Reinforcement Learning - Self-Driving Cars

Nayan Singhal
University of Texas at Austin
Austin, TX 78712
nayans@cs.utexas.edu

Abstract

This report presents the experiments and results of implementing modular reinforcement learning algorithms on grid world. The task of navigating an obstacle ridden grid is divided into five modules. One module is to learn how to reach the destination of the grid(end of the sidewalk), the other modules learns to avoid static obstacles, dynamic obstacles, traffic light and pedestrian crossing. We develop reinforcement learning agents that learns each of the modules separately. The final agent that navigates this grid is developed as a combination of these five individual modules.

1. Introduction

Reinforcement learning is an area of machine learning concerned with how the software should take the actions in the reward in order to get the maximum reward. Reinforcement learning algorithm always tries to learn a strategy to navigate the grid by experimenting in each step as opposed to the fixed or deterministic policy that is fixed ahead of time and fixed for each grid. The advantage of the reinforcement learning is that it is capable of choosing an optimal policy that can adapt to different grid layouts, different obstacles and complex reward policies. Further, there is a focus of finding the balance between the exploration(unknown territory) and exploitation (known territory).

A reinforcement learning is defined as a Markov Decision Process (MDP) or a POMDP (partially observable Markov decision process). This implies that an agents actions is determined by the state in which it is in and the reward it gets by performing a chosen action. The basic reinforcement learning algorithm consists of :

1. Set of environment and agent state, S
2. Set of actions that is defined for the agent
3. Policy defined for the transition from state to action
4. Reward that defines the immediate reward for the transition
5. The environment which agent observes.

The agent measures the value of the state, determine the policy or the best action to perform in a particular state. And then perform the action based on the balance between the exploration and exploitation. The Environment defines the reward for the action taken by the agent which agent used to update its measures so that it can take best action in future measures. The performance of the reinforcement learning agent depends largely on the design of the state space, actions and the reward model. After defining it, the learning rate can be controlled by the learning rate, the discount rate and the exploration probabilities so that agent can try other paths too instead of choosing the best policy all the time.

This assignment focuses on the task of implementing reinforcement learning algorithm on self-driving cars. The objective in the grid world is to design an agent that can traverse through the grid with obstacles, parked cars, dynamic cars, follow the traffic light, stop for pedestrians and get to the other end of the grid. In modular approach to reinforcement , the entire task is divided into sub-tasks and have the agent learn each of the sub-tasks individually. And, in the final step, combine all of the sub-tasks by learning how to weight them in order to solve the whole task successfully.

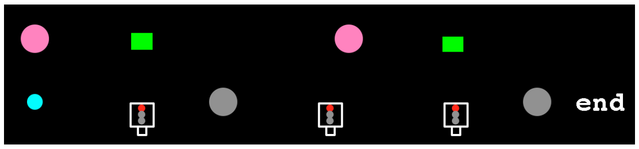

Figure 1: Simulated environment for the self-driving car. Blue dot represent the agent, pink represents the dynamic cars which have a speed of 1 cell per time-stamp.Grey represent the static obstacles. Green represent the pedestrian crossing the grid. And there are traffic lights also on the right side of the lane.

2. Methodology

The objective in the grid world is to design an agent that can traverse through the grid with obstacles, parked cars, dynamic cars, follow the traffic light, stop for pedestrians and get to the other end of the grid. Figure 1 shows the simulation or the environment created to complete the overall task. For the task, we have created five modules or subtask to complete the complete task: walk (simply walk from one start location to the end without obstacles), static obstacles, dynamic obstacles, traffic light, pedestrian. And, define five actions: north, south, east, west, stop.

Reward model is also decided based on the task and may vary for the same task if the state space is defined differently. Some examples for reward models are:

- reward only on completing the task.
- reward for living. i.e. you might want to reward an agent just for moving and exploring the space.
- reward for small subtasks such as moving in the direction of the eventual goal.
- reward (or penalty) for not doing (or doing) an action.

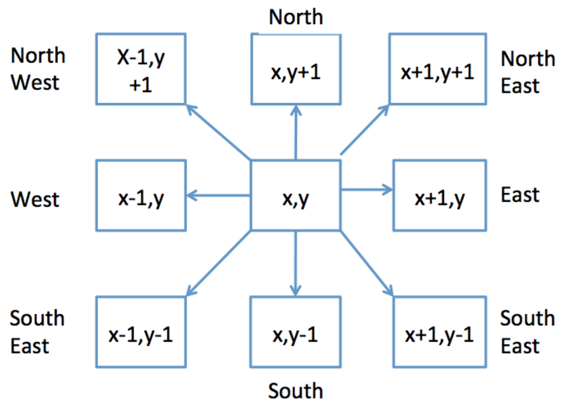

Figure 2: Center blog represents the current location of the car. And each direction represents the change of co-ordinate location in grid.

