Aggressive, Soft-Robust and Robust Policy for MDP via offline Q-learning

Code submitted for group project, DA671, 2023, IITG

Group Members: Vikky Masih, Omendra Gangwar,Prakhar Kumar Sonkar, Karnati Saipriya, Sushant Suresh Pargaonkar.

Reference paper for MDP model and Soft-Robust Policy: Derman, Esther, Daniel J. Mankowitz, Timothy A. Mann, and Shie Mannor. "Soft-robust actor-critic policy-gradient." arXiv preprint arXiv:1803.04848 (2018).


```
clc
clear all
close all
format longG
rng('default')
```

Model information

One step MDP with 7 states and 3 actions.

It has fixed starting state. (S1)

All other states are terminal states {S2,...,S7}.

Rewards are deterministic. (R(s,a,s')->constant)

Transition matrix is dependent on parameter p. (look at getMDP(p))

p belongs to a given uncertainty set.

The uncertainty set has a constant probability distribution.

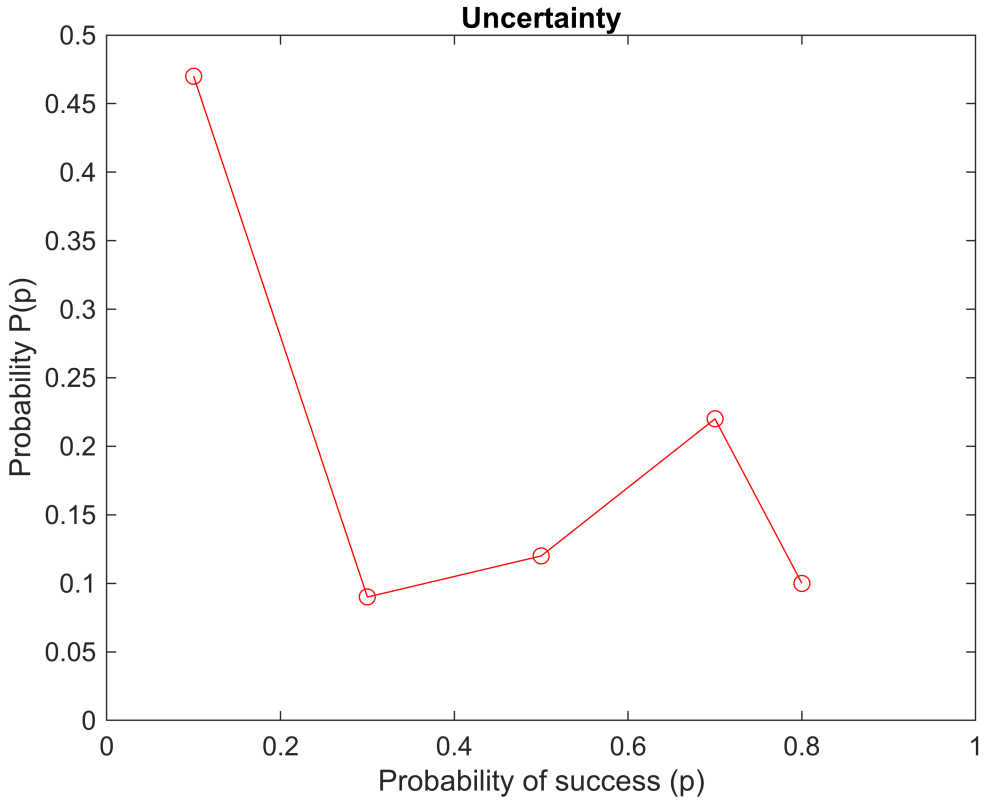
A nominal model is defined by a given nominal p.

```
Nominal_p=0.8;
Uncertainty_set=[0.1,0.7,0.8,0.3,0.5];
Weighting_distr=[0.47,0.22,0.1,0.09,0.12];

numEpisodesTrain=50000;
numEpisodesTest=10000;
epsilonGreedy=@(t) 7/(t^0.7);
stepSizeQ=0.01;
discountFactor=1;
```

Visualizing uncertainty

```
[~,I]=sort(Uncertainty_set);
f=figure;
plot(Uncertainty_set(I),Weighting_distr(I),'-or');
xlabel("Probability of success (p)")
```

```
ylabel("Probability P(p)")
title("Uncertainty")
xlim([0,1]);
ylim([0,0.5]);
exportgraphics(f,"Uncertainty.png","Resolution",300);
```



Displaying success probabilites for different modes

```
Expected_p=Weighting_distr*Uncertainty_set';
if true
    fprintf("p_aggressive = %g\n",max(Uncertainty_set));
    fprintf("p_soft_robust = E(p) = %g\n",Expected_p);
    fprintf("p_robust = %g\n",min(Uncertainty_set));
    disp("-------------------------------------------")
end
```

```
p_aggressive = 0.8
p_soft_robust = E(p) = 0.368
p_robust = 0.1
-------------------------------------------
```

