Assignment 6: Reinforcement Learning

Xiyu Wang

xw5638 xiyu_wang@utexas.edu

May. 10, 2020

1 Introduction

Reinforcement learning is the training of machine learning models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning, an artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the AI gets either rewards or penalties for the actions it performs, and the goal is to maximize the total reward. Although the designer sets the reward policy, it's up to the model to figure out how to perform the task to maximize the reward, starting from totally random trials and finishing with sophisticated tactics and superhuman skills.

In this report, an agent is trained to finish a combined task to walk down a sidewalk, avoiding hitting on the obstacles and picking up litters. First, four subtasks are analyzed as parts of the combined task. For each subtask there is a model to be trained with the algorithm of Q-learning. During this process, a state system, a R matrix, and a Q matrix are set for each model. After that, the models are combined together to finish the combined task in a grid with both obstacles and litters. No RL package is used in this experiment for better understand of the inner schema.

2 Methods

2 (1) Q-Learning

Q-learning will use the Q value as a cheat sheet for the agent. It will perform the sequence of actions that will eventually generate the maximum total reward. The strategy is as: stattee s and performing action a is the immediate reward r(s, a) plus the highest Q value possible from the next sate s'.

$$Q(s,a) = r(s,a) + \gamma \max_a Q(s',a)$$

With experience, it will converge to the optimal policy. In practical solutions, this is implemented as an update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Where alpha is the learning rate or step size. By adjusting gamma we will diminish or increase the contribution of future rewards.

The pseudo code is as follows:

```
Define epsilon, alpha, gamma
Initiate Q-matrix with zeros
Repeat:
        Randomly select an initial state
           Observe initial state s
           Repeat:
                      Select action with Greedy-epsilon algorithm so that
                      With rate epsilon:  a = random(a')
                      With rate (1 - epsilon): a = argmax(Q(s, a'))
                      Determine next state s'
                      Update Q(s, a) = (1 - alpha) * Q(s, a) + alpha * (R(s, a) + gamma * max(Q(s', :)));
                      S = s'
        Until reach an amount or reach the destination
        Normalize each row of Q
Until reach an amount and assume the model is well trained
```

2 (2) Avoid Obstacles

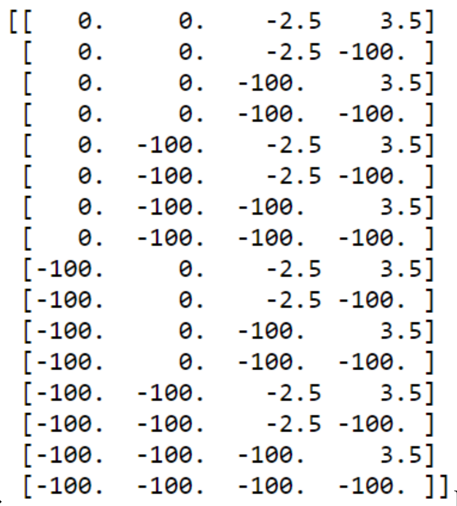
For the obstacle model, the map is designed with numbers 0 and 1, which 1 means there's an obstacle.

The state is defined with four digits representing up, down, left and right. For instance, if a state is 0101, then it has obstacles down and right. The value of the state is 5.

Especially, for when it comes to the edge of the map, going out of the map is regarded as an obstacle. For instance, a state 0100 can also be a state with no obstacle around but at the bottom of the map and can not do down.

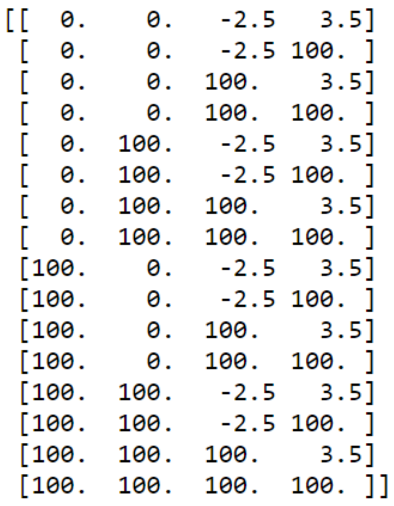
For the R matrix, given a state and an action, it will give a reward/punishment value. Initialize R with zeros and go through all the states. If the action will lead to an obstacle, do R[s][a] -= 100 to punish it. Elsewise, reward going forward with +3.5 and punish going backward with -2.5. The R matrix is shown in the figures.

R_Obstacle:



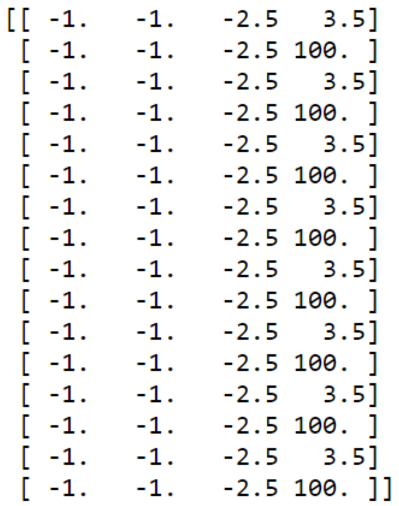
```
[[   0.      0.     -2.5    3.5]
 [   0.      0.     -2.5 -100. ]
 [   0.      0.   -100.     3.5]
 [   0.      0.   -100.  -100. ]
 [   0.   -100.     -2.5    3.5]
 [   0.   -100.     -2.5 -100. ]
 [   0.   -100.   -100.     3.5]
 [   0.   -100.   -100.  -100. ]
 [-100.      0.     -2.5    3.5]
 [-100.      0.     -2.5 -100. ]
 [-100.      0.   -100.     3.5]
 [-100.      0.   -100.  -100. ]
 [-100.   -100.     -2.5    3.5]
 [-100.   -100.     -2.5 -100. ]
 [-100.   -100.   -100.     3.5]
 [-100.   -100.   -100.  -100. ]]
```