Now, we will define the state space and the reward selected for each module.

2.1. Walk

For the walk module, co-ordinate position (x,y) of the grid is used as the state space. The reward (0.5) will be given only is it is moving in the direction of the goal i.e. in east direction, otherwise, -0.5 will be given for all other actions (north, south, west, stop). And, reward of 10 is given if it reaches the goal.

2.2. Static Obstacles

For the static obstacle module, the combination of (north, east, south) cells are considered as state space. The reward of 10 is given if we didnt hit the obstacle and -5 if we hit the obstacle.

2.3. Dynamic Obstacles

For the dynamic obstacle module, the combination of (north-west, North) cells are considered as state space. The reward of 10 is given if we didnt hit the obstacle and -5 if we hit the obstacle.

2.4. Traffic Lights

For the traffic light module, the traffic light condition at next column are considered as state space. The reward for the module are:

1. 10, if we move east while the traffic light is green.
2. 10, if we choose stop action while the traffic light is red and green light.
3. -5, if we choose action other than stop while the traffic light is red or yellow light.

2.5. Pedestrian Crossing

For the pedestrian crossing, we considered whether pedestrian is present in the next column or not are considered as the state space. The reward for the module are:

1. 10, if we move east while there is no pedestrian
2. 10, if we choose stop action while pedestrian is crossing.
3. -5, if we choose action other than stop while the pedestrian is crossing.

The q-learning algorithms with greedy strategy was implemented for the modules. The exact algorithm implemented can be found in the book by Sutton and Burto [SB98].

Algorithm defined for each episode:

```
continue till car location = finish
actions = getPossibleActions
action = chooseBestAction using q-learning
nextState, reward = PerformAction
update QLearning Policy(state, action, nextState, reward)
```

3. Experiment

To check the correctness of the Q-learning with different modules. We visualize it by simulating the road environment considering 4 10 grid.

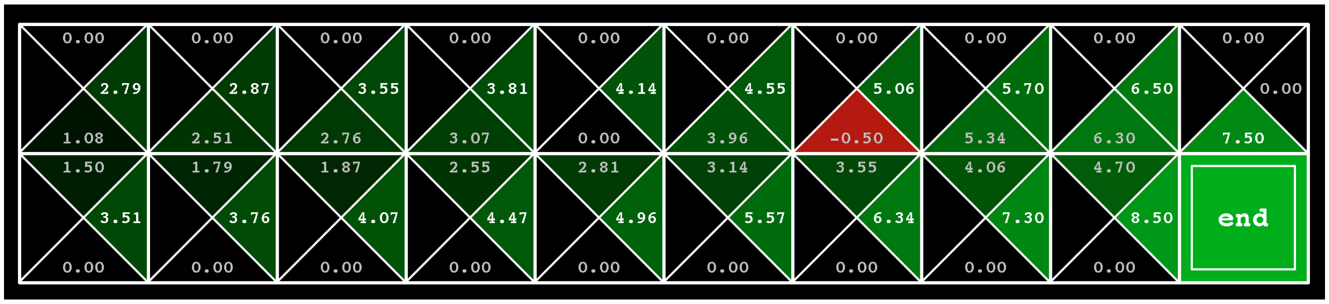

Figure 3: Q Value of each state and best policy learned for walking.

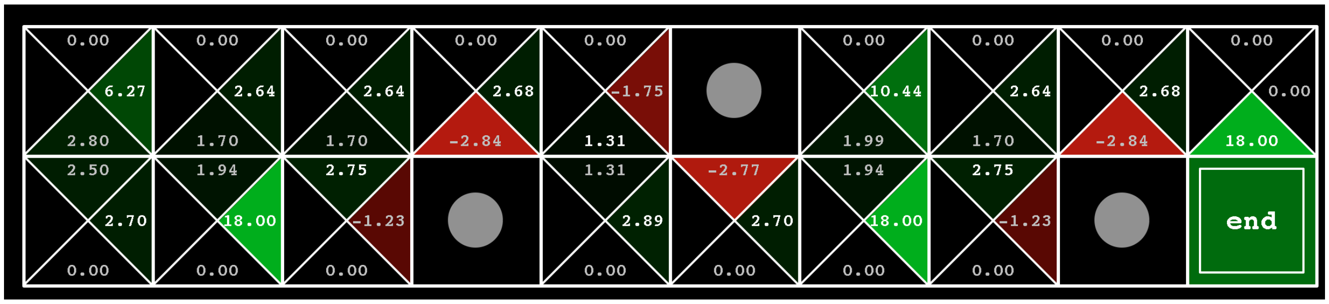

Figure 4: Q Value of each state and best policy learned for Static obstacles. Action which can take the agent towards static cars collision have negative values.