Offline Q-learning for different modes

```
modes=["Aggressive","Soft-Robust","Robust"];
numModes=length(modes);
[P_nom,R_nom] = getMDP(Nominal_p);
[P_bar,~] = getMDP(Expected_p);
```

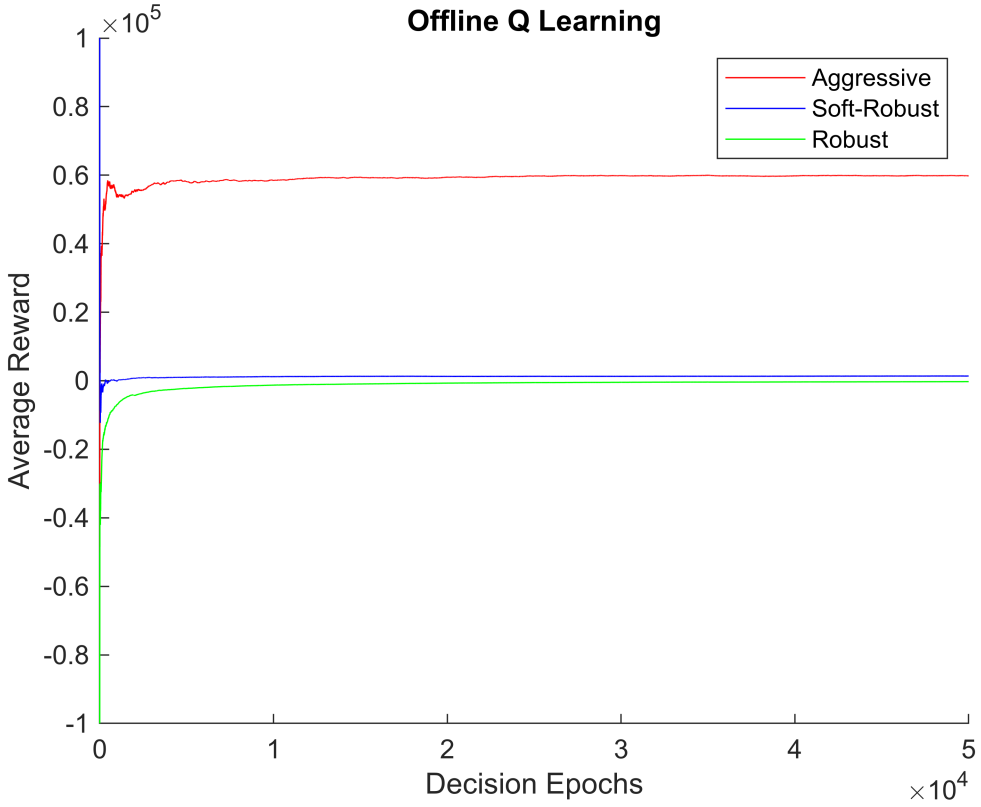
```
[P_robust,~]= getMDP(min(Uncertainty_set));
[P_agg,~] = getMDP(max(Uncertainty_set));
[numStates,numActions,~]=size(P_nom);
terminalStates=[2,3,4,5,6,7];

recordSAR_train=zeros(numModes,numEpisodesTrain,3);
recordQ=zeros(numModes,numStates,numActions);
for mode=modes
    Q=zeros(numStates,numActions);

    i=(modes==mode);
    if mode=="Aggressive"
        P=P_agg;
    elseif mode=="Robust"
        P=P_robust;
    else
        P=P_bar;
    end

    for episode=1:numEpisodesTrain
        while true
            % Initialize State
            S=1;

            % Epsilon greedy action selection
            if sum(Q(S,:).^2)==0
                A=mod((episode-1),numActions)+1;
            else
                [~,A]=max(Q(S,:));
                if rand<=epsilonGreedy(episode)
                    e=1:numActions;
                    e=e(e~=A);
                    A=e(randi(numActions-1));
                end
            end

            % Reward and Next Action
            S_next=discreteSamples(reshape(P(S,A,:),1,[]),1);
            R=R_nom(S,A,S_next);
            recordSAR_train(i,episode,:)=[S,A,R];

            % Q-learning update
            Q(S,A)=Q(S,A)+ ...
                   stepSizeQ*(R+discountFactor*max(Q(S_next,:))-Q(S,A));
            S=S_next;
            if any(S==terminalStates)
                break
            end
        end
    end
```

```
    disp("Q values for "+mode+" Policy:")
    disp(Q)
    disp("-------------------------------------------")
    recordQ(i,:,:)=Q;
end
```

```
Q values for Aggressive Policy:
          59225.5015846494          3652.54234958587          1553.15916220777
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
-------------------------------------------
Q values for Soft-Robust Policy:
         -22525.7216448199          1524.0536393624          649.737206032675
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
-------------------------------------------
Q values for Robust Policy:
         -75563.3036330202          68.0989257218792          133.807700092128
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
                         0                         0                         0
-------------------------------------------
```

