Reinforcement Learning applied to Smart Vehicles

Lakshmi Venugopal
Machine Learning with Networks
Texas A&M University, College Station, TX – 77840

***Abstract* – This paper presents reinforcement technique to train a vehicle to follow US traffic rules on its route to destination. A vehicle which doesn't have any prior knowledge about the driving rules and safety is allowed to learn the rules based on a typical real-world rules-based environment. Towards this objective the following tasks are performed: 1. Simulate a learning environment and train a learning agent using Q-learning technique. 2. Validate the evolved Q-table for its conformance with traffic/environment rules under current design. 3. Perform a parametric study/tuning to improve the learning efficiency. The learning profile is found to be sensitive to the learning environment. Effects of learning rate and discount factor on Q-values are finally revealed.**

Keywords – Machine learning, Reinforcement learning, Q-learning, smart cab, Q-table.

I. INTRODUCTION

Machine learning algorithms have recently gained much popularity due to the rapid increase in computational resources. There are mainly three types of machine learning algorithms: supervised learning, unsupervised learning and reinforcement learning. In supervised learning, the target variables are known and a model is built to map the input variables to the target. In contrast, the target variables are unknown in unsupervised learning. Therefore, the data is divided into different groups based on a specific set of properties/features. Reinforcement learning uses trial and error method to train to its environment and uses the past experience to take the best possible action.

In this work, Q-learning, a model-free reinforcement learning technique is used to train an alien vehicle to a smart cab. This technique evolves a Q-table containing potential Q-values for all the actions corresponding to any given state. Various applications of Q-learning include chess gambits, trading strategies, teaching robots and many more. A wide variety of games are efficiently taught to the machine using Q-learning technique. Such promising games trained with this technique are shown in Figure 1.

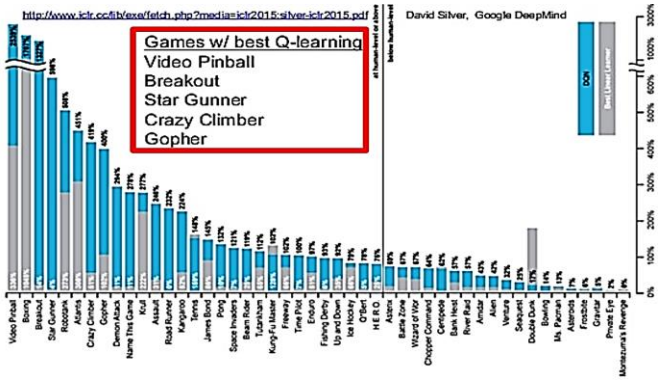
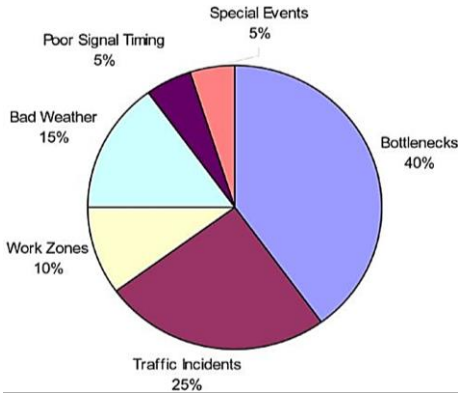
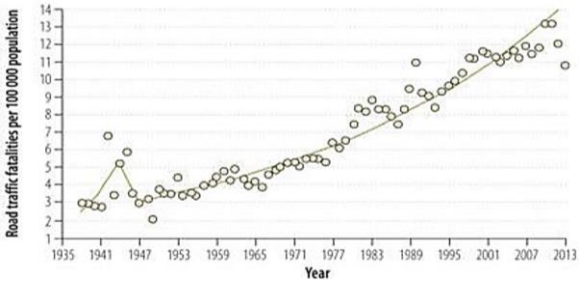


Fig. 1. Real-time practical applications : Games with best Q-learning

II. MOTIVATION



(a)



(b)