3.1. Walking module

We ran the module for 200 episode and considered the state and actions as described in the previous action.

- Destination reward = 10
- Reward for departing east = 0.5
- Penalty for wrong direction = 0.5
- Learning rate = 0.5
- Epsilon = 0.1(exploration rate)
- State space = grid position (x, y)

The figure 3 shows the 2x10 grid where (0,0) is the starting location and end is the final location. Value at each cell represents the Q value for north, east,south actions. If cars just follow the exploitation policy, then it will reach the final destination without taking north or south actions because Q values for east action is greater than the Q values for all other actions. There are some locations at grid location where value is 0. Its because our agent didnt explore those path and initially your Q-values are 0. There is one location at grid where Q value for south is -0.5, its because agent must have tried to explore the path to move in south direction and our environment gave the reward of -05 because we are not moving in the direction of the destination.

3.2. Static Obstacles Module

We ran the module for 200 episode and keep changing the static obstacles location at each episode.

- Destination reward = 10
- Reward for departing east = 0.5
- Penalty for hitting obstacles = 0.5
- Learning rate = 0.5
- Epsilon = 0.1(exploration rate)
- State space = (north, east, south)

The figure 4 shows the 2x10 grid where (0,0) is the starting location and end is the final location. Value at each cell represents the Q value for north, east,south actions. It is clearly seen that Q values for action moving in the direction of static obstacles is negative which is shown with red triangle. It happened because when our agent explore the situation, it got negative reward from the environment which help the agent to update its Q values. If cars just follow the exploitation policy, then it will reach the final destination without hitting any static obstacles. There are some locations at grid location where value is 0. Its because our agent didnt explore those path and initially your Q-values are 0.

3.3. Dynamic Obstacles Module

We ran the module for 200 episodes and in each episode we have 2 cars which is always changing their location at each time-stamp.

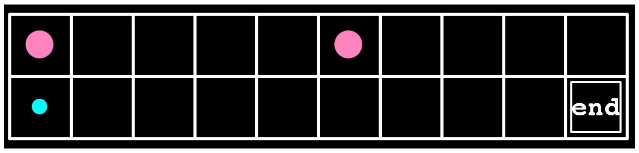

Figure 5: Simulated environment for the dynamic Cars module. Blue dot represent the agent and pink represents the dynamic cars which are moving at 1 cells per time-stamp.

- Destination reward = 10
- Reward for not hitting target = 5
- Penalty for hitting obstacles = 5
- Learning rate = 0.5
- Epsilon = 0.1(exploration rate)
- State space = (North-west, North)

State	**action**	**Q-Value**
'C','F'	north	-1.54
'C','F'	south	9.6
'C','F'	east	0
'C','F'	stop	3.36
'F','C'	north	9.10
'F','C'	south	4.36
'F','C'	east	0
'F','C'	stop	2.98

Table 1: Q values for the dynamic obstacle avoidance. The combination of (north-west, North) cells are considered as state space where C represents car and F represents free space. There are other states space and Q values also but not showing it because of page constraints.

From the table 1, we can see that when we have a dynamic car at the north-west cell, then the Q-Value for north action is negative. Because when agent took the step while exploring, environment changed the location of the car from north-west to north since car is traveling at the speed of 1.0 which caused the agent to have collision with dynamic car. And, because of it, environment gave the negative reward. However, when the car is at north cell, Q-value for north cell is positive. Because, by the time, agent takes a north action, environment changed the dynamic car location from north to north-east and collision didn't take place between the agent and the dynamic car.

3.4. Traffic Light Module

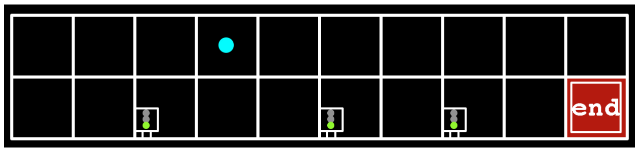

Figure 6: Simulated environment for the traffic light module. Blue dot represent the agent and there are traffic lights along the right side of the lane.

We ran the module for 200 episodes and in each episode we have 3 traffic light which change their status from Red to Yellow to Green after 5 time-stamp.

- Destination reward = 10
- Reward for stopping at red or yellow light = 10
- Penalty for taking action at red or yellow light = -5
- Reward for moving at green light = 10
- Learning rate = 0.5
- Epsilon = 0.1(exploration rate)
- State space = (traffic light at next column)

State	**action**	**Q-Value**
'R'	north	-10.00
'R'	south	-20.30
'R'	east	-13.42
'R'	stop	50.00
'Y'	north	-5.00
'Y'	south	-15.45
'Y'	east	-20.31
'Y'	stop	50.00
'G'	north	33.42
'G'	south	42.4
'G'	east	27.03
'G'	stop	-33.42

Table 2: Q values for the traffic light. The traffic light situation for the next column is considered as state space where R, Y, G represents red, yellow and green light signals. There are other states space, action and Q values combination also but not showing it because of page constraints.