R_Litter:



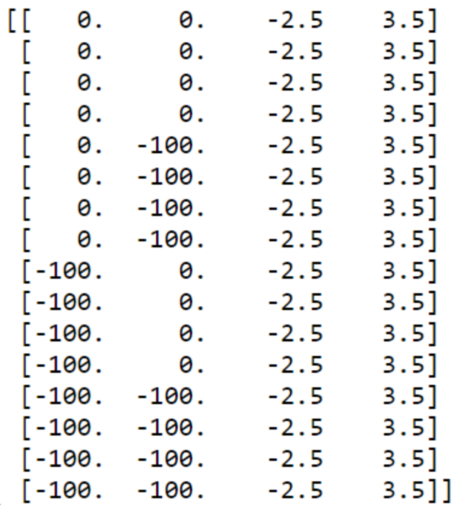
```
[[  0.    0.   -2.5   3.5]
 [  0.    0.   -2.5 100. ]
 [  0.    0.  100.    3.5]
 [  0.    0.  100.  100. ]
 [  0.  100.   -2.5   3.5]
 [  0.  100.   -2.5 100. ]
 [  0.  100.  100.    3.5]
 [  0.  100.  100.  100. ]
 [100.    0.   -2.5   3.5]
 [100.    0.   -2.5 100. ]
 [100.    0.  100.    3.5]
 [100.    0.  100.  100. ]
 [100.  100.   -2.5   3.5]
 [100.  100.   -2.5 100. ]
 [100.  100.  100.    3.5]
 [100.  100.  100.  100. ]]
```

R_Godown:



```
[[ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]
 [ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]
 [ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]
 [ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]
 [ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]
 [ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]
 [ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]
 [ -1.   -1.   -2.5   3.5]
 [ -1.   -1.   -2.5 100. ]]
```

R_Stay:



```
[[   0.      0.     -2.5     3.5]
 [   0.      0.     -2.5     3.5]
 [   0.      0.     -2.5     3.5]
 [   0.      0.     -2.5     3.5]
 [   0.   -100.     -2.5     3.5]
 [   0.   -100.     -2.5     3.5]
 [   0.   -100.     -2.5     3.5]
 [   0.   -100.     -2.5     3.5]
 [-100.      0.     -2.5     3.5]
 [-100.      0.     -2.5     3.5]
 [-100.      0.     -2.5     3.5]
 [-100.      0.     -2.5     3.5]
 [-100.   -100.     -2.5     3.5]
 [-100.   -100.     -2.5     3.5]
 [-100.   -100.     -2.5     3.5]
 [-100.   -100.     -2.5     3.5]]
```

2 (3) Picking Up Litters

For the litter model, the map is designed with numbers 0 and 1, which 1 means there's a litter to pick up. If the agent picks up the litter, the map will be changed, and the litter will be gone. The W will be changed from 1 to 0 consequently.

Similarly, the state is defined with four digits representing up, down, left and right. For instance, if a state is 0101, then it has litters down and right. The value of the state is 5.

For the R matrix, initialize R with zeros and go through all the states. If the action will lead to a litter, do R[s][a] += 100 to reward it. Elsewise, reward going forward with +3.5 and punish going backward with -2.5. The R matrix is shown in the figures.

2 (4) Going Down the Sidewalk

For the go-down model, it does not matter about what inside the map. The shape of map matters.

Similarly, the state is defined with four digits representing up, down, left and right. But only the last column is the target so for any state if going right reaches the target, it will be set as 1. For instance, if a state is 0001, then it can reach target by going right. The value is 1.

For the R matrix, initialize R with zeros and go through all the states. If the action will go right and reach the target, do R[s][a] += 100 to reward it. Elsewise, reward going forward with +3.5 and punish going backward with -2.5. Specially, for this case punish the agent by going up and down for nothing by -1. The R matrix is shown in the figures.

2 (5) Staying Within the Sidewalk

For the stay model, it does not matter about what inside the map either.

Similarly, the state is defined with four digits representing up, down, left and right. But only the first row and last row are set as "outside of the sidewalk". For any state if going up or down reaches "outside", it will be set as 1. For instance, if a state is 0100, then it can go outside of the sidewalk by going down. The value is 4.

For the R matrix, initialize R with zeros and go through all the states. If the action will go up/down and reach outside, do R[s][a] -= 100 to reward it. Elsewise, reward going forward with +3.5 and punish going backward with -2.5. The R matrix is shown in the figures.

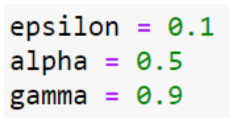
2 (6) Combine Models

Using the states, R matrix, and Q matrix of all the models, combined model can directly decide the best action by defining the state for four models, looking up the Q values and find the best action with maximum mean value of 4 Qs.

For combined goals, the map W is set as 1 as obstacles and 2 as litters.

