# Q Learning	start time:

Before you start, share this document with your team member(s) and then complete the form
below to assign the role of speaker.

Team Role	Team Member
Speaker: shares your team’s ideas with the class.	Arogya, Shreeya, Tim

(30 min) A. Q Learning	start time:

Review the TD Learning algorithm.

$$V^{\pi}(s) = (1 - \alpha)V^{\pi}(s) + (\alpha)\, sample \quad (1)$$

V(s) is the current value and sample is the newly estimated value. We use learning rate alpha to balance the weighted average of current and new values together as the next estimated value at s.

At each time step, an agent takes an action π(s) from a state s, transitions to a state s′, and receives a reward R(s,π(s),s′). We can obtain a sample value by summing the received reward with the discounted current value of s′ under π:

sample = R(s,π(s),s′) + γ $V^{\pi}(s')$

Substitute the sample into equation (1) and reorganize the terms, we have

$$\begin{aligned} V^{\pi}(s) &= (1 - \alpha)V^{\pi}(s) + (\alpha)\, sample \\ &= (1 - \alpha)V^{\pi}(s) + (\alpha)\,(R(s,a,s') + \gamma * V^{\pi}(s')) \\ &= V^{\pi}(s) + \alpha * (R(s,a,s') + \gamma * V^{\pi}(s') - V^{\pi}(s)) \end{aligned}$$

In Q learning, we make two modifications in above formula:

1. We estimate a Q table instead of the utility table. A Q table is a dictionary from state-action pair (s,a) to the Q value, which is the estimated utility collected after taking action a from state s, until the environment is terminated.

2. We replace V(s′) by the maximum Q value obtained over all actions a′ at the next state s′.

Then we have

$$\begin{aligned} Q(s, a) &= (1 - \alpha)Q(s,a) + \alpha * sample \\ &= (1 - \alpha)Q(s,a) + \alpha\,(R(s,a,s') + \gamma * max_{a'}\, Q(s', a')) \\ &= Q(s,a) + \alpha\,(R(s,a,s') + \gamma * max_{a'}\, Q(s', a') - Q(s,a)) \end{aligned}$$

1. What is Q(s,a)?
    - total estimated reward for current state after taking action a and forward

2. How is s’ obtained from s?
    - We don’t know because there is a probability involved which determines the s’

3. How is the action decided at state s?
    - action with highest Q value

4. What is Q(s’, a’)?
    - Q-state of the successor of state s and the next action | this gives an estimated reward for all the path going from s’ and a’

5. What is max Q(s’, a’)?
    - Maximum Q-value of all possible actions on s’

6. Explain the meaning of alpha value. What does it mean if alpha is too small (near 0) or too big (near 1)?
    - 0 means disregarding the new sample value, and 1 means disregarding the old value.

7. Does Q learning require a policy?
    - No, we look into all possible actions instead of relying on a specified policy.

8. If there are 10 states and 4 actions at each state, how large is the Q table? What is the key and what is the value of this table if it is saved as a dictionary?
    - The Q table will be 40 rows large. The key will be a pair of (state, action), and the value is the estimation of the Q value for this pair

9. How to initialize the Q table?
    - initialize as 0

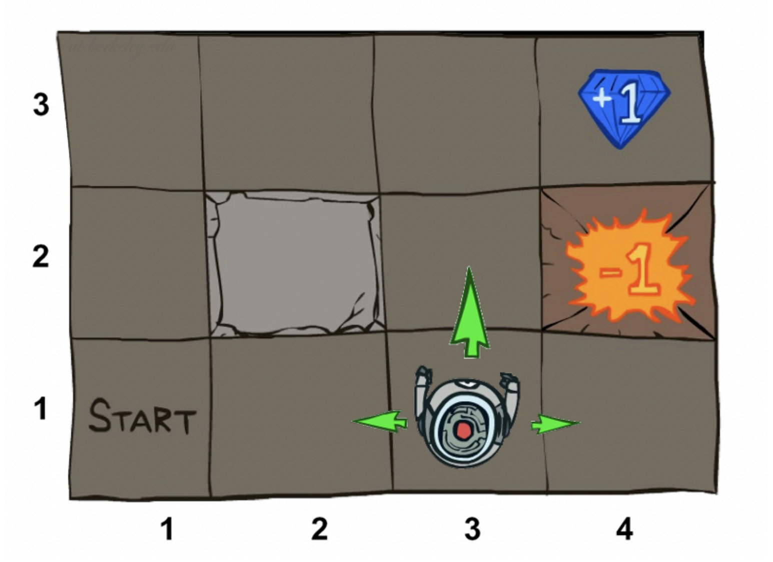
10. How to update the Q table?
    - play the game, and update the table for every move using the formula
    - $Q(s,a) \ + \ \alpha \, (R(s,a,s') \ + \ \gamma * max_{a'} \, Q(s', \, a') \ - \ Q(s,a))$
    - We stop when the Q table is stable and the values are very similar

11. After having the final Q table, how to generate a policy?
    - We look at the highest Q value for a state-action pair and choose the best action (policy) for a state s

(30 min) B. Q Learning Example	start time:

Recall the GridWorld we have examined, where there are only two terminal states: state (4, 3) and state (4, 2), with rewards 1 and -1. All other transitions between non-terminal states have a -0.04 reward (penalty).

In the following grid, for each state, there are five possible actions (N, S, E, W, and exit). Each state has 80% chance to move following the planned action and 10% chance to move following the perpendicular direction against the planned action. If it hits a wall, it does not move.