Fig. 2. (a) Sources of traffic congestion, (b) Graph showing annual increasing fatalities

Around 65 percent of the traffic congestions constitute bottlenecks and traffic incidents (Figure 2 (a)). Moreover, it can also be observed from Figure 2 (b) that there is an annual increase in the number of road traffic fatalities. One could easily foresee the advantages if Q-learning can be used for auto-piloting every vehicle. Few immediate outcomes include prevention of fatal accidents, significant decrease in traffic delays and replacement of manpower with much systematic road transportation network. This approach also has the potential to provide feedback to engineers for further improvement in the transportation system by predicting construction schemes and traffic rules that benefit the transportation system. It is therefore not surprising to see that some auto-piloted vehicles are already in practice (google/baidu smart cab), while others are yet in the research and development stage.

III. METHODOLOGY

The objective of this work is to train a vehicle to reach any given destination from a random starting location by carefully obeying the US traffic rules using the Q-learning technique. Towards this objective, we achieve three main tasks in this paper. Firstly, a learning environment is simulated and a Q-learning algorithm is applied to train the smart car. Secondly, the final Q-values are validated to check if the smart car has learnt the correct actions for all the possible states. Thirdly, a parametric study is carried out to improving the learning efficiency through appropriate parameter tuning. The effect of environmental parameters, the effect of learning rate and the effect of discount rate on the evolution of Q-values are studied.

A. Learning environment & different states

A learning environment is created with predefined set of rules. In our case, the environment consists of Manhattan type roads with traffic signal lights attached with each intersection. Client-like objects that follow the environmental rules are also created. For simplicity, these objects are called 'clients' in the current work. An alien car with no prior knowledge is then allowed to learn and adapt to the environment by analyzing the behavior of client cars. As mentioned before, the Q-learning technique will be used for learning purpose. In other words, the client cars must be simulated and that is the sole creator of input data that will in turn be used to train the alien car.

A typical environment consists of *m* x *n* Manhattan type roads (Figure 3) with the boundaries assumed to be periodic, meaning that a car which is in the extreme right end moving towards east direction will end up coming from extreme left end heading towards east. An instance of the environment simulated using 'pygame' is illustrated in Figure 3. The states of the traffic lights are represented as vertical or horizontal green lines at each intersection. A vertical line implies that the north-south traffic is open (east-west traffic closed) whereas a horizontal line indicates that the east-west traffic is open (north-south traffic closed). Different client cars (non-red cars) can also be seen in Figure 3. These client vehicles travel across different roads according to the traffic rules and regulations defined by the environment. As a whole, the environment has roads, intersections, traffic lights and randomly (but according to environmental rules) moving client vehicles; with the boundaries assumed to be periodic. Now we generate a smart car (red car) that initially has no prior knowledge about the environment policies. The smart car is now asked to start from a source to a destination (marked as red dot) which is randomly selected. The minimum distance between the source and the destination is chosen to be at least 4 units to allow efficient training.

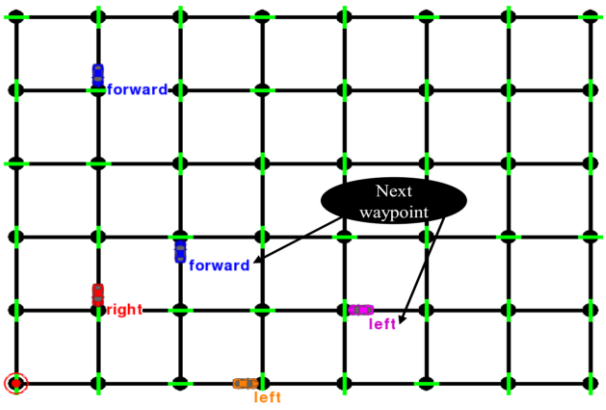


Fig. 3. An instance of environment simulated using pygame (6x8 Manhattan roads)