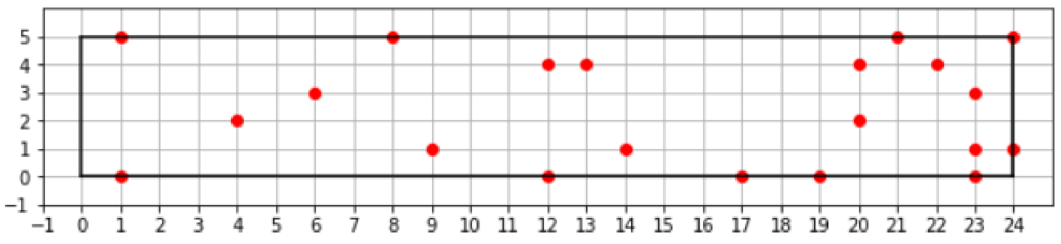
3 Results

In the experiment, fore modes discussed above are trained separately and combined as one. The grid is of dimensions 6 x 25 and the parameters are set as follows:



```
epsilon = 0.1
alpha = 0.5
gamma = 0.9
```

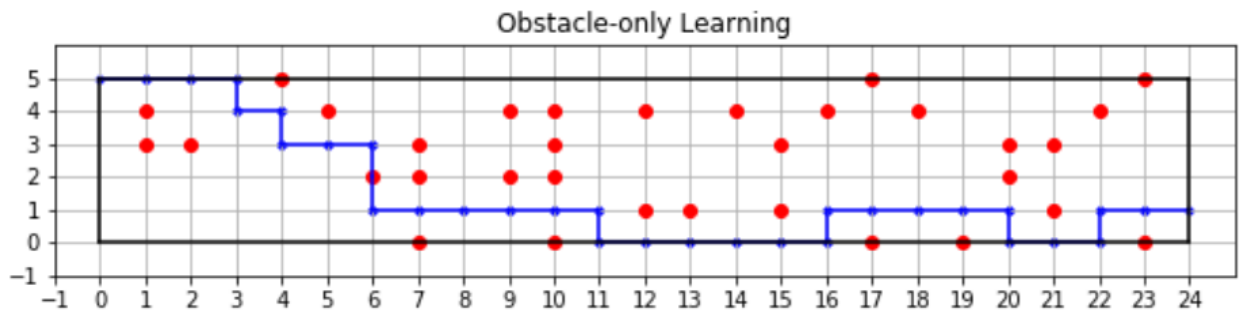
When generationg the map W, obstacles and litters are appering at the rate of 1/5. For the separated models, W only contains one variables such as obstacles. We can plot the grid before we set the agent:

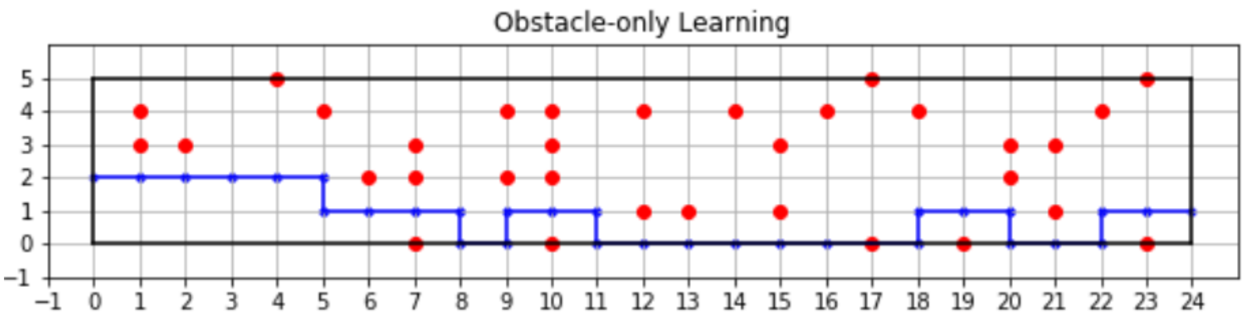


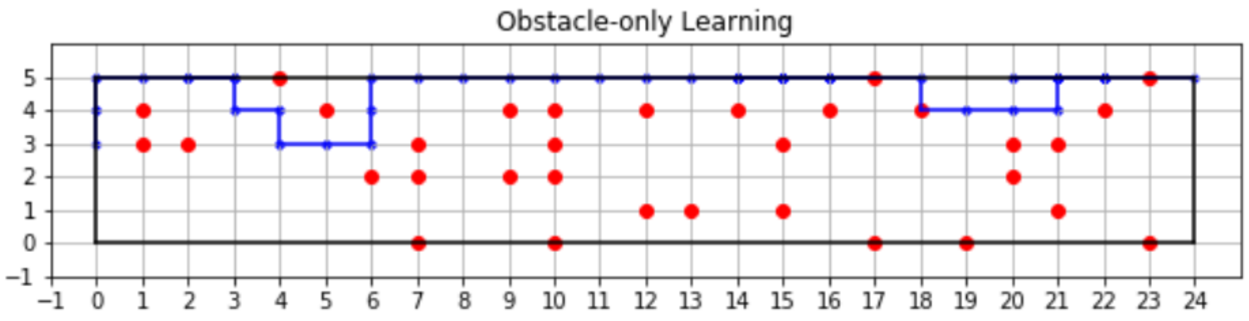
During the experiment, python3 is used and no RL package is used for better understading of the model. Within each training, 20000 episodes are trained, within chich the agent will try to go 1000 steps unless it reaches the destination: the last column. After the training, a demo round is applied for plotting the path of the agent.

3 (1) Avoid Obstacles

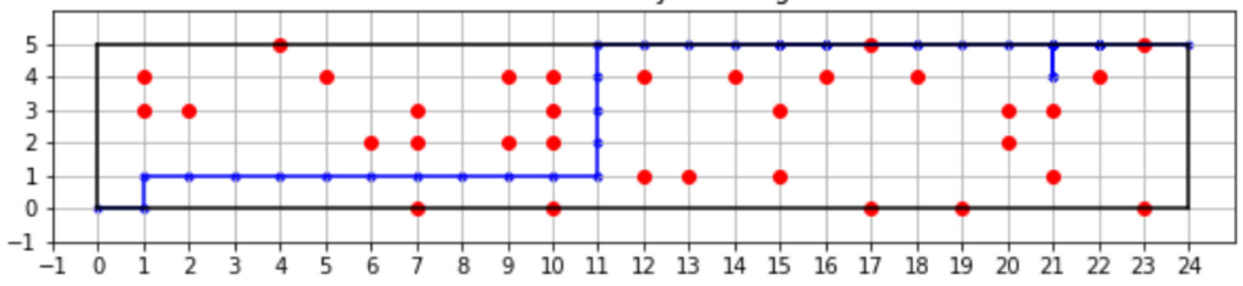
Given a random start point at (0, random.randomint(0, 5)), demo the agent's path. The red dots are regarded as obstacles.



