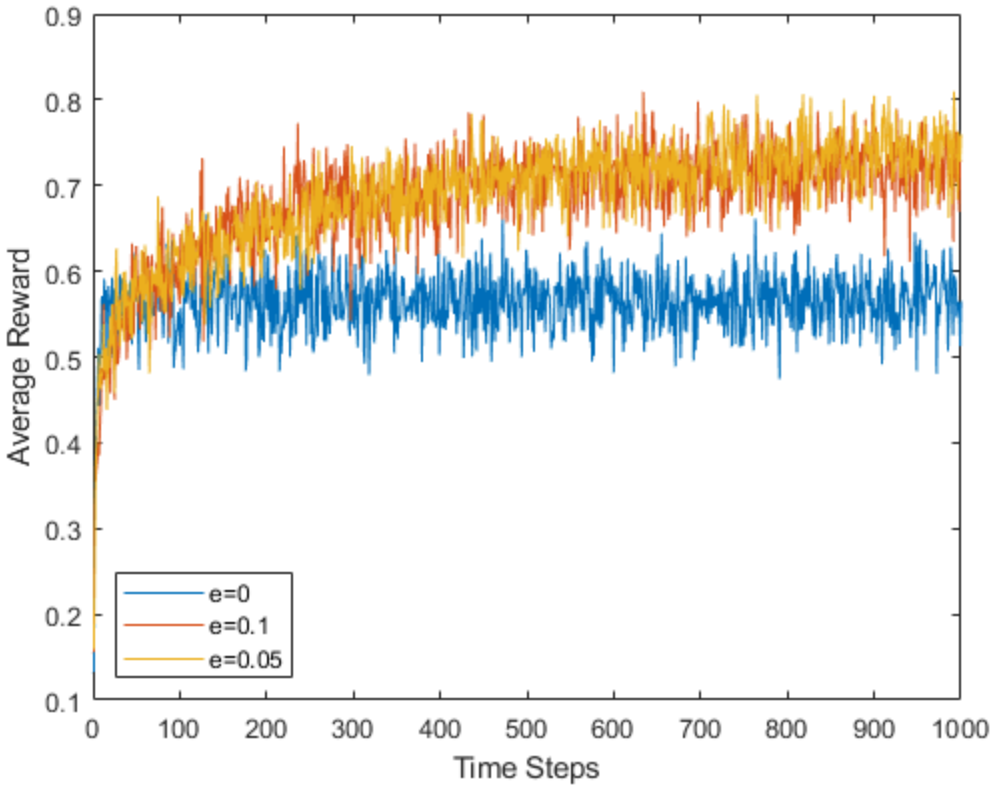
```
% Achyuth Nandikotkur
% Multiarm bandit problem
%
% a. Greedy algorithm;
% b. #-greedy algorithm with #=0.1;
% c. #-greedy algorithm with #=0.05.
%
% Goal: Average Reward vs Steps
clc;
close all;

% No. of runs = 1000
% Time steps per run: 1000
Runs = 1000;
Steps = 1000;

e = [0, 0.1, 0.05]

% Creating a cell array that holds rewards from differernt runs, for
 each epsilon value.
rewardsOf = cell(1,3);

% Creating a cell array that holds average rewards obtained at each
 step by
% taking different actions in every run.
averageRewardOfEpsilon = cell(1,3);

averageRewardOfEpsilon{1} = zeros(1,Steps);
averageRewardOfEpsilon{2} = zeros(1,Steps);
averageRewardOfEpsilon{3} = zeros(1,Steps);

rewardsOf{1} = zeros(Runs, Steps);
rewardsOf{2} = zeros(Runs, Steps);
rewardsOf{3} = zeros(Runs, Steps);

% average rewards of each action
avgRewards = [0.1, -0.7, 0.8, 0.3, 0.5]

% Iterator for runs

for epsilonIndex = 1:3
    for run = 1:Runs
        % Initialized Q as a matrix 5xSteps. Each column represent
 action values of 5
        % actions at every step until N steps.
        % Action Value Matrix with initial value estimates set to 0
        Q = zeros(5, Steps);

        % Initialized R as a matrix 5xSteps. Each column represent
 rewards of 5
        % actions at every step until N steps.
        % Rewards Matrix
```

```
        R = zeros(5,Steps);

        k = zeros(5,1); %how many times have we chosen each action

        for step = 1:Steps
            % epislon greedy algorithm.
            if rand <= (1 - e(epsilonIndex))
                % In Greedy Algorithm, we select an action which has
 the maximum value of expected reward at
                % every step
                [maxValuedActions, I] = max(Q(:, step));

                % Tie breaking between different same max valued
 actions
                sameValueActions = find(Q(:,(step)) ==
 maxValuedActions);
                r = randi(length(sameValueActions));
                selectedAction = sameValueActions(r);
            else
                selectedAction = randi(5);
            end

            % Since we have selected an action, we are incrementing
            % k value of selected action by 1.
            k(selectedAction) = k(selectedAction) + 1;

            % Rewards follow a normal distribution with variance 1
 hence
            % I am using normrnd method to obtain the same given
 variance is 1.
            rewardtemp = normrnd(avgRewards(selectedAction), 1);

            % Here we are calculating an action's value estimate
            % based on the action's previous value estimates.
            % qa(n+1) = qa(n) + alpha * (new-reward - qa(n))
            for action = 1:5
                if(action == selectedAction)
                    Q(action, step+1) = Q(action, step) + (1/
k(selectedAction)) * (rewardtemp - Q(action, step));
                else
                    Q(action, step+1) = Q(action, step);
                end
            end

            % Rewards matrix
            R(selectedAction, step)= rewardtemp;
        end

    % sum(R) gives a vector in which each element represents the sum
 of total
    % rewards from various actions at each step.
    rewardsOf{epsilonIndex}(run,:)=sum(R);
    end
```

```
    % Calculate the mean of each column. Column represents reward
 values for each
    % step over different runs.
    for t=1:Steps
        averageRewardOfEpsilon{epsilonIndex}(t)=
 mean(rewardsOf{epsilonIndex}(:,t));
    end
end


t=1:Steps;
plot(t,averageRewardOfEpsilon{1}, t, averageRewardOfEpsilon{2}, t,
 averageRewardOfEpsilon{3})
xlabel('Time Steps')
ylabel('Average Reward')
legend({'e=0','e=0.1', 'e=0.05'},'Location','southwest')


e =

         0    0.1000    0.0500


avgRewards =

    0.1000   -0.7000    0.8000    0.3000    0.5000
```



Published with MATLAB® R2021a