Reinforcement Learning Coursework 2

Wenzhe,Liu s1631854

March 28, 2017

1 Actor-Critic Architecture

(1)Actor-Critic methods are temporal difference methods, which is the combination of Policy Gradient(Actor) and Function Approximation(Critic). Actor chooses the action based on probability distribution, while critic decide the score based on the action chosen by actor. Meanwhile, actor changes the probability of choosing the action according to the score from critic.
Specifically, critic is a state-value function, which evaluates the new state to determine whether the action makes things better or worse. Critic uses TD error to give feedback to the action:

$$\delta_t = r_t + \gamma V(S_{t+1}) - V(S_t), \tag{1}$$

where V is the current value function implemented by the critic.
So if the TD error is positive, actor will enhance the probability to choose this action and vice verse.
Assume that the actions are generated by the Gibbs softmax method:

$$\pi_t(s,a) = \Pr\Big(a_t = a | s_t = s\Big) = \frac{e^{p}(s,a)}{\sum_b e^{p}(s,b)}, \tag{2}$$

where p(s,a) is the probability value, which shows the tendency to choose this action. Based on the feedback δ_t from critic, p(s,a) can be changed by:

$$p(s_t,a_t) \leftarrow p(s_t,a_t) + \beta\delta_t, \tag{3}$$

where β is another positive step-size parameter.
The advantage of Actor-Critic method is the update speed. Comparing with the traditional policy gradient, it is faster because it updates in every singe step. However, the disadvantage of Actor-Critic method is that it converges very hard.
(2)Actor-Critic method is suitable for working in continuous space with an infinite number of possible actions because it can reduce the dimensionality of the problem as opposed to other temporal difference (TD) learning architectures. A successful application of Actor-Critic method is the Walking Control of a 5-Link Bipedal Robot[1], which has a quite large and continuous state-action space. Actor-Critic method is useful in this case because it reduces computation complexity. It does not need to store so many state-action space information.
(3)Sarsa algorithm and Actor-Critic method are both the on-policy methods. Sarsa is the value-based method, which can be viewed as critic, one part of Actor-Critic method.

2 RL with Function Approximation

2.1 linear approximation

The linear approximation of the Q(s,a) state-action value function at time t is given by:

$$Q(s,a) = \Theta_t^T \Phi_{s,a} = \sum_{n=1}^{n} \theta_t^i \phi_{s,a}^i, \tag{4}$$

where θ_t^i and $\phi_{s,a}^i$ denote the i^{th} component of the corresponding n dim vector. A set of features is defined as $\phi_s = (\phi_s(1), \phi_s(2).....\phi_s(n))^T$ to be used as the representation of states. θ_t has the same size as ϕ_s. In order to reproduce Q-table, we need to represent Q value function by action and state:

$$Q(s,a) = \theta_0 + \theta_1 \phi_{s,a}^1 + \theta_2 \phi_{s,a}^2 + ... + \theta_n \phi_{s,a}^n, \tag{5}$$

where $\phi_{s,a}^n$ is a boolean value. If the value is 1, it denotes the agent is in this state s with action a and vise verse.

2.2 Implement a reinforcement learning agent

1.Collisions should be avoided
(1) Using sensing of cars to get the number of opponents, there is no opponent in the road and the agent takes the accelerate action, the feature value is 1, otherwise is 0.
(2) Using sensing of cars to get the size and location of each car, apply the Pythagorean theorem to calculate the distance between the nearest opponent and the agent. If the distance is more than the dangerous distance and the agent takes the accelerate action, the feature value is 1, otherwise is 0.
(3) Using sensing of cars to get the size and location of each car, define a safe area, which is the size of 2w*2h , where w,h is the width,height of the agent. If after taking actions, there is no opponent in this safe area, the feature value is 1, otherwise is 0.
2.Moving faster results in passing by more cars
(1) Using sensing of speed to get the speed of current state and the speed of next state, if the speed of next state is faster than the speed of current state, the feature value is 1, otherwise is 0.
(2) Using sensing of speed to get the speed of next state, if the speed is less than 0, which means the agent is slower than the opponent, and the agent takes the accelerate action, the feature value is 1, otherwise is 0.
(3)Using the sensing of the speed, the feature value is normalized speed.
3.Staying in the center of the road is preferred when possible
(1) Using sensing of grid to get the location of the agent, if the column of the agent is 4 or 5 and the agent takes the accelerate action, the feature value is 1, otherwise is 0.
(2) Using the sensing of grid to get the location of the agent, if the column of the agent is less than 4 and the agent takes the right action, the feature value is 1, otherwise is 0.
(3) Using the sensing of grid to get the location of the agent, if the column of the agent is more than 5 and the agent takes the left action, the feature value is 1, otherwise is 0.