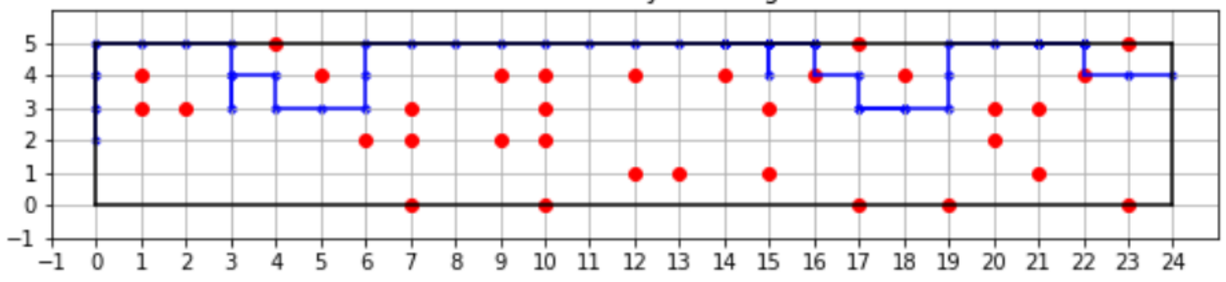




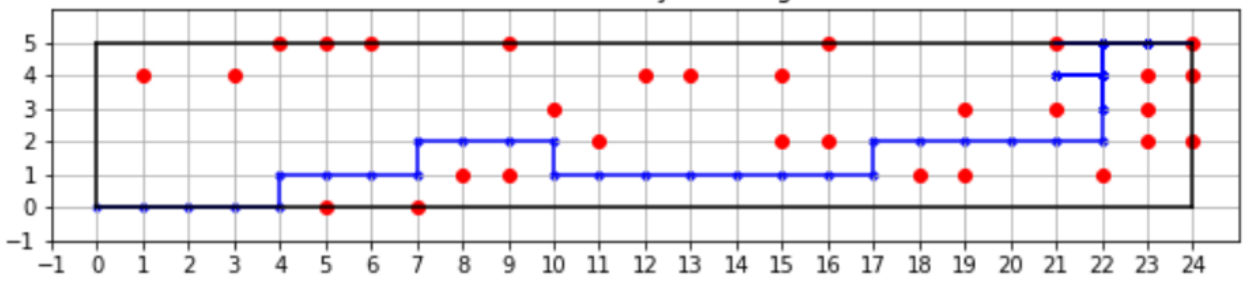
Obstacle-only Learning



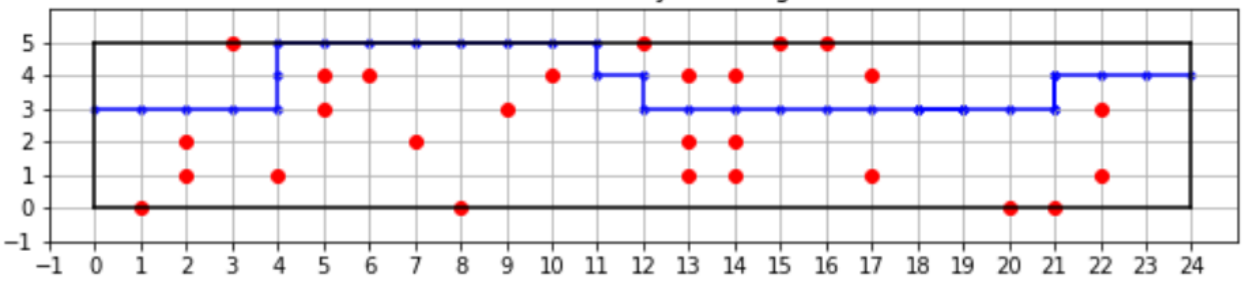
Obstacle-only Learning



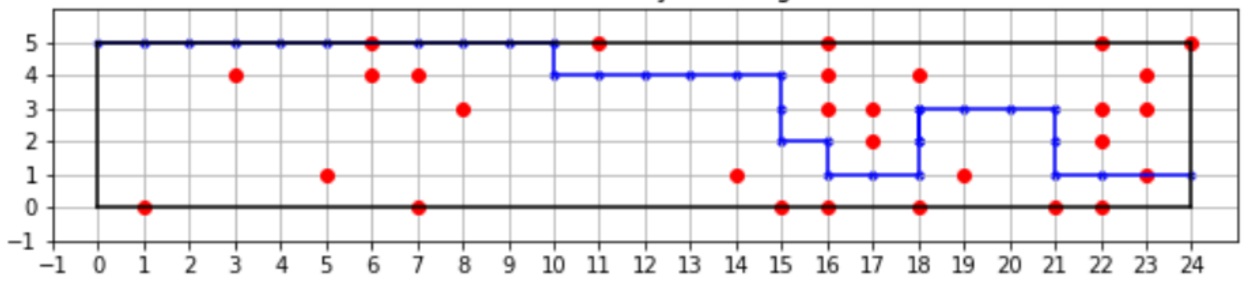
Obstacle-only Learning



Obstacle-only Learning



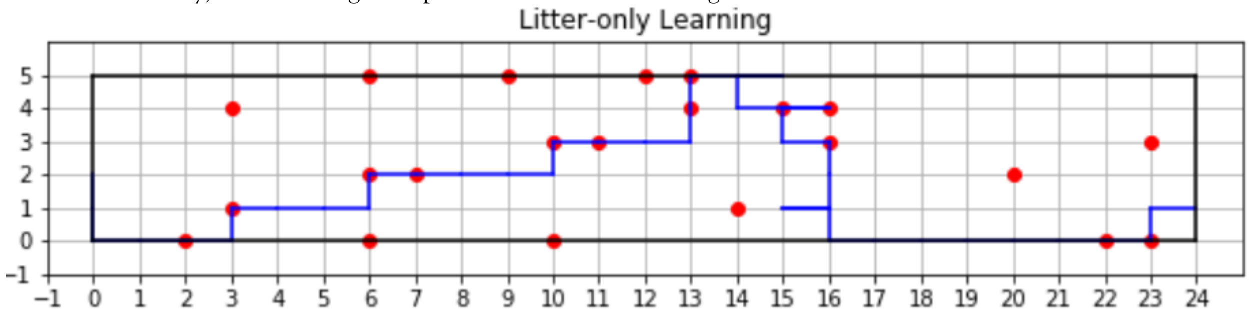
Obstacle-only Learning

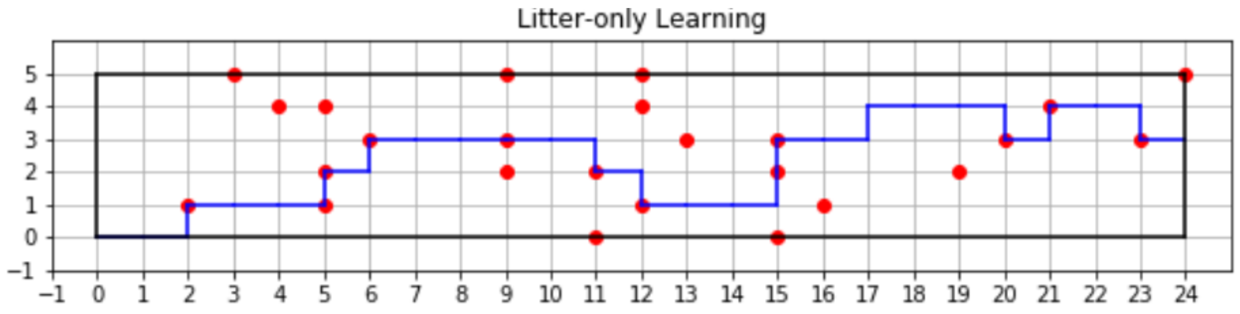


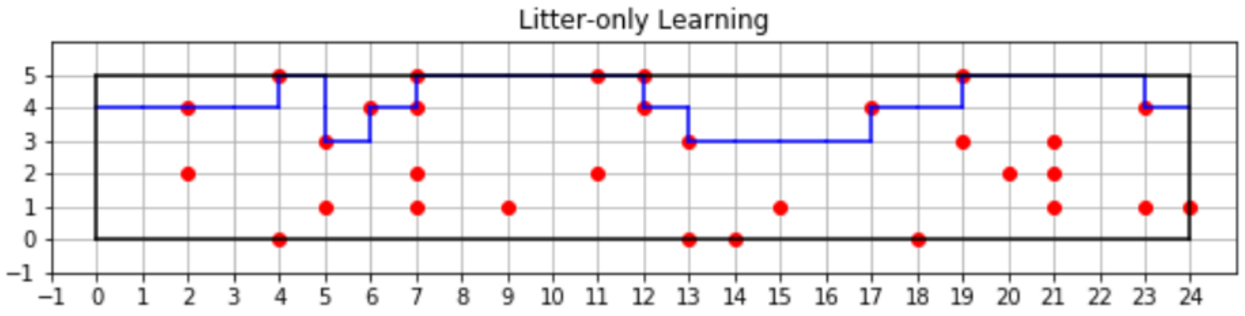
