(http://people.revoledu.com/kardi/)

MENU

Q-Learning Using Matlab

by Kardi Teknomo (../../index.html)

(purchase.html)

< Previous (Tower-of-Hanoi.htm) | Next (Q-Learning-Excel.htm) | Contents (index.html) >

Q-Learning using Matlab

I have made simple Matlab Code below for this tutorial example and you can modify it for your need. You can copy and paste the two functions into separate text files and run it as ReinforcementLearning .

To model the environment you need to make the instant reward matrix R . Put zero for any door that is not directly to the goal and put value 100 to the door that lead directly to the goal. For unconnected states, use minus Infinity (-Inf) so that it become very negative number. We want to maximize the Q values, thus very negative number will not be considered at all.

The state is numbered 1 to N (in our previous example N = 6). The result of the code is only normalized Q matrix.

You may experiment in the effect of parameter gamma to see how it influences the results.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Q learning of single agent move in N rooms
% Matlab Code companion of
% Q Learning by Example, by Kardi Teknomo
% (http://people.revoledu.com/kardi/)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function q=ReinforcementLearning
clc;
format short
format compact
```

```
% Two input: R and gamma
% immediate reward matrix;
% row and column = states; -Inf = no door between room
R=[-inf,-inf,-inf,-inf,   0, -inf;
  -inf,-inf,-inf,   0,-inf, 100;
  -inf,-inf,-inf,   0,-inf, -inf;
  -inf,   0,   0,-inf,   0, -inf;
     0,-inf,-inf,   0,-inf, 100;
  -inf,   0,-inf,-inf,   0, 100];
gamma=0.80;           % learning parameter
goalState=6;
```

```
q=zeros(size(R));     % initialize Q as zero
q1=ones(size(R))*inf;  % initialize previous Q as big number
count=0;              % counter
```

```
for episode=0:50000
  % random initial state
  y=randperm(size(R,1));
  state=y(1);

  while state~=goalState,          % loop until reach goal state
     % select any action from this state
     x=find(R(state,:)>=0);       % find possible action of this state
     if size(x,1)>0,
       x1=RandomPermutation(x);   % randomize the possible action
       x1=x1(1);                  % select an action
     end
     qMax=max(q,[],2);
     q(state,x1)= R(state,x1)+gamma*qMax(x1);   % get max of all actions
     state=x1;
  end

  % break if convergence: small deviation on q for 1000 consecutive
  if sum(sum(abs(q1-q)))<0.0001 & sum(sum(q >0))
    if count>1000,
      episode       % report last episode
      break         % for
    else
      count=count+1; % set counter if deviation of q is small
    end
  else
    q1=q;
    count=0; % reset counter when deviation of q from previous q is large
  end
end
```

```
%normalize q
g=max(max(q));
if g>0,
  q=100*q/g;
end
```

The Matlab code above is using my own function RandomPermutation which return random permutation of matrix A. Unlike built-in Matlab function randperm(n) that give permutation of integer 1:n only, RandomPermutation rearrange member of matrix A randomly. This function is useful for MonteCarlo Simulation, Bootstrap sampling, game, etc.

Download the complete Matlab code of this tutorial here (purchase.html)

< Previous (Tower-of-Hanoi.htm) | Next (Q-Learning-Excel.htm) | Contents (index.html) >

Preferable reference for this tutorial is

Teknomo, Kardi. 2005. Q-Learning by Examples. http://people.revoledu.com/kardi/tutorial/ReinforcementLearning/index.html