```
function [Q_sarsa,Q_qlearn,rpt_sarsa,rpt_qlearn,n_sarsa,n_qlearn] =
learn_cw(alpha,CF,s_start,s_end,MAX_N_EPISODES)
% LEARN_CW - Performs on-policy sarsa and Q-learning to learn the policy for the
% cliff walking problem example.
%
% Written by:
% --
% John L. Weatherwax                2007-12-03
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

PLOT_STEPS = 0;

gamma = 1;    % <- take this is an undiscounted task

epsilon = 0.1;  % for our epsilon greedy policy

% the number of states:
[sideII,sideJJ] = size(CF);
nStates         = sideII*sideJJ;

% on each grid we can choose from among this many actions (except on edges where this
action is reduced):
nActions = 4;

% An array to hold the values of the action-value function
Q_sarsa    = zeros(nStates,nActions);
Q_qlearn   = zeros(nStates,nActions);
rpt_sarsa  = zeros(1,MAX_N_EPISODES);
rpt_qlearn = zeros(1,MAX_N_EPISODES);
n_sarsa    = zeros(nStates,nActions); % <- lets store the number of times we are in this
state and take this action
n_qlearn   = zeros(nStates,nActions);


% keep track of how many timestep we take per episode and for method (episode time
steps):
ets = zeros(MAX_N_EPISODES,2);
for ei=1:MAX_N_EPISODES,
  if( PLOT_STEPS )
    close all;
    f_plot_steps=figure; subplot(2,1,1); imagesc( CF ); colorbar; hold on;
    plot( s_start(2), s_start(1), 'x', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
    plot( s_end(2), s_end(1), 'o', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
    subplot(2,1,2); imagesc( CF ); colorbar; hold on;
    plot( s_start(2), s_start(1), 'x', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
    plot( s_end(2), s_end(1), 'o', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
  end
  tic;
%   if( ei==1 )
%     fprintf('working on episode %d...\n',ei);
%   else
%     fprintf('working on episode %d (ptt=%10.6f secs)...\n',ei, toc); tic;
%   end
```

```
  %set the control variables of sarsa finished and q-learning finished
  sarsa_finished=0;
  qlearning_finished=0;
  % initialize the starting state
  st_sarsa  = s_start; sti_sarsa  = sub2ind( [sideII,sideJJ], st_sarsa(1),  st_sarsa(2)
);
  st_qlearn = s_start; sti_qlearn = sub2ind( [sideII,sideJJ], st_qlearn(1), st_qlearn(2)
);

  % pick an initial action using an epsilon greedy policy derived from Q:
  %
  [dum,at_sarsa] = max(Q_sarsa(sti_sarsa,:));  % at \in [1,2,3,4]=[up,down,right,left]
  if( rand<epsilon )         % explore ... with a random action
    tmp=randperm(nActions); at_sarsa=tmp(1);
  end
  [dum,at_qlearn] = max(Q_qlearn(sti_qlearn,:));  % at \in [1,2,3,4]=[up,down,right,left]
  if( rand<epsilon )         % explore ... with a random action
    tmp=randperm(nActions); at_qlearn=tmp(1);
  end

  % begin an episode
  R_sarsa=0; R_qlearn=0;
  while( 1 )

    %fprintf('st=(%d,%d); act=%3d...\n',st(1),st(2),at);
    % propagate to state stp1 and collect a reward rew
    [rew_sarsa, stp1_sarsa]  = stNac2stp1(st_sarsa, at_sarsa,CF, s_start,s_end,1);
R_sarsa=R_sarsa+rew_sarsa;
    [rew_qlearn,stp1_qlearn] = stNac2stp1(st_qlearn,at_qlearn,CF,s_start,s_end,2);
R_qlearn=R_qlearn+rew_qlearn;

    %fprintf('stp1=(%d,%d); rew=%3d...\n',stp1(1),stp1(2),rew);
    % pick the greedy action from state stp1:
    stp1i_sarsa = sub2ind( [sideII,sideJJ], stp1_sarsa(1), stp1_sarsa(2) );
    stp1i_qlearn = sub2ind( [sideII,sideJJ], stp1_qlearn(1), stp1_qlearn(2) );

    % SARSA:
    %
    if (~sarsa_finished)
      ets(ei,1)=ets(ei,1)+1;

      % --make the greedy action selection:
      [dum,atp1_sarsa] = max(Q_sarsa(stp1i_sarsa,:));
      if( rand<epsilon )         % explore ... with a random action
        tmp=randperm(nActions); atp1_sarsa=tmp(1);
      end
      n_sarsa(sti_sarsa,at_sarsa) = n_sarsa(sti_sarsa,at_sarsa)+1;
      if( ~( (stp1_sarsa(1)==s_end(1)) && (stp1_sarsa(2)==s_end(2)) ) ) % stp1 is not the
terminal state
        Q_sarsa(sti_sarsa,at_sarsa) = Q_sarsa(sti_sarsa,at_sarsa) + alpha*( rew_sarsa +
gamma*Q_sarsa(stp1i_sarsa,atp1_sarsa) - Q_sarsa(sti_sarsa,at_sarsa) );
      else                                                % stp1 IS the terminal state
... no Q_sarsa(s';a') term in the sarsa update
        Q_sarsa(sti_sarsa,at_sarsa) = Q_sarsa(sti_sarsa,at_sarsa) + alpha*( rew_sarsa -
Q_sarsa(sti_sarsa,at_sarsa) );
        if (PLOT_STEPS);
          fprintf('Sarsa: On episode %d i have reached the objective in %d
steps\n',ei,ets(ei,1));