As observed in Figure 4, the number of states for the smart vehicle at an intersection is determined by the following factors: (a) the direction that the car needs to go in order to reach the destination, (b) the traffic signal at that intersection and (c) the heading direction of other cars at that intersection. The states of the smart car thus include possible traffic light color (red or green), the next direction the smart vehicle should choose to head (forward, left or right), the possible heading directions for the oncoming traffic (forward, left, right, none) and the possible heading directions for cars coming from the left (same as oncoming). Therefore, the total number of states for the learning car = 2x3x4x4 = 96. It is

important to note that cars coming from the right are ignored as all the states are already covered by the traffic signal rules. The car on the right will head forward or to its left only if they see a green signal, which means that the signal for the learning agent is red (hence learning agent has to halt). If the car heading from right had to take a right turn on red, it would have to yield to the learning cab. Hence, the actions of the car heading from the right are superfluous.

Fig. 4. Instance of traffic intersection

B. Q-learning algorithm

As discussed before, there are 96 states for the smart car. At each state, there are 4 possible actions for the smart car. The four possible actions are 'none', 'forward', 'right' and 'left'. When the smart car encounters a state, the car gets rewards for each of the actions. Our main aim is to maximize the net reward. Therefore, the optimal action in a given state would be the action with the highest net reward. A state-action combination is thus mapped to a function, Q. The Q values are updated according to the following equation:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \,.\{r_{t+1} + \gamma \,.\max_a Q(s_{t+1}, a) - Q(s_t, a_t)\},$$

where $Q(s_t, a_t)$ is the old Q-value, α is the learning rate, r_{t+1} is the rewards assigned, γ is the discount factor and $\max_a Q(s_{t+1}, a)$ is the estimate of optimal future Q-value. The rewards for each action of the smart car are shown in Table 1.

Learning rate determines the extent of using new estimate over the old one. A learning rate of 'one' takes into account only the new values and neglects the previous estimates. Generally, a learning rate value closer to 'one' trains the learning agent faster but can exhibit unstable Q-values. Discount factor determines the importance that is given to future rewards. A discount factor of 'zero' considers only immediate rewards whereas a discount factor close to 'one' gives more weights to the future rewards. These parameters can be highly sensitive to the learning environment under consideration. Therefore, it is critical to conduct a parametric study and hence tune the parameters according to the corresponding environment.

TABLE 1: DISTRIBUTION OF REWARD POINTS

Actions of smart car	Rewards
Obeys traffic rules & stationary	0
Obeys traffic rules & moves towards destination	2
Obeys traffic rules & moves away from destination	-0.5
Violates traffic signal	-1
Reaches destination within deadline	10

Given a state 'x', action with maximum Q of all 'x'-action combinations is performed. Also, it can be carefully observed that the smart car learns indirectly from the behavior of client cars. For instance, a collision can occur due to an incorrect move taken by the smart car, where the sole contributor is the client car (which obeys traffic rules). Such an act also modifies the Q-value because of the assigned negative reward. This is how client cars are bridged with the smart car. One can see this whole play as an alien car turning into a smart car by watching the behavior of the client cars.

