Deep Q-Learning

The key question that we still need to answer is how to train a neural network so that the network would learn to select the best actions at every time step. To do this we will implement what is called a "Deep Q-learning Algorithm". Let us write the function computed by our neural network as Q. Then, we can use this function to determine the value of a specific action a at a specific state S by simply looking up a value $Q(S, a)$.

More specifically, given a state S as an input, our network outputs an 8-dimensional vector corresponding to each of the eight actions. For example, after performing a forward pass of the network with a state S as an input, we can obtain a value $Q(S, 3)$ by accessing the variable `nn.a{n_layers}(3).` The higher the value associated with this action is, the more reward the robot can expect. Then to select the best action a^* at the current state S we can simply select the action associated with the highest output value:

$$a^* = \arg\max_a Q(S, a)$$

However, this still doesn't answer our original question of how to train a network to produce meaningful action predictions. To tackle this problem, let us focus on a single transition from a state S. At this point, our goal is to select an action that 1) yields the best immediate reward r and 2) also has the highest long-term reward. This intuition can be expressed by writing $Q(S, a)$ as:

$$Q(S, a) = r + \gamma \max_{a_{new}} Q(S_{new}, a_{new})$$

where r is the immediate reward obtained from taking action a from state S. Note that taking an action a from state S then leads to a new state S_{new}. Then, the term $Q(S_{new}, a_{new})$ is a scalar indicating our prediction of how beneficial taking an action a_{new} from the new state S_{new} will be, and γ is a scalar that represents how much we care about the long term reward.

Intuitively, this equation makes a lot of sense. It is saying that the maximum future reward $Q(S, a)$ for a state S and action a, should be equal to the immediate reward r of taking action a plus the maximum future reward obtained by taking the best action from the next state S_{new}.

Thus, our goal is to teach the network to predict $Q(S, a)$ value that is close to $r + \gamma \max_{a_{new}} Q(S_{new}, a_{new})$. This looks very similar to what we did in the previous lab when we had the ground truth labels, and we tried to force the network to produce predictions that were similar to those ground truth labels. However, now instead of using the provided ground truth labels we are using the values $r + \gamma \max_{a_{new}} Q(S_{new}, a_{new})$ as our "ground truth" labels to predict $Q(S, a)$.

Your job is to compute the value $r + \gamma \max_{a_{new}} Q(S_{new}, a_{new})$ and store it into a `target(action)` variable that will be used to update the network's parameters during backpropagation. Note that you should use the `ForwardPass.m` function from the previous lab. You can simply call this function in your code.

Your Function

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
function nn=DeepQLearning(action,reward,S,new_S,nn)
    % Update the weights of the network
    %
    % Input:
    % - action: a scalar indicating which action was selected in state S
    % - reward: a scalar indicating the reward value that was obtained by transition to state new_S
    % - S: n x m matrix that stores the previous state
    % - new_S: n x m matrix that stores the current state
    % - nn: a variable storing a neural network
    % Output:
    % - nn: a variable storing a neural network with the updated weights after a single Q-learning step.


    lr=0.01; %learning rate
    gamma=0.8;
    n_layers=numel(nn.W)+1;

    %% Initializing target variable
    nn = ForwardPass(nn, S(:)');
    target=nn.a{n_layers}';


    %% Computing the maximum Q value of next action by feeding the new state new_S through a network
    new_nn = ForwardPass(nn, new_S(:)');
    output_vec=new_nn.a{n_layers};
    out_max = max(output_vec);

    %% Setting a target(action) value
    target(action)= reward + gamma * out_max;

```

Code to call your function

Reset

```
% Current state
rows=7; cols=7;
walls=[2 4; 3 4; 4 4; 5 4];
cur_row=2; cur_col=1; rot_idx=1;
S=MakeState(rows,cols,walls,cur_row,cur_col,rot_idx);

% Neural Network
nn = InitializeNetwork([rows*cols 8]);

% Picking action deterministically
epsilon=0;
action=pickAction(S,cur_row,cur_col,rot_idx,nn,epsilon);

% Getting a reward
reward=GetReward(S,cur_row,cur_col,rot_idx,action);

%Transitioning to a New State
[new_S,~,~,~]=MakeNextState(S,cur_row,cur_col,rot_idx,action);

% Deep Q-Learning
nn=DeepQLearning(action,reward,S,new_S,nn);
```

▶ Run Function

Previous Assessment: All Tests Passed

Submit

Was the 'target' Variable Set Correctly?