3 Performance

3.1 Learning curve

The parameter setting of function approximation agent is almost the same as the provided Q-learning agent, such as epsilon. The overall performance of function approximation agent is not as good as the Q-learning agent. For example, the average total reward of function approximation agent is less than the Q-learning agent. The function approximation agent starts to increase its total reward at epoch 450, while the Q-learning agent increases its total reward at epoch 120. So the increasing rate of function approximation agent is slower than the Q-learning agent. Because the features of function approximation agent are too few, which cannot express the state as successful as the Q-learning agent, the performance of function approximation agent is worse than the Q-learning agent.

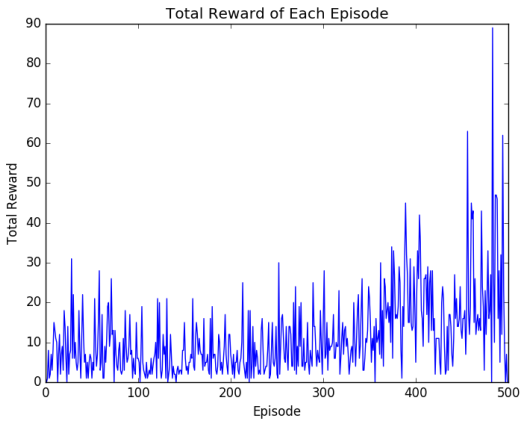

Figure 1: The learning curve of function approximation agent.

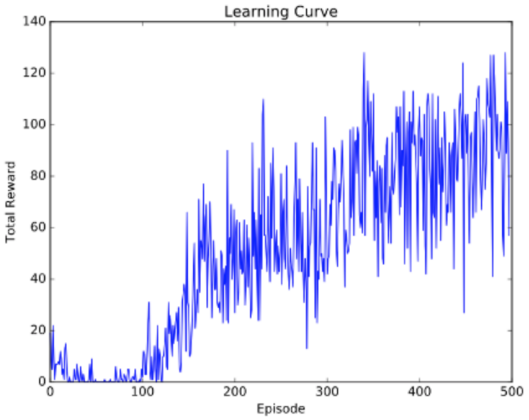

Figure 2: The learning curve of Q learning agent.

3.2 Usefulness of each feature

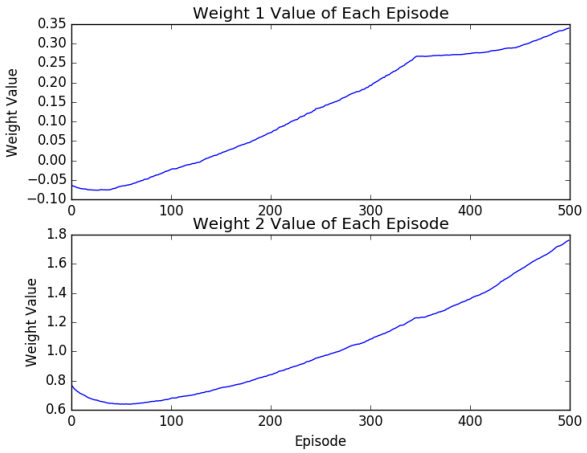

Figure 3: The weight of feature 1 and 2 .

From 3 we can see the weights of feature 1 and feature 2 change with the increase of episode. Feature 1 represents that the agent driving in the center of the road and taking accelerate action will get value 1. The weight of feature 1 increases from -0.05 to 0.35, which means the importance of this feature goes up gradually, but the value of this weight is not very high. So we can know the feature is useful but not very important. Besides, feature 2 represents that no collision and the agent taking acceleration action will get value 1.The weight of feature 2 increases from 0.8 to 1.8, which means this feature is important. Also the value of this weight is large comparing with others, which shows that this feature plays a leading role in choosing the action.