IV. RESULTS

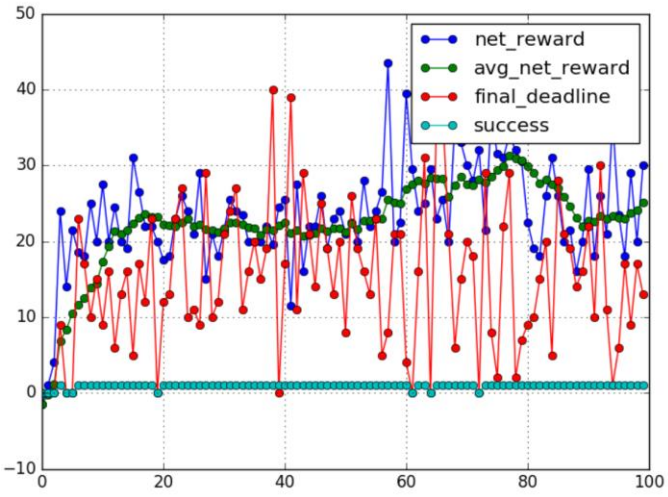


Fig. 5. Statistics of relevant global parameters

Figure 5 depicts the statistic evolution of major global variables in 100 iterations/steps. Each step is a trial of learning agent trying to reach its destination in the environmental constraints. In Figure 5, it is observed that the average net reward increases which means that the car is learning to adapt to its environment in each step. It is also noted that the frequency of successes is more. Therefore, the car reaches the destination within the deadline without any traffic violation or collision for most of the times. Deadline is defined as five times the minimum distance between the start and the destination.

A. Validation of Q table

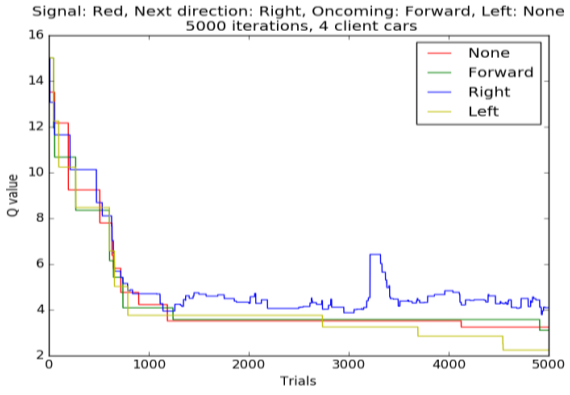
Typical Q values after 5000 iterations with 24 client cars are shown in Table 2. The maximum Q value for each state is marked as red and the corresponding action is taken (column 3). Therefore, given a state for the smart car, the optimal action is chosen (action with maximum Q value). For instance when the car needs to head towards right, the traffic signal is red, car on the left of smart car is moving in its forward direction and there is no oncoming car, then the smart car should remain in its position and yield to the client car to avoid accidents. Therefore, the Q table is consistent with the traffic rules.

TABLE 2: TYPICAL Q VALUES AFTER 5000 ITERATIONS WITH 24 CLIENT CARS

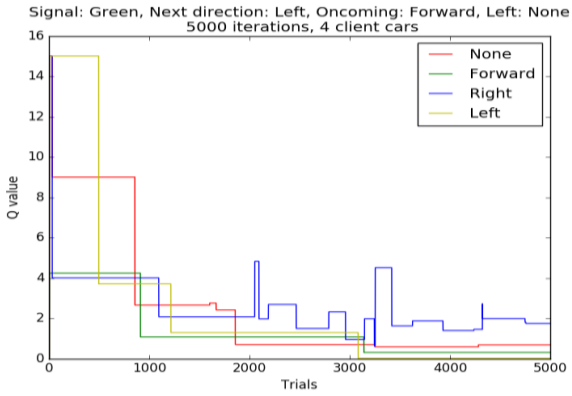
States (Next direction, Signal, Oncoming, Left)	**Q values (None, Forward, Right, Left)**	**Action taken**
Right, red, none, forward	1.84, 0.46, 0.20, 0.47	None
Left, green, forward, left	3.66, 1.06, 0.73, 3.38	None
Left, green, none, forward	0.9, 1.57, 0.72, 2.5	Left
Forward, green, forward, left	2.5, 3.5, 0.51, 1.87	Forward
Right, red, forward, left	1.95, 3.58, 3.64, 1.5	Right