According to the figures, the agent can generally avoid hitting into most of the obstacles. In some cases it's impossible to avoid an obstacle and in others it will require a long detour to avoid them. So we can see in come cases it gets stucked into the surroundings of obstacles and hits into one of them.

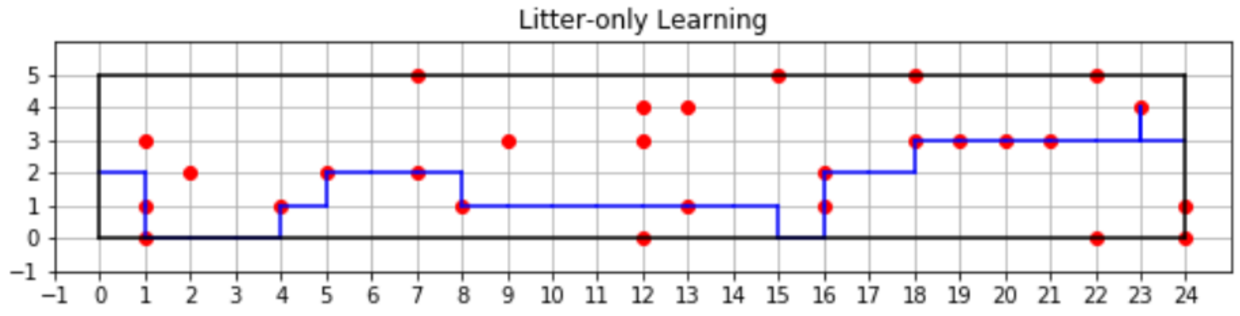
3 (2) Picking Up Litters

Similarly, demo the agent's path. The red dots are regarded as litters.





