Group: Wang, Achoukhi, Ching

Exercises – Third week

1. ε-greedy method on the 10-armed bandit problem

Task:

Implement the ε -greedy algorithm for solving a 10-armed bandit problem with the following setup:

- $n = 10$ possible actions
- Each $Q^*(a)$ is chosen randomly from a normal distribution: $\eta(0; 1)$
- Each r_t is also normal: $\eta(Q^*(a_t); 1)$
- 1000 plays
- Repeat the whole thing 2000 times and average the results

Run experiments with ε = 0:1; ε = 0:01 and ε = 0:009. Finally, plot the average curves for each value of ε.

Implementation:

```
clc
clear all
close all
action = zeros(10,1);
%rt = zeros(1000,1);
Q = [1:10];
Qstar = [1:10];
count = [1:10];
rmean = [1:1001];
rmean(1) = 0;

for i = 1:1:10
     Qstar(i) = normrnd(0,1);
end
epsvec = [1:3];
epsvec(1) = 0.1;
epsvec(2) = 0.01;
epsvec(3) = 0.001;

rmeanmat = zeros(3,1001);

for k=1:1:3
    eps = epsvec(k);
    for i=1:1:1000
        rmean(i+1) = 0;
    end
for m = 1:1:2000
    for n = 1:1:10
        Q(n) = 0;
        count(n) = 0;
    end

    for j = 1:1:1000;

        prob = rand();
        if prob < (1-eps)
            [~,a] = max(Q);
        else
            a = randi([1,10],1,1);
        end
        count(a) = count(a) + 1;
        r = normrnd(Qstar(a),1);
        rmean(j+1) = rmean(j+1) + r;
        Q(a) = Q(a)+ 1/count(a)*(r-Q(a));

    end
```

```
end
for i=1:1:1001
    rmeanmat(k,i) = rmean(i)/2000;
end
end

x=[0:1000];
figure
plot(x,rmeanmat(1,:),'c',x,rmeanmat(2,:),'b',x,rmeanmat(3,:),'r');
```

Results:

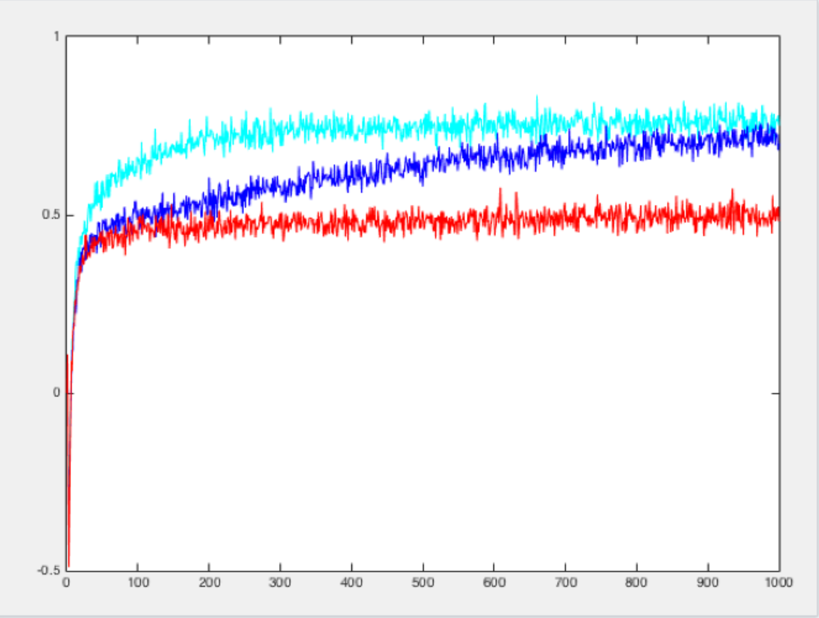

Figure 1: Average depicts curves for ε = 0:1; ε = 0:01 and ε = 0:009.

2. Iterative Policy Evaluation

Tasks:

1) Implement the Iterative Policy Evaluation.
2) Perform experiments with the problem of A Small Gridworld depicted in page 5 of the presentation on dynamic programming.
3) In the report for this exercise you should include the Final Value Function.

Implementation:

```
function [V] = iter_poly_eval()

gamma = 1;
sideL  = 4;
nGrids = sideL^2;
V = zeros(sideL);
MAX_N_ITERS = 100;  iterCnt = 0;
CONV_TOL    = 1e-6; delta = 1e10;
pol_pi = 0.25;

while((delta > CONV_TOL) && (iterCnt <= MAX_N_ITERS))
  delta = 0;

  for ii=1:sideL,
    for jj=1:sideL,
      if((ii==1 && jj==1) || (ii==sideL && jj==sideL)) continue; end
```

```
      v     = V(ii,jj);
      v_tmp = 0.0;

      % action = UP
      if(ii==1)
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii,jj));
      elseif(ii==2 && jj==1)
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii-1,jj));
      else
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii-1,jj));
      end

      % action = DOWN
      if( ii==sideL )
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii,jj));
      elseif( ii==sideL-1 && jj==sideL )
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii+1,jj));
      else
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii+1,jj));
      end

      % action = RIGHT
      if( jj==sideL )
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii,jj));
      elseif( jj==sideL-1 && ii==sideL )
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii,jj+1));
      else
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii,jj+1));
      end

      % action = LEFT
      if( jj==1 )                          %
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii,jj));
      elseif( jj==2 && ii==1 )
        v_tmp = v_tmp + pol_pi*(-1 + gamma*V(ii,jj-1));
      else                                 %
        v_tmp = v_tmp + pol_pi*( -1 + gamma*V(ii,jj-1) );
      end

      % update V(ii,jj):
      V(ii,jj) = v_tmp;

      delta = max([delta, abs(v-V(ii,jj))]);
    end
  end

  iterCnt=iterCnt+1;
  if(0 && mod(iterCnt,1)==0)
    fprintf('iterCnt=%5d; delta=%10.5f\n', iterCnt, delta);
    disp(round(V*10)/10);
    pause
  end
end
```

Result:

Final Value Function

```
    0.00000  -13.99785  -19.99692  -21.99661
  -13.99785  -17.99737  -19.99714  -19.99717
  -19.99692  -19.99714  -17.99759  -13.99820
  -21.99661  -19.99717  -13.99820    0.00000
```