B. Effect of number of client cars

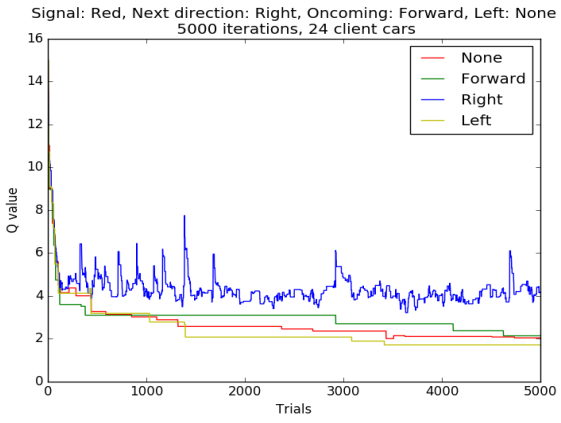
Figures 6(a) and 6(b) shows two different states of the smart car picturizing the evolution of Q values for each action in 5000 iterations with 4 client cars. In Figure 6(a), when the smart car encounters the state when the traffic light is red, the next direction that the car needs to head towards is right, the oncoming car is moving in its forward direction and there is no car coming from the left, the optimal action is taken (action with maximum Q value is chosen which is 'right'). But in Figure 6(b), when the smart car is in another state where the traffic light is green, the next direction that the car needs to head towards is left, the oncoming car is moving in its forward direction and there is no car coming from the left, the smart car takes wrong action. This is because with 4 client cars, only partial learning is achieved. It is noted that all 96 states were not learnt and only 60 out of 96 states were learnt. When the number of cars is increased to 24, all 96 states were learnt. This is quite evident in Figures 6(c) and 6(d) as with 24 client cars after 5000 iterations, optimal actions were taken in both cases.



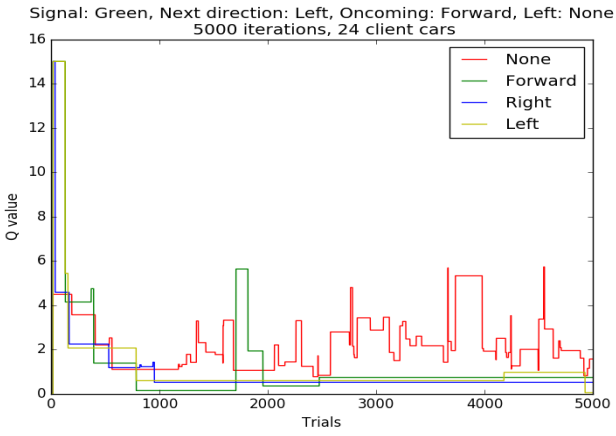
(a)



(b)



(c)



(d)

Fig. 6. Q values for two different states of smart car after 5000 iterations. (a) and (b) With 4 client cars. (c) and (d) With 24 client cars.

C. *Effect of learning rate*

Figure 7 illustrates the evolution of maximum and minimum Q values for a given state of the smart car in 5000 iterations with different learning rates of 0.2 and 0.8. It is observed from the figure that if the learning rate is high, the Q values converge faster but fluctuations are observed. This can lead to bad decision making as Qmax can cross Qmin. In contrast, if the learning rate is low, the Q values take more iterations to converge, but it will lead to correct action. Hence, a low learning rate with larger number of iterations is desirable for accurate results.

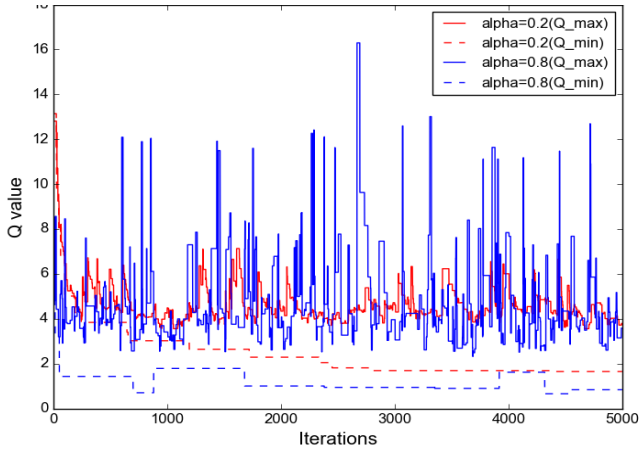
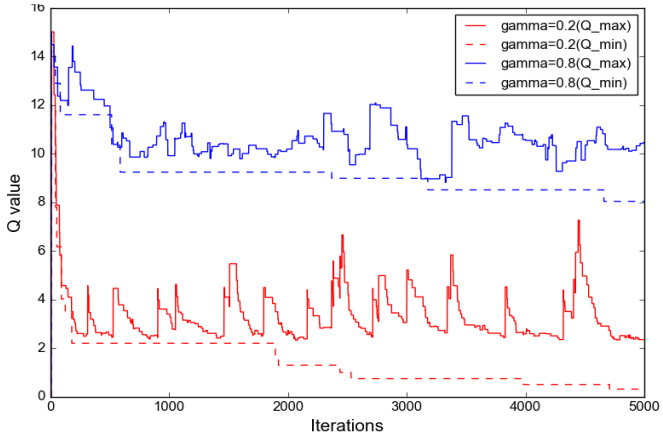