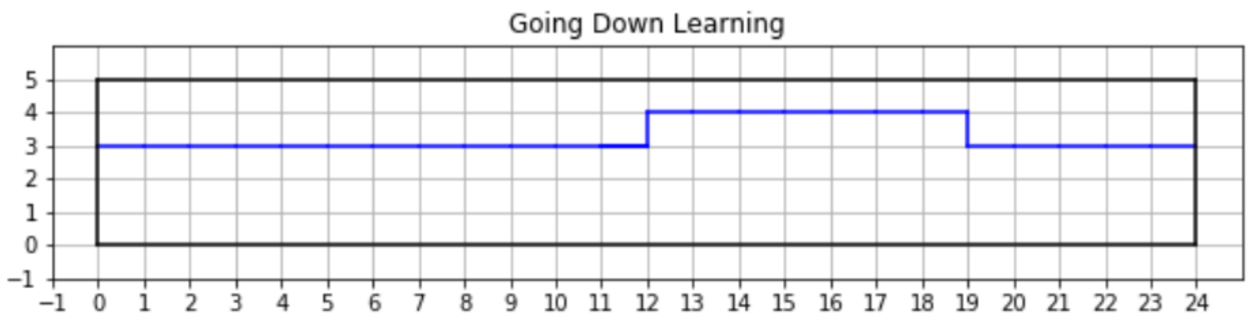


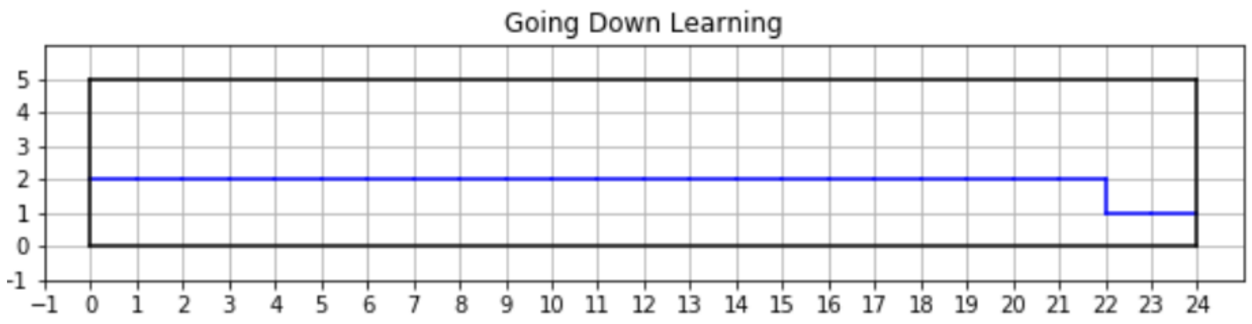


According to the figures, the agent can generally pick up litters and go for it on purpose instead of randomly picking up the ones on it's path. Because of the reward of going right and punishment of going left, it may not go for a far way or return back for a signle litter.

3 (3) Going Down the Sidewalk

Regardless the obstacles / litters, demo going down. The less it returns / goes up / goes down, the better.

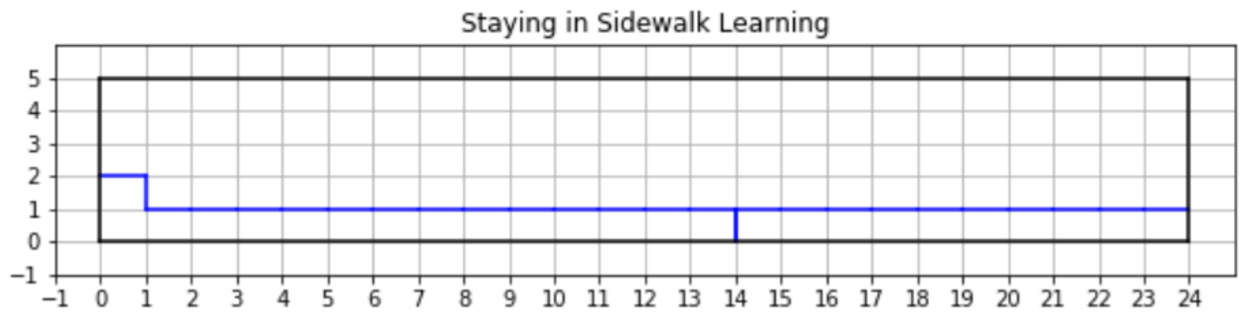


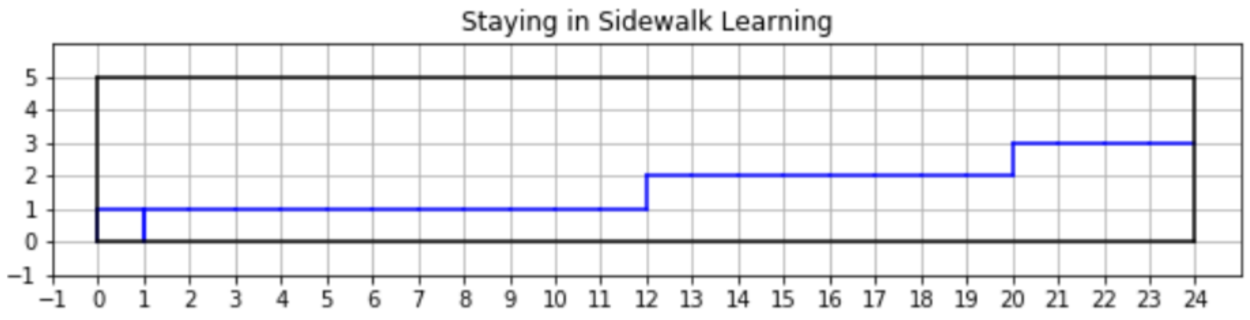


It can be generally accomplished with the reward of going right and punish of going left. The punish of going up and down can help decrease unnecessary move and keep heading to the target.

