Report : Reinforcement Learning

Shmuel Naaman

In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?

The agent eventually make it to the target location, the number of steps that it required increase exponentially as the distance between the start point and the end point increase.

Identify and update state:

The most interesting states is the 'Next waypoint', this state provide with the cab the prefer direction that should be taken if possible.

The next important states are the 'Red-Green light' that restricts the possibilities for the cab. Following the 'Red-Green light' rules will make sure that we do not lose too many points on the way to the destination.

Another important state is the 'oncoming' that fine tune the 'Red-Green light' and provide us some additional possibilities and in some cases might reduce the number of steps needed to arrive at destination. The reason that this parameter is not so important is that not many cars are on the road.

What changes do you notice in the agent's behavior?

As we start to use the Q table the number of steps required to arrive at destination reduce considerably. In addition when you are looking at the cab it seems that it chooses reasonable direction when possible. As the number of runs increase, it seems that also the number step to destination decrease. That is obviously due to the fact that 'next_waypoint' is one of the states, following this state will provide higher reword. As the agent experience more state it learn which behaviors will provide higher reword and we set the direction according to the maximum reword.

Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform?

One change was, when choosing direction, if the self state was all zeros, the agent chooses random state, this is to prevent from the max function to choose similar value for this cases.

The table below depict the changes I made to the parameters Alpha Gamma Epsilon and initial Q. whenever value in table is empty, no change was made. The most significant change that I notice was that high epsilon provide less variable success rate. Yellow color indicate success rate lower than 5 we can see that this models have mostly low alpha. Gray colors correspond to success rate higher than 10 and lower than 16. It seems that they are mostly related with high initial Q.

Alpha	*Gamma*	*Epsilon*	*Q*	*Success*	*Count*
0.5	*0.1*	*0.1*	*0*	*17 / 20*	*583.5*
0.1				*4 / 20*	*443.5*
0.5	*0.5*	*0.5*	*12*	*12/20*	*551*
		0.2		*11/20*	*586.5*
			0	*16/20*	*618*
		0.1		*10/20*	*654*
		0.5/(1+ N)		*14 / 20*	*641.0*
			12	*15/20*	*587*
0.1			*0*	*3/20*	*600*
0.9				*15/20*	*641*
0.5				*17/20*	*629*
	0.1			*8/20*	*770*
	0.9			*6/20*	*600*
	0.5	*0.9/(1+ N)*		*17/20*	*600*
			12	*13/20*	*610*
	0.7		*0*	*17/20*	*650*
0.7				*10/20*	*580*
0.3	*0.7*	*0.9/(1+ N)*	*0*	*18/20*	*580*

N – number of runs, one run correspond to the start till the arrival of the cab to destination

After I did all that calculations and had a glance on the Q-table. I realized that a great contribution to the algorithm will be: each time we get 'all zeros' states we will use the state that was learned by the 'no one in the cross road' state. In most cases that will be the best solution.

Alpha	*Gamma*	*Epsilon*	*Q*	*Success*	*Count*
0.3	*0.7*	*0.9/(1+ N)*	*0*	*19/ 20*	*565*

Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties?

Ideal or optimal policy would be

The problem that I see is the local minimum, for the state green light and next way point is right, the cab stay in place. I could not think of a good way to get out of this minimum.

But the agent performs well

As we can see in the Q table here below (Q table for 100 runs, success 84/100), on green light the cab follows the next_waypoint, (except for right turn). On red the cab STOPS unless s it is right turn.

State:

('Light' , 'next_waypoint', 'oncoming')

We can see that when the light is red the cab stop unless it is right turn.

('red', 'left', None): {None: 3.3332, 'forward': 0.3414, 'right': 0.8399, 'left': 0.1667}, STOP

('red', 'left', 'left'): {None: 0.8214, 'forward': 0, 'right': 0, 'left': 0}, STOP

('red', 'left', 'right'): {None: 0.573, 'forward': 0, 'right': 0, 'left': 0}, STOP

('red', 'forward', None): {None: 3.3333, 'forward': 0.7614, 'right': 2.7976, 'left': 0.8830}, STOP

('red', 'forward', 'left'): {None: 2.1521, 'forward': 0, 'right': 0, 'left': -0.17967}, STOP

('red', 'forward', 'right'): {None: 0.8214, 'forward': 0, 'right': 0, 'left': 0}, STOP

('red', 'forward', 'forward'): {None: 1.4404, 'forward': 0, 'right': 0, 'left': 0}, STOP

('red', 'right', None): {None: 2.220, 'forward': 0.8089, 'right': 6.0346, 'left':1.8223},

('red', 'right', 'left'): {None: 0, 'forward': 0, 'right': 1.299, 'left': 0},

('red', 'right', 'forward'): {None: 0, 'forward': 0, 'right': 3.455, 'left': 0},

When green, the next waypoint is chosen unless it is the right turn.

{('green', 'left', None): {None: 0, 'forward': 0, 'right': 0.7772, 'left': 5.7501},

('green', 'left', 'left'): {None: 0, 'forward': 0, 'right': 0, 'left': 1.299},

('green', 'forward', None): {None: 4.7319, 'forward': 8.264, 'right': 1.24073, 'left': 3.966},

('green', 'forward', 'right'): {None: 0, 'forward': 2.0023, 'right': 0, 'left': 0},

('green', 'forward', 'forward'): {None: 0, 'forward': 1.2999, 'right': 0, 'left': 0},

('green', 'forward', 'left'): {None: 0, 'forward': 6.5477, 'right': 0, 'left': 0},

('green', 'right', None): {None: 3.33, 'forward': 0.548, 'right': 1.782'left': 0.3225},this might be local minimum

('green', 'right', 'left'): {None: 0.3, 'forward': 0, 'right': 0, 'left': 0}, this might be local minimum

('green', 'right', 'forward'): {None: 0.3, 'forward': 0, 'right': 0, 'left': 0},

The cab need more experience.

('red', None, 'left'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('red', None, 'forward'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('green', None, None): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('green', 'left', 'forward'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('green', 'left', 'right'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('red', None, None): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('red', 'left', 'forward'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('green', None, 'forward'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('green', 'right', 'right'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('red', 'right', 'right'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('red', None, 'right'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('green', None, 'right'): {None: 0, 'forward': 0, 'right': 0, 'left': 0},

('green', None, 'left'): {None: 0, 'forward': 0, 'right': 0, 'left': 0}}