From the table 2, we can see that when we have a red or yellow light, then the Q-Value for all other actions except stop action are negative. Because when the agent took the step while exploring during yellow or red light, environment gave the negative reward. However, when the car chooses to stop while red or yellow light, environment gave the positive reward. However, when the car chooses to stop during green light, it got negative reward because during green light, the agent should move.

3.5. Pedestrian Crossing Module

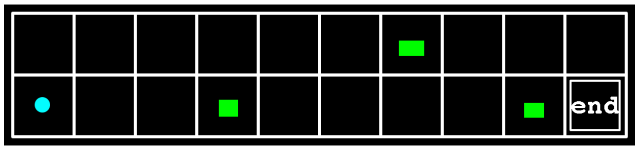

Figure 7: Simulated environment for the pedestrian crossing module. Blue dot represent the agent and green represent the pedestrian crossing the grid. And there are traffic lights also on the right side of the lane.

We ran the module for 200 episodes and in each episode we have 3 pedestrian crossing which traverse in north direction and change their cell after 5 timestamps.

- Destination reward = 10
- Reward for stopping at red or yellow light = 10
- Penalty for taking action at red or yellow light = -5
- Reward for moving at green light = 10
- Learning rate = 0.5
- Epsilon = 0.1(exploration rate)
- State space = (traffic light at next column)

From the table 3, we can see that when we have pedestrian in next column, then the Q-Value for all other actions except stop action are negative. Because when the agent took the step while exploring during pedestrian crossing, environment gave the negative reward. However, when the car chooses to stop while pedestrian crossing, environment gave the positive reward.

3.6. Combined Module

For the combined module, Q values from all the learned modules are used. We have tried different weighting factor and figured out the best one:

- Destination reward = 35
- Learns from individual modules

State	**action**	**Q-Value**
'P'	north	-40.00
'P'	south	-45.30
'P'	east	-63.42
'P'	stop	75.00
'NP'	north	5.00
'NP'	south	15.45
'NP'	east	20.31
'NP'	stop	10.40

Table 3: Q values for the pedestrian crossing. The pedestrian situation for the next column is considered as state space where P represents pedestrian and NP represent no-pedestrian.

- Learning rate = 0.5
- Epsilon = 0.1(exploration rate)
- State space = Neighboring cells(N,S,E,W)
- Module Weights = walk(0.25), static obstacles (0.25), dynamic obstacles (0.2), traffic lights(0.5), pedestrian crossing (0.25)

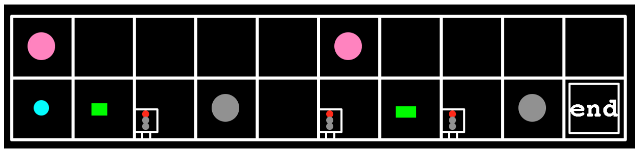

Figure 8: Simulated environment for the self-driving car. Blue dot represent the agent, pink represents the dynamic cars which have a speed of 1 cell per time-stamp.Grey represent the static obstacles. Green represent the pedestrian crossing the grid. And there are traffic lights also on the right side of the lane.

4. Observation

The following are some observations based on the experiment:

1. On a small grid, the number of iterations required by the walking agent to learn is less and that of the obstacle and combination agent is significantly more. This can be explained by the fact that in each episode the walking grid remains the same, where as the obstacle layout keep changing for the other agents.
2. The walking agent takes much longer to learn on long side walk (since its state space is defined by the grid position). Whereas the obstacle and combination

agents learn faster on long side walks because their state space is agnostic of grid position but it depends on the environment condition. And, in long side walk, agent will encounter more obstacles, traffic lights and pedestrian crossings.

3. The learning times of the obstacle and dynamic agent is comparable.

4. The weights of the combined modular agent can change dramatically when the rewards for walking or obstacles is varied. This was quite tricky, and it seemed to take a long time to find an appropriate weight for the combined agent . Normalizing the rewards helps.

5. Conclusion

We develop reinforcement learning agent that can learn driving in the obstacle environment. From the experiments, we can say that driving a car in the obstacle environment can be learn by dividing it into five modules. One module is to learn how to reach the destination of the grid(end of the sidewalk), the other modules learns to avoid static obstacles, dynamic obstacles, traffic light and pedestrian crossing. The final agent that navigates this grid is developed as a combination of these five individual modules. But, the weights of the combined modular agent can change dramatically when the rewards for walking or obstacles is varied. This was quite tricky, and it seemed to take a long time to find an appropriate weight for the combined agent.