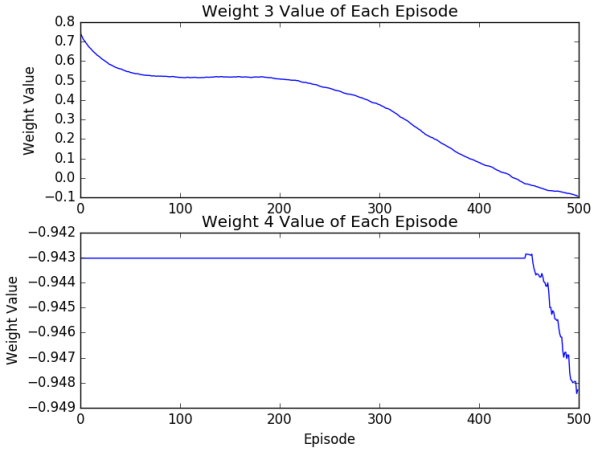

Figure 4: The weight of feature 3 and 4 .

From 4 we can observe that the weight of feature 3 decreases with the increase of episode. Feature 3 represents that the speed of next state faster than the speed of current state will receive value 1. But from the curve we can know this feature is not useful because the weight decreases from 0.7 to -0.1. The reason why this feature is not useful is this feature cannot distinguish between different actions. More specifically, every time these four actions will receive the same reward, either 1 or 0. Feature 4 means no opponent in the road and the agent taking acceleration action will receive value 1.From the curve we can know the weight kept constant in the first 450 epoch and decreases slightly from epoch 450 to 500, which shows that this feature is not very important and useful as well. Because this situation, no opponent in the road, happens rarely, feature 4 works seldom.

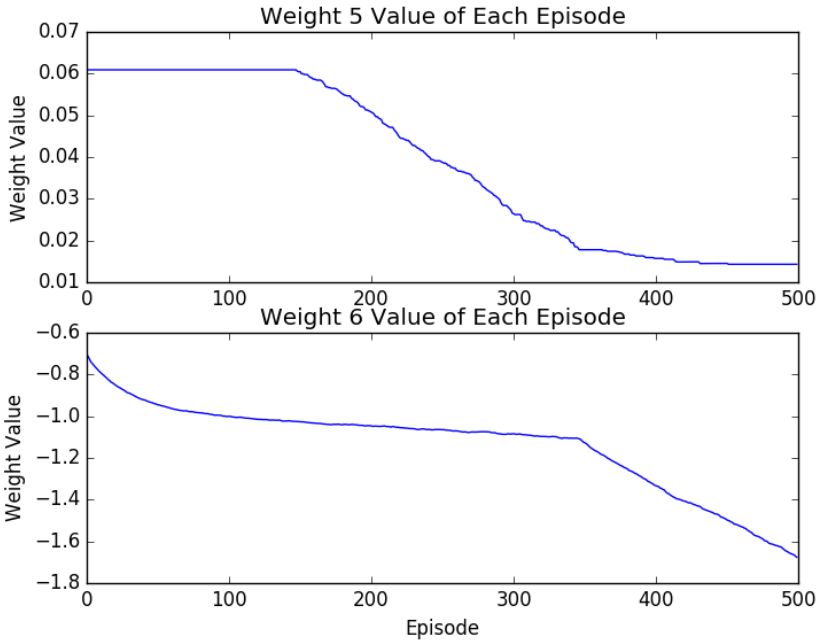

Figure 5: The weight of feature 5 and 6.

From figure 5 we can know the weight of feature 5 kept constant at first and then decreased from 0.06 to 0.02. Feature 5 represents that agent taking the left or right action to avoid collide with the opponent in front of it. This feature is not useful because the condition of this feature is not very strict. Only depending on the agent is at the same column with the opponent to decide the collision is not enough. So we can see the weight decreases and the value is not very high. Feature 6 shows almost the same trend as the feature 5. Feature 6 represents the agent taking the action that probably hits the wall will receive 0. But most of the time, the agent drives the car in the middle of the road, so the return value is 1. That is why we can see the weight is negative correlation.

