

```
function [] = reinforcement_comparison_methods(nB,nA,nP,sigmaReward)
%
% Demonstrate the technique of reinforcement comparison
% We compare three methods
%
% M1) action-value method with constant update (alpha = 0.1) and epsilon=0.1 greedy
% M2) action-value method with 1/k update and epsilon=0.1 greedy
% M3) the reinforment comparison method
%
% Inputs:
%   nB: the number of bandits
%   nA: the number of arms
%   nP: the number of plays (times we will pull a arm)
%   sigmaReward: the standard deviation of the return from each of the arms
%
% Written by:
% --
% John L. Weatherwax                2007-11-13
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

%close all;
%clc;
%clear;

if( nargin<1 ) % the number of bandits:
    nB = 2000;
end
if( nargin<2 ) % the number of arms:
    nA = 10;
end
if( nargin<3 ) % the number of plays (times we will pull a arm):
    nP = 2000;
end
if( nargin<4 ) % the standard deviation of the return from each of the arms:
    sigmaReward = 1.0;
end

%randn('seed',0);

%-----
% Compare two methods:
% M1) An epsion greedy (eps=0.1) algorithm with constant update (alpha=0.1)
% M2) An epsion greedy (eps=0.1) algorithm with 1/k action value update
% M3) the reinforment comparison method
%-----
tEps = 0.1;
alpha = 0.1;
beta = 0.1;

avgReward = zeros(2,nP);
perOptAction = zeros(2,nP);

allRewards_M1 = zeros(nB,nP);
```

```

pickedMaxAction_M1 = zeros(nB,nP);
allRewards_M2      = zeros(nB,nP);
pickedMaxAction_M2 = zeros(nB,nP);
for bi=1:nB, % pick a bandit
    % generate the TRUE reward  $Q^{\star}$ :
    qStarMeans = randn(1,nA);
    qT_M1 = zeros(1,nA); % <- initialize action value estimates
    qT_M2 = zeros(1,nA);
    qN_M2 = zeros(1,nA); % <- initialize number of draws on this arm
    % reinforcment comparison method:
    %
    pT = zeros(1,nA); % <- initialize play preference
    rT = 0;           % <- initialize a reference reward (one for all rewards recieved)

    for pi=1:nP, % make a play ... one for each algorithm

        % EXPLORITORY v.s. GREEDY MOVES (first two methods):
        if( rand(1) <= tEps ) % pick a RANDOM arm ... explore:
            [dum,arm_M1] = histc(rand(1),linspace(0,1+eps,nA+1));
            arm_M2 = arm_M1;
        else % pick the GREEDY arm:
            [dum,arm_M1] = max( qT_M1 );
            [dum,arm_M2] = max( qT_M2 );
        end; clear dum;
        % EXPLORITORY v.s. GREEDY MOVES (reinforcment comparison):
        piT = exp(pT) ./ sum( exp(pT) );
        % draw an arm from the distribution piT
        arm_M3 = sample_discrete( piT, 1, 1 );

        % determine if the arm selected is the best possible:
        [dum,bestArm] = max( qStarMeans );
        if( arm_M1==bestArm ) pickedMaxAction_M1(bi,pi) = 1; end
        if( arm_M2==bestArm ) pickedMaxAction_M2(bi,pi) = 1; end
        if( arm_M3==bestArm ) pickedMaxAction_M3(bi,pi) = 1; end
        % get the reward from drawing on that arm:
        reward_M1 = qStarMeans(arm_M1) + sigmaReward*randn(1);
        reward_M2 = qStarMeans(arm_M2) + sigmaReward*randn(1);
        reward_M3 = qStarMeans(arm_M3) + sigmaReward*randn(1);
        allRewards_M1(bi,pi) = reward_M1;
        allRewards_M2(bi,pi) = reward_M2;
        allRewards_M3(bi,pi) = reward_M3;
        % update qT:
        qT_M1(arm_M1) = qT_M1(arm_M1) + alpha*( reward_M1-qT_M1(arm_M1) );
        qT_M2(arm_M2) = qT_M2(arm_M2) + ( reward_M2-qT_M2(arm_M2) )/(qN_M2(arm_M2)+1);
        qN_M2(arm_M2) = qN_M2(arm_M2) + 1;
        % The reinforcment comparison update (play preference pT and average reward)
        pT(arm_M3) = pT(arm_M3) + beta * ( reward_M3 - rT );
        rT          = rT + alpha*( reward_M3 - rT );
    end
end

avgRew      = mean(allRewards_M1,1);
avgReward(1,:) = avgRew(:).';
avgRew      = mean(allRewards_M2,1);
avgReward(2,:) = avgRew(:).';
avgRew      = mean(allRewards_M3,1);
avgReward(3,:) = avgRew(:).';

percentOptAction = mean(pickedMaxAction_M1,1);
perOptAction(1,:) = percentOptAction(:).';

```

```
percentOptAction = mean(pickedMaxAction_M2,1);
perOptAction(2,:) = percentOptAction(:).';
percentOptAction = mean(pickedMaxAction_M3,1);
perOptAction(3,:) = percentOptAction(:).';

% produce the average rewards plot:
%
figure; hold on;
all_hnds = plot( 1:nP, avgReward );
legend( all_hnds, { 'alpha=0.1', 'alpha=1/k', 'rein\_comp' }, 'Location', 'SouthEast' );
axis tight; grid on;
xlabel( 'plays' ); ylabel( 'Average Reward' );

% produce the percent optimal action plot:
%
figure; hold on;
all_hnds = plot( 1:nP, perOptAction );
legend( all_hnds, { 'alpha=0.1', 'alpha=1/k', 'rein\_comp' }, 'Location', 'SouthEast' );
axis( [ 0, nP, 0, 1 ] ); axis tight; grid on;
xlabel( 'plays' ); ylabel( '% Optimal Action' );
```