Offline Q-learning results

```
f=figure;
hold on
z=((1:length(recordSAR_train))');
plot(cumsum(reshape(recordSAR_train(1,:,3),[],1))./z,'-r')
plot(cumsum(reshape(recordSAR_train(2,:,3),[],1))./z,'-b')
plot(cumsum(reshape(recordSAR_train(3,:,3),[],1))./z,'-g')
legend(modes)
ylabel("Average Reward")
xlabel("Decision Epochs")
title("Offline Q Learning")
hold off
exportgraphics(f,"Offline-Q-Learning.png","Resolution",300);
```



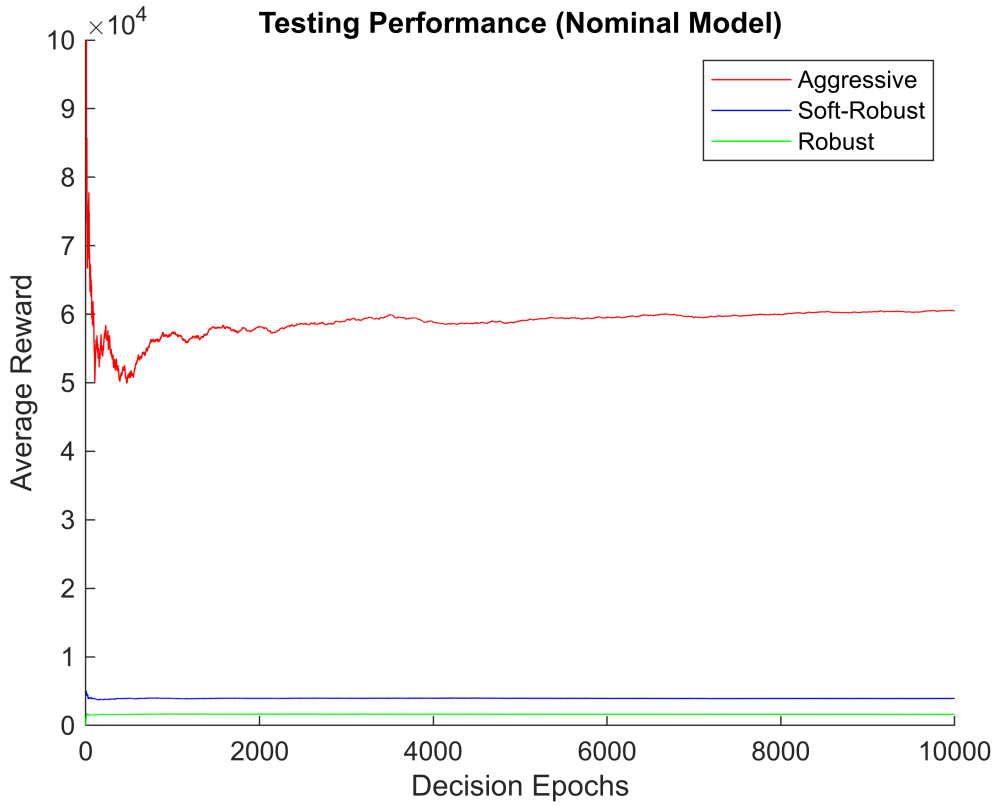
Policy evaluation on nominal model

```
P=P_nom;
recordSAR_test=zeros(numModes,numEpisodesTest,3);
for mode=modes
    i=(modes==mode);
    for episode=1:numEpisodesTest
        while true
            % Initialize State
            S=1;

            % Action Selection
            [~,A]=max(recordQ(i,S,:));

            % Reward and Next Action
            S_next=discreteSamples(reshape(P(S,A,:),1,[]),1);
            R=R_nom(S,A,S_next);
            recordSAR_test(i,episode,:)=[S,A,R];

            S=S_next;
            if any(S==terminalStates)
                break
            end
        end
    end
```

```
end
```

Policy evaluations results for nominal model

```
f=figure;
hold on
z=((1:length(recordSAR_test))');
plot(cumsum(reshape(recordSAR_test(1,1:end,3),[],1))./z,'-r')
plot(cumsum(reshape(recordSAR_test(2,1:end,3),[],1))./z,'-b')
plot(cumsum(reshape(recordSAR_test(3,1:end,3),[],1))./z,'-g')
legend(modes)
ylabel("Average Reward")
xlabel("Decision Epochs")
title("Testing Performance (Nominal Model)")
hold off
exportgraphics(f,"Testing_Performance_NominalModel.png","Resolution",300);
```