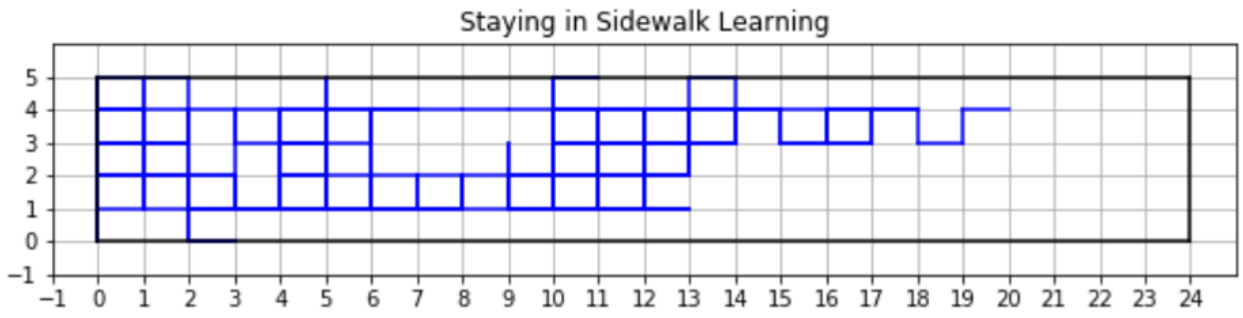
3 (4) Staying on the Sidewalk

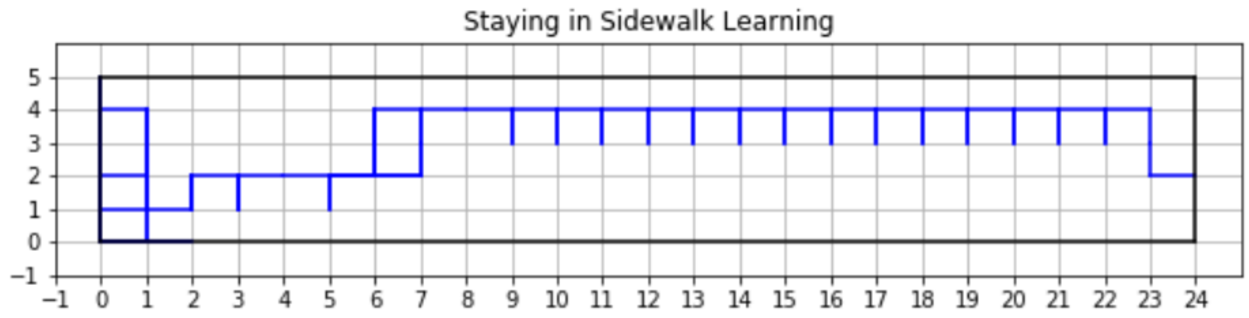
Regardless the obstacles / litters, demo staying on the sidewalk. Assume x within [1, 4] are the sidewalk and x =0 and x = 5 are outside.

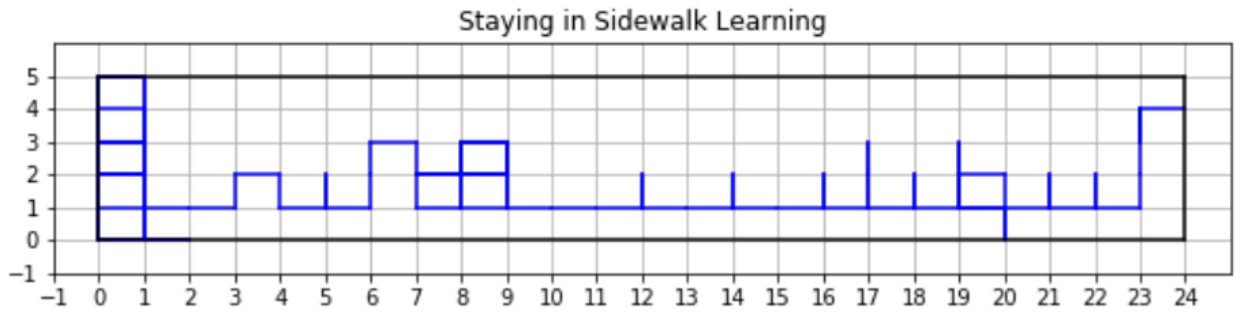




It might be difficult to observe how the model work. The demo can be more observable if we remove the award of going right and the punishment going left. Then the agent will not try to heading forward but walking randomly. In this way we can find if it can stay in the sidewalk.