          if (~qlearning_finished)
```

```
          fprintf('Sarsa: now waiting for q-learning to finish\n');
        end
        %pause;
        %close all;
        %subplot(2,1,1); imagesc( CF ); colorbar; hold on;
        %plot( s_start(2), s_start(1), 'x', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
        %plot( s_end(2), s_end(1), 'o', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
      end
      sarsa_finished=1;
    end

    if (PLOT_STEPS)
      subplot(2,1,1);
      num2act={'UP','DOWN','LEFT','RIGHT'};
      plot(st_sarsa(2),st_sarsa(1),'o','MarkerFaceColor','g');
      title(['Sarsa action= ',num2act(atp1_sarsa)] );
      plot(stp1_sarsa(2),stp1_sarsa(1),'o','MarkerFaceColor','k');
      drawnow;
    end
    %update (st,at) pair:
    st_sarsa = stp1_sarsa; sti_sarsa = stp1i_sarsa; at_sarsa = atp1_sarsa;

  end

  % Q-learning:
  %
  if (~qlearning_finished)
    ets(ei,2)=ets(ei,2)+1;

    % --make the greedy action selection:
    n_qlearn(sti_qlearn,at_qlearn) = n_qlearn(sti_qlearn,at_qlearn)+1;
    [dum,atp1_qlearn] = max(Q_qlearn(stp1i_qlearn,:));
    if( rand<epsilon )         % explore ... with a random action
      tmp=randperm(nActions); atp1_qlearn=tmp(1);
    end
    if( ~( (stp1_qlearn(1)==s_end(1)) && (stp1_qlearn(2)==s_end(2)) ) ) % stp1 is not
the terminal state
      Q_qlearn(sti_qlearn,at_qlearn) = Q_qlearn(sti_qlearn,at_qlearn) + alpha*(
rew_qlearn + gamma*max(Q_qlearn(stp1i_qlearn,:)) - Q_qlearn(sti_qlearn,at_qlearn) );
    else                                                  % stp1 IS the terminal state
... no Q_qlearn(s';a') term in the qlearn update
      Q_qlearn(sti_qlearn,at_qlearn) = Q_qlearn(sti_qlearn,at_qlearn) + alpha*(
rew_qlearn - Q_qlearn(sti_qlearn,at_qlearn) );
      if (PLOT_STEPS);
        fprintf('Q-learning: on episode %d i have reached the objective in %d steps
\n',ei,ets(ei,2));
        if (~sarsa_finished)
          fprintf('Q-learning: waiting for sarsa to finish\n');
        end
        %pause;
        %close all;
        %subplot(2,1,2); imagesc( CF ); colorbar; hold on;
        %plot( s_start(2), s_start(1), 'x', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
        %plot( s_end(2), s_end(1), 'o', 'MarkerSize', 10, 'MarkerFaceColor', 'k' );
      end
      qlearning_finished=1;
    end

    if (PLOT_STEPS)
      subplot(2,1,2);
```

```
        num2act={'UP','DOWN','LEFT','RIGHT'};
        plot(st_qlearn(2),st_qlearn(1),'o','MarkerFaceColor','g');
        title(['Q Learning action= ',num2act(atp1_qlearn)] );
        plot(stp1_qlearn(2),stp1_qlearn(1),'o','MarkerFaceColor','k');
        drawnow;
      end
      %update (st,at) pair:
      st_qlearn = stp1_qlearn; sti_qlearn = stp1i_qlearn; at_qlearn = atp1_qlearn;

    end

    if (sarsa_finished && qlearning_finished)
      break;
    end
  end % end policy while
  rpt_sarsa(ei) = R_sarsa; rpt_qlearn(ei) = R_qlearn;

  if (PLOT_STEPS)
    fprintf('On episode %d the rewards were:\n',ei);
    fprintf('sarsa: %d \t Q-learning: %d\n', rpt_sarsa(ei),rpt_qlearn(ei));
  end;

end % end episode loop


function [rew,stp1] = stNac2stp1(st,act,CF,s_start,s_end,caller_id)
% STNAC2STP1 - state and action to state plus one and reward
%

% extract dimensions:
[sideII,sideJJ] = size(CF);

% convert to row/column notation:
ii = st(1); jj = st(2);

% incorporate any actions and fix our position if we end up outside the grid:
%
switch act
 case 1,
  %
  % action = UP
  %
  stp1 = [ii-1,jj];
 case 2,
  %
  % action = DOWN
  %
  stp1 = [ii+1,jj];
 case 3,
  %
  % action = RIGHT
  %
  stp1 = [ii,jj+1];
 case 4
  %
  % action = LEFT
  %
  stp1 = [ii,jj-1];
 otherwise
  error(sprintf('unknown value for of action = %d',act));
```

```
end

% adjust our position of we have fallen outside of the grid:
%
if( stp1(1)<1      ) stp1(1)=1;      end
if( stp1(1)>sideII ) stp1(1)=sideII; end
if( stp1(2)<1      ) stp1(2)=1;      end
if( stp1(2)>sideJJ ) stp1(2)=sideJJ; end

% get the reward for this step:
%
if( (ii==s_end(1)) && (jj==s_end(2)) )  % were at the end :)
  %rew = +1;
  rew = 0;
elseif( CF(stp1(1),stp1(2))==0 )        % we fell off the cliff :(
  rew  = -100;
  stp1 = s_start;
%   if (caller_id==1)
%       fprintf('Sarsa has fallen the clif\n');
%   else
%       fprintf('Q-Learning has fallen the clif\n');
%   end
else                                    % normal step
  rew = -1;
end
```