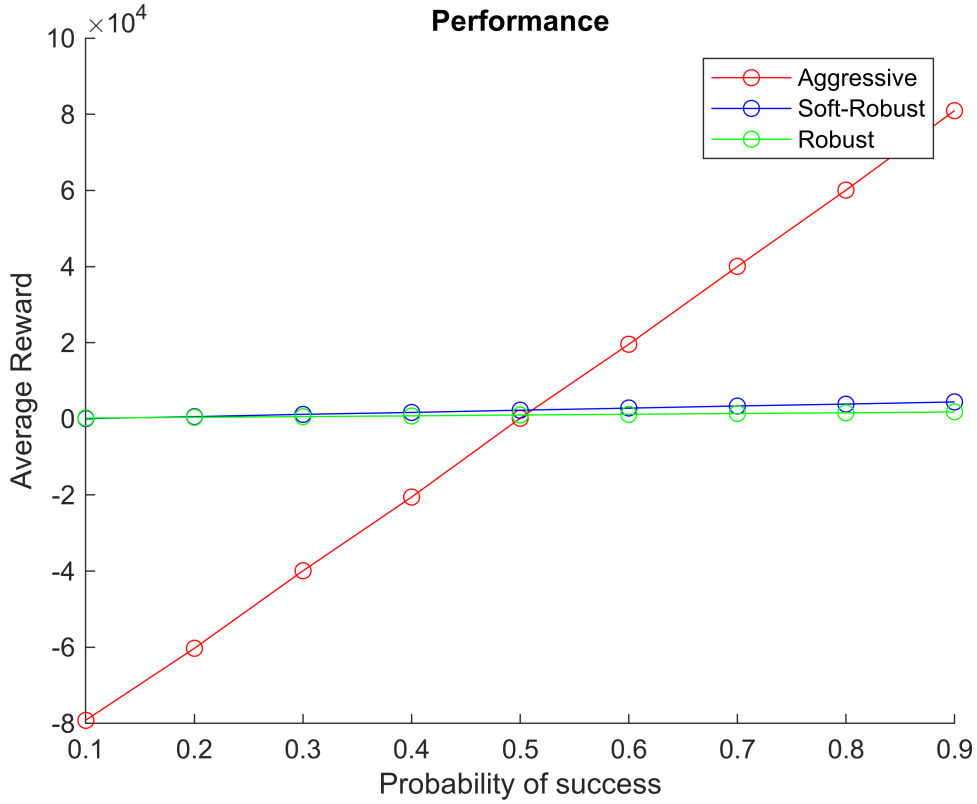
Policy performance vs probability of success

```
p_values=linspace(0.1,0.9,9);
perf_all=zeros(numModes,length(p_values));
for p=p_values
    [P,~] = getMDP(p);
    for mode=modes
        i=(modes==mode);
        cum_R=0;
```

```
        for episode=1:numEpisodesTest
            while true
                % Initialize State
                S=1;

                % Action Selection
                [~,A]=max(recordQ(i,S,:));

                % Reward and Next Action
                S_next=discreteSamples(reshape(P(S,A,:),1,[]),1);
                R=R_nom(S,A,S_next);
                cum_R=cum_R+R;
                S=S_next;
                if any(S==terminalStates)
                    break
                end
            end
        end
        perf_all(i,p==p_values)=cum_R/numEpisodesTest;
    end
end
```

Results for Policy performance vs probability of success

```
f=figure;
hold on
plot(p_values,perf_all(1,:),'-or')
plot(p_values,perf_all(2,:),'-ob')
plot(p_values,perf_all(3,:),'-og')
legend(modes)
ylabel("Average Reward")
xlabel("Probability of success")
title("Performance")
hold off
exportgraphics(f,"Performance.png","Resolution",300);
```



Discrete Sampling (Roulette wheel)

```
function x=discreteSamples(p,n)
    edges=[0 cumsum(p)];
    le=edges(1:end-1);
    re=edges(2:end);
    rv=rand(n,1)*edges(end);
    [~,x]=max(((rv>=le).*(rv<re)),[],2);
end
```

MDP Specification T(s,a,s'), R(s,a,s')

One step MDP with 7 states and 3 actions.

Rewards are deterministic. (R(s,a,s')->constant)

Transition matrix is dependent on parameter p.

```
function [T,R]=getMDP(p)
    T=zeros(7,3,7);
    R=zeros(7,3,7);
    T(1,1,2)=p;
    R(1,1,2)=10^5;
    T(1,1,3)=1-p;
    R(1,1,3)=-10^5;
    T(1,2,4)=p;
```

```
    R(1,2,4)=5000;
    T(1,2,5)=1-p;
    R(1,2,5)=-500;
    T(1,3,6)=p;
    R(1,3,6)=2000;
    T(1,3,7)=1-p;
    R(1,3,7)=0;
    T(2:7,:,1)=1;
end
```