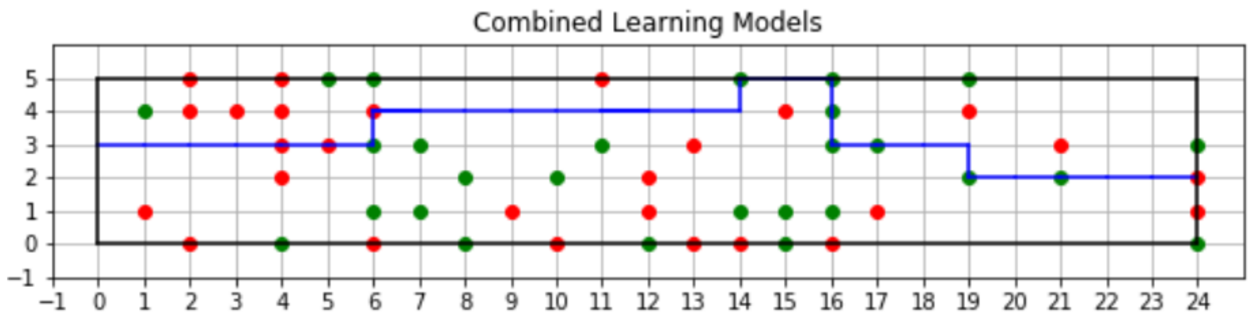


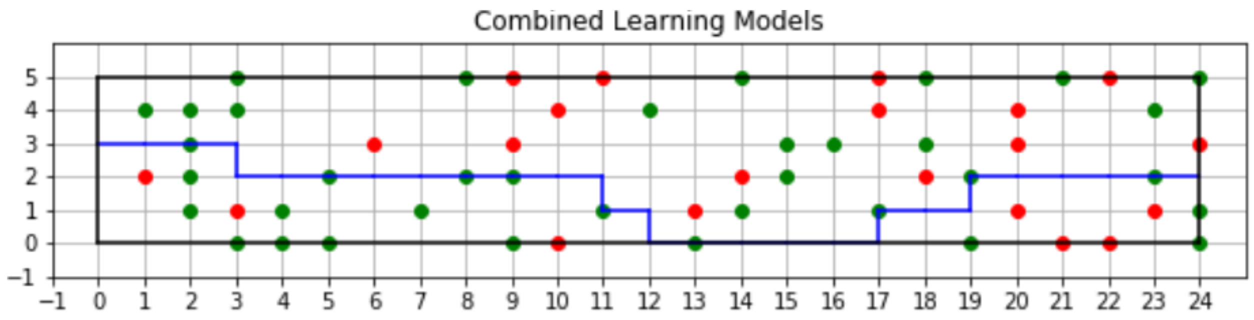


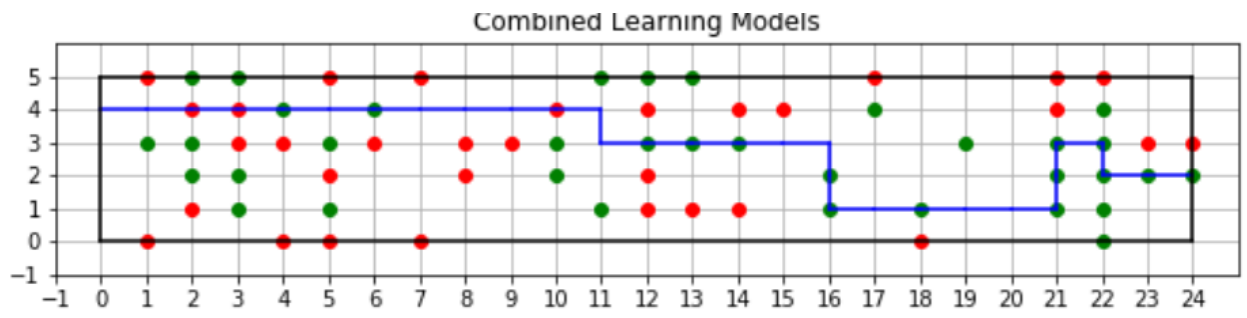
In this way, the agent can generally avoid going outside of the sidewalk by reaching the row 0 and row 6.

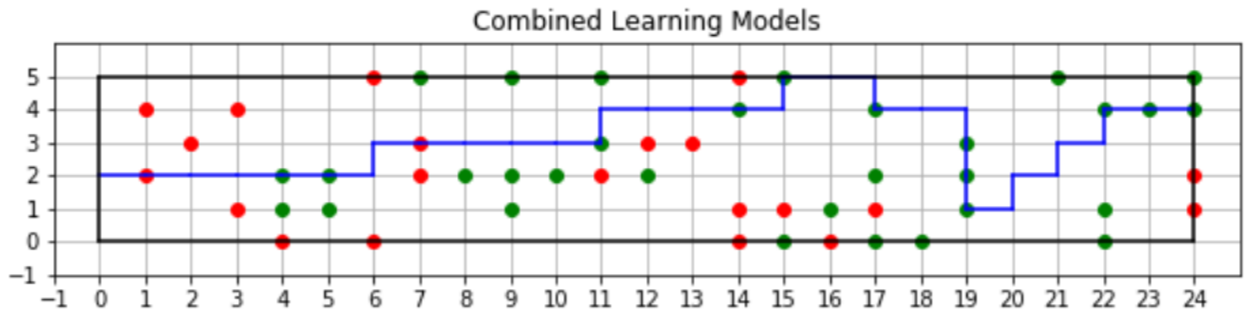
3 (5) Combined model

Combine the four models as discussed above and plot the figures. The red dots are obstacles and the green ones are litters.









According to the figures, the agent can pick up litters on purpose and avoid most of the obstacles. It goes out of the sidewalk sometimes if there's a litter to pick up. But generally, it can keep heading to the target forward. If there's no litter set outside of the sidewalk, it will mostly not go outside.

4 Summary

In this experiment, reinforcement learning is applied to build an agent who intend to avoid obstacles, pick up litters and go down within the sidewalk. Generally, all the four models are working well in their purpose. The Q-learning algorithm is working in the proposed story.

When combine the models together, the effect is slightly not as good as the ones in separated models. The main reason might be the weight of the different goals. Normalization may not influence the result of single model but for four models to compute the average value, there might be some misunderstanding between the designer and the agent. The agent might think it's most important to going down than avoiding the obstacles in some cases so it just go straight to the obstacles. I don't think this problem can be solved by simply combine the four models. It requires a new definition of R matrix with much more states when the obstacles, litters, sidewalk are all making effect.