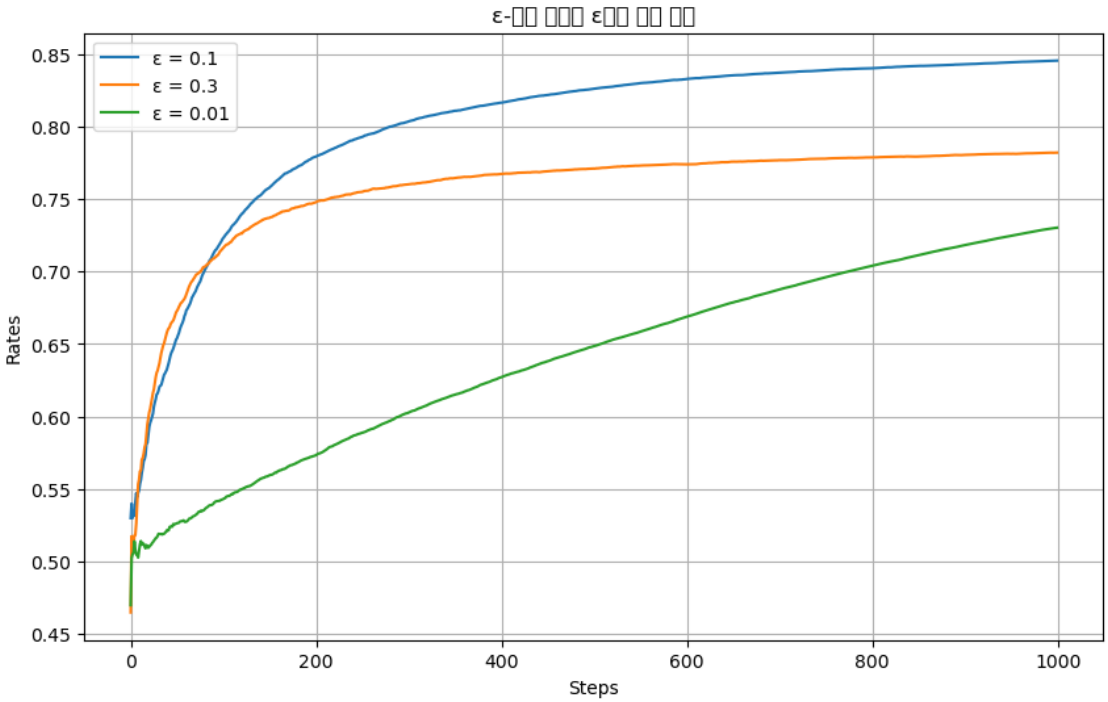
Untitled

March 18, 2025

[5]:
```python
import numpy as np
import matplotlib.pyplot as plt


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

    def play(self, arm):
        rate = self.rates[arm]
        self.rates += 0.01 * (np.random.rand(self.arms) - 0.5)  #
        return 1 if rate > np.random.rand() else 0


class Agent:
    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):
        self.ns[action] += 1
        self.Qs[action] += (reward - self.Qs[action]) / self.ns[action]

    def get_action(self):
        if np.random.rand() < self.epsilon:
            return np.random.randint(0, len(self.Qs))
        return np.argmax(self.Qs)


def simulate(epsilon, runs=200, steps=1000, arms=10):
    all_rates = np.zeros((runs, steps))
    for run in range(runs):
        bandit = NonStatBandit(arms)
        agent = Agent(epsilon, arms)
        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

    return np.mean(all_rates, axis=0)  #


if __name__ == '__main__':
    steps = 1000
    epsilons = [0.1, 0.3, 0.01]  #
    results = {}

    for epsilon in epsilons:
        results[epsilon] = simulate(epsilon)

    plt.figure(figsize=(10, 6))
    for epsilon, rates in results.items():
        plt.plot(rates, label=f'  = {epsilon}')
    plt.xlabel('Steps')
    plt.ylabel('Rates')
    plt.title(' -           ')
    plt.legend()
    plt.grid()
    plt.show()
```


ε-□□ □□□ ε□□ □□ □□
ε = 0.1
ε = 0.3
ε = 0.01
Rates
Steps
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
200
400
600
800
1000

[ ]:

[ ]: