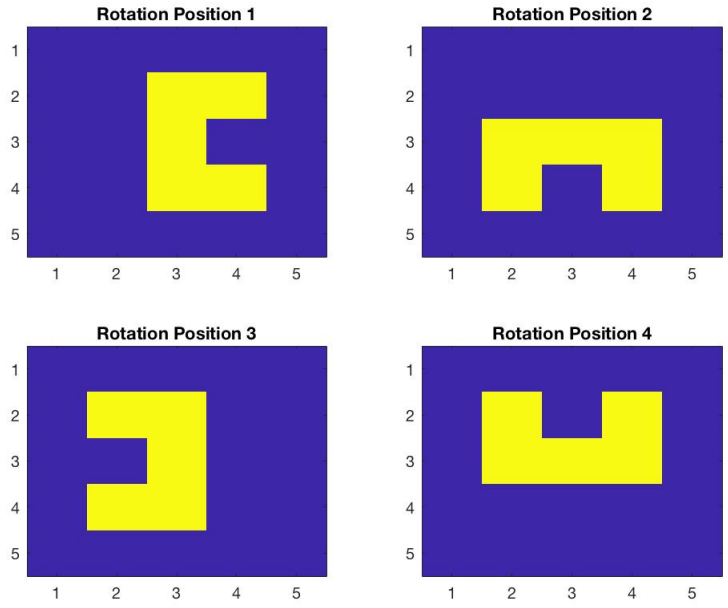
Selecting an Action

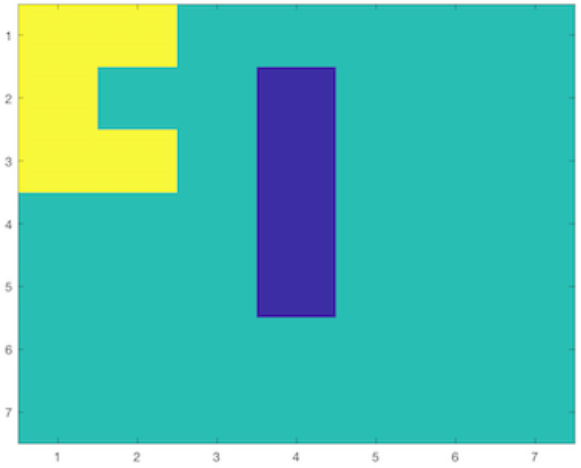
In order to reach the goal state the robot needs to navigate through the environment. It does so by selecting one of the eight actions at every step of its exploration. Specifically, the robot can choose to move $\{up, down, right, left\}$. Furthermore, as discussed earlier, the robot can rotate into one of the four positions.



At every step, the robot chooses one of these eight actions, and carries it out assuming that the selected action is valid (i.e. the robot stays within the grid boundaries, doesn't hit the walls, etc). We want our robot to learn to select actions such that each selected action would have a good chance of leading to a reward (i.e. reaching a goal state).

To achieve this goal, we will use a neural network. Specifically, we will train a neural network that takes a current robot's state as an input and then outputs an eight dimensional vector corresponding to the likelihood of each of the eight actions.

As illustrated previously, at every step, the state that the robot is in can be represented as a $n \times m$ dimensional grid such as the one illustrated below:



We can flatten this 2D grid into an $1 \times nm$ dimensional vector and feed it to the network as an input. Then, the network will output an eight dimensional vector corresponding to the likelihood of each of the eight actions. We will then select an action that corresponds to a highest likelihood value, and carry that action out. Note that we haven't discussed how we will train this network yet. We will do this in the later sections. For now let us assume that we know how to train a network to pick the best actions and that we already have such a trained network that can do so.

Before going into technical details of this part of the assignment, we also want to point out that every reinforcement learning application has to balance between the so called "exploitation" and "exploration" effects. Exploitation refers to a process when we know, which actions lead to the reward and we simply select those actions greedily without exploring the environment much further. In the context of our problem, this could be described as always taking actions corresponding to the maximum prediction values from our neural network.

The problem with a purely exploitative strategy is that the robot stops exploring: it simply follows the strategy that already worked once or twice. However, that strategy may not be optimal. In order to discover a strategy that may work even better, we must allow the robot to explore, i.e. take actions that have not been tried before and that may lead to discovering better ways to get rewarded. In the context of our problem, we will implement an exploration strategy by allowing the robot to select random actions.

To summarize, we present a pseudo code for selecting an action at any given step.

Algorithm 1 Selecting an Action

```
if rand()>ϵ then
    next_action=NeuralNetwork(current_state)
else
    next_action=Randomly select one of eight actions.
end if
```

With a certain probability specified by an ε parameter you should pick an action based on a neural network's output, otherwise you should pick one of the eight actions randomly. Note that every action is encoded by a number between 1 and 8. Thus, your function should output a single number associated with a selected action. Furthermore, you should use the same `ForwardPass.m` function for a neural network's part of the algorithm that you implemented in the previous lab. In your code, you can call this function as `ForwardPass(nn,x),` where `nn` is a variable storing the neural network structure, and `x` is a feature vector. Then, once the forward pass of the network has been performed the desired output will be stored in 8×1 dimensional `nn.a{n_layers}` variable where `n_layers` depicts the number of layers in the network.

Your Function

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

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

```
33 function action=pickAction(S,cur_row,cur_col,rot_idx,nn,epsilon)
34     % Pick an action for the robot to carry out
```

Code to call your function

Reset

```
1  % Current state
2  rows=7; cols=7;
3  walls=[2 4; 3 4; 4 4; 5 4];
4  cur_row=2; cur_col=1; rot_idx=1;
5  S=MakeState(rows,cols,walls,cur_row,cur_col,rot_idx);
6
7  % Neural Network
8  nn = InitializeNetwork([rows*cols 8]);
9
10 % Picking an action
11 epsilon=0.5;
12 action=pickAction(S,cur_row,cur_col,rot_idx,nn,epsilon);
```

▶ Run Function

Previous Assessment: All Tests Passed

Submit

- ✓ **Does the Function Return Correct Action from a Neural Network?**
- ✓ **Does the Function Randomly Select an Action?**