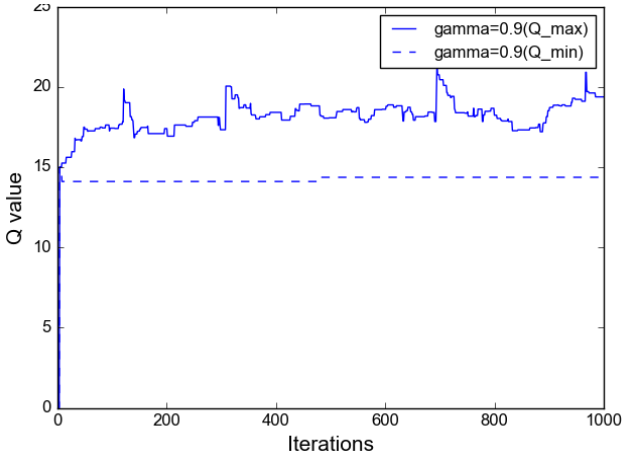
Fig. 7. Effect of learning rate with 24 client cars (gamma=0.5)
State of smart car -> Signal: Red, Next direction towards destination: Right, Oncoming traffic: Forward, Left traffic: None

D. *Effect of discount factor*

Figure 8 picturizes the evolution of maximum and minimum Q values for a given state of the smart car in 5000 iterations with different discount factors of 0.2 and 0.8. From Figure 8(a), it is observed that the discount factor does not affect the Q table as there is a clear separation between the maximum and minimum Q values for both gamma values. However, if the value of gamma is kept very close to 1 (see Figure 8(b) gamma = 0.9), the Q values will not converge as the Q values will sum up to infinity according to our equation.



(a)



(b)

Fig. 8. Effect of discount factor with 24 client cars (alpha=0.2)
(a) With gamma = 0.2 and 0.8 (b) With gamma = 0.9
State of smart car -> Signal: Red, Next direction towards destination: Right, Oncoming traffic: Forward, Left traffic: None

V. CONCLUSION

The environment is simulated successfully and client cars were created that obey the environmental rules, providing input data for new alien cars. Q-learning is implemented to train the learning agent. The final Q-values were found to be consistent with the required environmental rules. The effect of number of client cars, learning rate and discount factor was studied.

With large number of iterations and client cars, desirable target is achieved. Learning rate can be tuned to a low value for accurate results. With less number of iterations, the results are not accurate as it does not converge to correct values. Even with more number of iterations, all 96 states are learnt only with larger number of client cars, which implies client cars have a large impact in the learning process of the smart car. The real world problems are even more complex (4 lane roads, other traffic rules). More complex simulations need to be done and fine parametric tunings should be performed to achieve acceptable learning of such environments.

VI. REFERENCES

[1] https://www.udacity.com/course/reinforcement-learning--ud600

[2] https://github.com/sidharths/UMLND/tree/master/smartcab

[3] Google Deep Mind *(Figure 1)*

[4] Federal Highway Administration Research and Technology & WHO *(Figure 2)*

[5] Oscium *(Figure 4)*