```
# -*- coding: utf-8 -*-
"""Machine Learning

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1_6PjyzZE3CETDIPXGxiWpMtlICgnz_qm
"""

#Vijeth Balasubramaniam - 201664207
#Manoj Nagaraja - 201667140
#Pooja Sriramaneni - 201671298
#Pavithra Shankar - 201633650
import numpy as np
import matplotlib.pyplot as plt

# Define the bandit class
class Bandit:
    def __init__(self, k_arm=10, epsilon=0):
        self.k = k_arm
        self.mean_reward = 0
        self.epsilon = epsilon
        self.q_true = np.random.randn(self.k)
        self.q_est = np.zeros(self.k)
        self.action_count = np.zeros(self.k)

    def act(self):
        # Epsilon-greedy action selection
        if np.random.random() < self.epsilon:
            action = np.random.choice(self.k)
        else:
            max_q = np.max(self.q_est)
            action = np.random.choice([a for a, q in enumerate(self.q_est) if q ==
max_q])
        reward = np.random.normal(self.q_true[action], 1)
        self.action_count[action] += 1
        self.mean_reward += (reward - self.mean_reward) / np.sum(self.action_count)
        self.q_est[action] += (reward - self.q_est[action]) /
self.action_count[action]
        return reward


# Define the experiment function
def experiment(runs=1500, time_steps=1000, bandits=[10], epsilon_values=[0]):
    rewards = np.zeros((len(epsilon_values), len(bandits), runs, time_steps))
    optimal_actions = np.zeros((len(epsilon_values), len(bandits), runs,
time_steps))
    for i, eps in enumerate(epsilon_values):
        for j, k in enumerate(bandits):
            for r in range(runs):
                bandit = Bandit(k, eps)
                for t in range(time_steps):
                    reward = bandit.act()
                    rewards[i, j, r, t] = reward
                    optimal_actions[i, j, r, t] = 1 if bandit.q_est.argmax() ==
bandit.q_true.argmax() else 0
    mean_rewards = np.mean(rewards, axis=2)
    optimal_action_perc = 100 * np.mean(optimal_actions, axis=2)
    return mean_rewards, optimal_action_perc
```

```
# Define the parameters
runs = 1500
time_steps = 1000
bandits = [10]
epsilon_values = [0, 0.01, 0.1]

# Run the experiment
mean_rewards, optimal_action_perc = experiment(runs, time_steps, bandits,
epsilon_values)

# Plot the rests
plt.figure(figsize=(8, 4))
plt.subplot(2, 1, 1)
for i, eps in enumerate(epsilon_values):
    plt.plot(mean_rewards[i][0], label='epsilon = {}'.format(eps))
plt.xlabel('Steps')
plt.ylabel('Average reward')
plt.legend()

plt.subplot(2, 1, 2)
for i, eps in enumerate(epsilon_values):
    plt.plot(optimal_action_perc[i][0], label='epsilon = {}'.format(eps))
plt.xlabel('Steps')
plt.ylabel('% Optimal action')
plt.ylim([0, 100])
plt.legend()

plt.show()
```