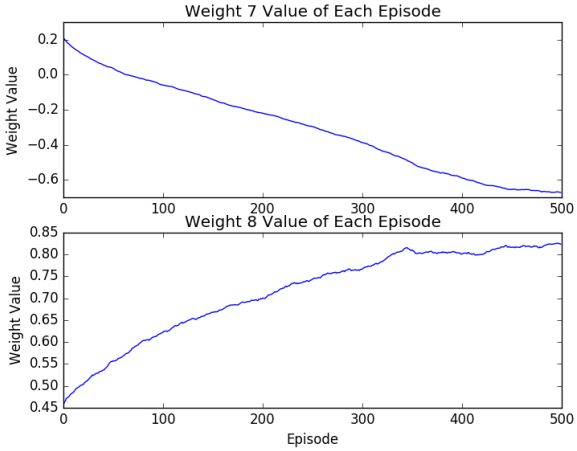

Figure 6: The weight of feature 7 and 8 .

From figure 6 we can notice that the weight of feature 7 decreases gradually, which means this feature is not very useful. Feature 7 represents if the action takes the agent into the safe area , the value is 1. But I think the definition of safe area, there is no opponent in the area 2w*2h, is too strict. This situation happens rarely, so the feature is not very helpful. However, the weight of feature 8 increases first and then converges at episode 400. Feature 8 represents if the agent is not in the center of the road and the agent takes the correct action to move the car to the center, the value is 1. We can see this feature plays a leading role in choosing the action. On the one hand , the weight is large comparing with others. On the other hand, this feature guarantees the agent in the center of the road.

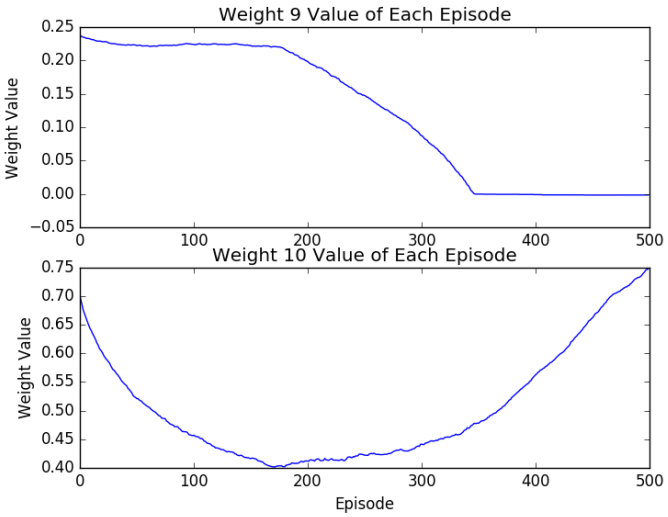

Figure 7: The weight of feature 9 and 10.

From figure 7 we can notice that the weight of feature 9 keeps constant first and then decreases ,but finally it converges to 0, which shows that this feature is not very useful and important. Feature 9 returns the normalized speed of the agent. No matter which action the agent takes, th return value is the same. So it means this feature cannot distinguish between different actions. Besides, the weight of feature 10 decreases first and then increases. It means this feature is useful sometimes. Feature 10 represents if the speed is less than 0 and the agent takes accelerate action, the value is 1. The speed is less than 0, which means the agent is slower than the opponent, so we encourage the agent to accelerate.

3.3 Convergence rate

From figure 1 and figure 2 we can see that the Q-learning agent starts to converge at episode 400, but the function approximation agent does not converge even it runs for 500 episode. Intuitively, the Q-learning agent converges faster than the function approximation agent. But from figure 1 we can observe that the function approximation agent shows the trend to converge. Because of the time limitation, the agent does not have enough time to train. Assume that the function approximation agent converges at λ. According to the formula:

$$\lim_{n\to\infty} \frac{|x_{n+1} - r|}{|x_n - r|^\alpha} = \lambda, \tag{6}$$

where α is the rate of convergence, r is the value that the function converges to and $\alpha >= 1, \lambda > 0$, using Taylor's theorem to estimate x_{n+1} and x_n, the rate of convergence can be estimated as:

$$\alpha \approx \frac{\log \frac{|x_{n+1}-x_n|}{|x_n-x_{n-1}|}}{\log \frac{|x_n-x_{n-1}|}{|x_{n-1}-x_{n-2}|}}. \tag{7}$$

References

[1] Vaghei Y, Ghanbari A, Noorani S M R S. Actor-critic neural network reinforcement learning for walking control of a 5-link bipedal robot[C]// Second Rsi/ism International Conference on Robotics and Mechatronics. IEEE, 2014:773-778.