```
%function [] = binary_bandit_exps(nB,nP,p_win)
%
% Duplicates the binary bandit experiments.
%
% Inputs:
%   nB: the number of bandits
%   nP: the number of plays (times we will pull a arm)
%   p_win: p_win(i) is the probability we win when we pull arm i.
%
% 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 plays (times we will pull a arm):
%   nP = 2000;
% end
% if( nargin<3 )
%   p_win = [ 0.1, 0.2 ];
%   p_win = [ 0.8, 0.9 ];
% end

% the number of arms:
nA = 2;

[dum,bestArm] = max( p_win );

%randn('seed',0);

if( 1 )
  % run the SUPERVISED experiment for two epsilon:
  % 0   => fully greedy
  % 0.1 => inbetween
  % 1.0 => explore on each trial
  epsArray = [ 0, 0.1 ];

  perOptAction = zeros(length(epsArray),nP);
  for ei=1:length(epsArray),
    tEps = epsArray(ei);

    pickedMaxAction = zeros(nB,nP);
    for bi=1:nB, % pick a bandit
      % pick an arm to play initially ...
      %arm = 1;
      [dum,arm] = histc(rand(1),linspace(0,1+eps,nA+1)); clear dum;
      for pi=1:nP, % make a play
```

```
        % determine if this move is exploritory or greedy:
        if( rand(1) <= tEps ) % pick a RANDOM arm:
          [dum,arm] = histc(rand(1),linspace(0,1+eps,nA+1)); clear dum;
        end
        if( arm==1 ) otherArm=2; else otherArm=1; end
        % determine if the arm selected is the best possible:
        if( arm==bestArm ) pickedMaxAction(bi,pi)=1; end
        % get the reward from drawing on that arm:
        prob = p_win(arm);
        if( rand(1) <= prob )               % this arm gave SUCCESS keep playing it ...
          % do nothing ...
        else                                % this arm gave FAILURE switch to the other
...
          arm = otherArm;
        end
      end
    end

    percentOptAction   = mean(pickedMaxAction,1);
    perOptAction(ei,:) = percentOptAction(:).';
  end
end


%-----------------------------------------------------------------------
% Learning with the L_{R-P} (linear reward penalty) algorithm:
%-----------------------------------------------------------------------
alpha = 0.1;
if( 1 )
  perOptActionRP = zeros(1,nP);

  %qT = zeros( nB, nA );  % initialize to zero the probability this arm gives a success
(no knowledge)
  qT = 0.5*ones( nB, nA );  % initialize to uniform the probability this arm gives a
success (no knowledge)

  pickedMaxAction = zeros(nB,nP);
  for bi=1:nB, % pick a bandit
    for pi=1:nP, % make a play
      % pick an arm based on the distribution in qT:
      if( rand(1) < qT(bi,1) )
        arm = 1;
      else
        arm = 2;
      end
      if( arm==1 ) otherArm=2; else otherArm=1; end
      % determine if the arm selected is the best possible:
      if( arm==bestArm ) pickedMaxAction(bi,pi)=1; end
      % get the reward from drawing on that arm:
      prob = p_win(arm);
      if( rand(1) <= prob )               % this arm gave success increment ...
        addTo = arm;
      else                                % this arm gave failure increment the other
...
        addTo = otherArm;
      end
      if( addTo==1 ) otherArm=2; else otherArm=1; end
      % update qT:
      qT(bi,addTo)    = qT(bi,addTo) + alpha*( 1 - qT(bi,addTo) );
      qT(bi,otherArm) = 1.0-qT(bi,addTo);
```

```
    end
  end

  percentOptAction   = mean(pickedMaxAction,1);
  perOptActionRP(1,:) = percentOptAction(:).';
end
perOptAction = [ perOptAction; perOptActionRP ];


%----------------------------------------------------------------------
% Learning with the L_{R-I} (linear reward inaction) algorithm:
%----------------------------------------------------------------------
alpha = 0.1;
if( 1 )
  perOptActionRI = zeros(1,nP);

  qT = zeros( nB, nA );  % initialize to zero the probability this arm gives a success
(no knowledge)
  qT = 0.5*ones( nB, nA );  % initialize to uniform the probability this arm gives a
success (no knowledge)

  pickedMaxAction = zeros(nB,nP);
  for bi=1:nB, % pick a bandit
    for pi=1:nP, % make a play
      % pick an arm based on the distribution in qT:
      if( rand(1) < qT(bi,1) )
        arm = 1;
      else
        arm = 2;
      end
      if( arm==1 ) otherArm=2; else otherArm=1; end
      % determine if the arm selected is the best possible:
      if( arm==bestArm ) pickedMaxAction(bi,pi)=1; end
      % get the reward from drawing on that arm:
      prob = p_win(arm);
      if( ~(rand(1) <= prob) )               % this arm gave failure no learning occurs
...
        continue
      end                                    % this arm gave a success ...
      % I selected arm "arm".  I won when I played this arm therefore I infer this play
to be correct
      addTo = arm;
      if( addTo==1 ) otherArm=2; else otherArm=1; end
      % update qT:
      qT(bi,addTo)    = qT(bi,addTo) + alpha*( 1 - qT(bi,addTo) );
      qT(bi,otherArm) = 1.0-qT(bi,addTo);
    end
  end

  percentOptAction  = mean(pickedMaxAction,1);
  perOptActionRI(1,:) = percentOptAction(:).';
end
perOptAction = [ perOptAction; perOptActionRI ];



% produce the percent optimal action plot:
%
figure; hold on; clrStr = 'brkc'; all_hnds = [];
for ei=1:size(perOptAction,1)
```

```
  %all_hnds(ei) = plot( [ 0, avgReward(ei,:) ], [clrStr(ei)] );
  all_hnds(ei) = plot( 1:nP, perOptAction(ei,:), [clrStr(ei),'-'] );
end
%legend( all_hnds, { '0', '0.1', 'L_{RI}' }, 'Location', 'Best' );
legend( all_hnds, { '0', '0.1', 'L_{RP}', 'L_{RI}' }, 'Location', 'Best' );
axis( [ 0, nP, 0, 1 ] ); axis tight; grid on;
xlabel( 'plays' ); ylabel( '% Optimal Action' );

return;
```