12. Begin by initializing all values to 0 and fill in three more rows:

State-Action	Q-value
(1,1), North	0
(1,1), South	0
(1,1), East	0
(1,1), West	0
(1,2), South	0
(1,2), North	0
(1,2), East	0
(1,2), West	0

13. Next, using α=0.1, gamma=0.95, and a first trial of:

$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$$

North North East West East North South

Update Q values on the table:

$$Q(s,a) \; + \; \alpha \, (R(s,a,s') \; + \; \gamma \, * \, max_{a'} \, Q(s', \, a') \; - \; Q(s,a))$$

State-Action	Q-value
(1, 1), North	0 + (0.1*(-0.04 + (0.95 * (0)))) = -0.004
(1, 2), North	0 + (0.1 *(-0.04 + (0.95* (0)))) = -0.004
(1, 3), East	0 + (0.1*(-0.04 + (0.95 * (0)))) = -0.004
(1, 2), West	0 + (0.1*(-0.04 + (0.95 * (0)))) = -0.004

14. Again, using α=0.01, calculate the values after a second trial:

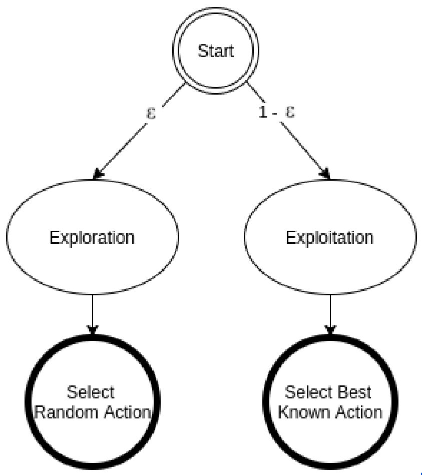
$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (2,1)_{\mathbf{-.04}} \rightsquigarrow (3,1)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (4,2)_{\mathbf{-1}} \, .$$

East North East South

$$Q(s,a) \; + \; \alpha \, (R(s,a,s') \; + \; \gamma \, * \, max_{a'} \, Q(s', \, a') \; - \; Q(s,a))$$

State-Action	Q-value
(1, 1), East	0 + (0.01*(-0.04 + (0.95*(0)))) = -0.0004
(2, 1), North	0 + (0.01*(-0.04 + (0.95 * (0)))) = -0.0004
(3, 1), East	0 + (0.01*(-0.04 + (0.95 * (0)))) = -0.0004
(3, 2), South	0 + (0.01*(-0.04 + (0.95 * (0)))) = -0.0004
(4, 2)	0 + (0.01*(-1 + (0.95 * (0)))) = -0.01

(30 min) C. Epsilon-Greedy Action Selection	start time:



https://www.baeldung.com/cs/epsilon-greedy-q-learning

15. In previous sections, actions are given in the trial. However, in the real game, if the action at state s is always chosen by the action with the highest q value at s, what is the disadvantage of choosing this strategy to choose the action at state s?

    - If the agent always chooses the state and action with the highest q value, it will always try to maximize its rewards, causing infinity loop in trying to max out rewards and never-ending the game.

16. A good approach is epsilon-greedy as shown in the picture above, each time when the agent to choose an action, it compares a random number with a pre-fixed epsilon, explain the graph showing which action will be taken based on the random number?

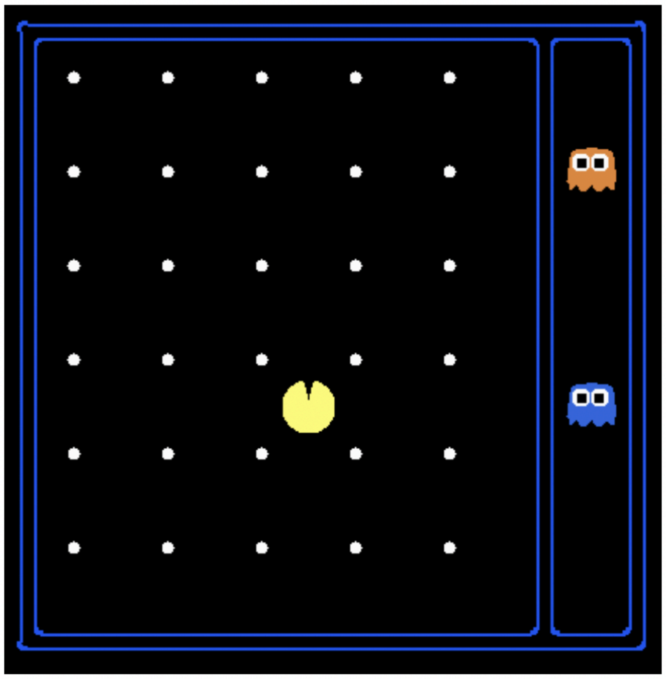
    - The pathway 1-epsilon causes choosing the highest reward with exploitation, otherwise exploration and random action is encouraged.
    - If epsilon was 1, it would explore, if epsilon was 0, it would exploit.

- Define epsilon, choose a random number, if random number is larger than epsilon, choose best action

17. A more sophisticated strategy uses epsilon decay. It gradually reduces the value of epsilon from initial epsilon to final epsilon during the iteration. Explain the benefit of using this strategy.
    - Because as the value of epsilon decreases, the trials would be encouraged to exploit more than explore - causing the agent to choose best action.

18. In which environment, q learning works best?
    - Q-learning works best when the environment is model free and the environment is very structured and defined. Eg: Frozen lake and gridworld

D. Approximate Q Learning	start time:



19. There are food, pacman, and ghosts in the environment. How do you represent the state?

- Matrix or list
- Coordinates of pacman and ghosts and if there is a food on that coordinate

20. If pacman has 4 actions (NSEW), how many pairs of state and action? In other words, if using q-learning, what is the size of the q-table?

- 120 rows, each ghost have a position so 120*3, food 2^30 * 4 actions
- 120*3 * 2^30… * 4

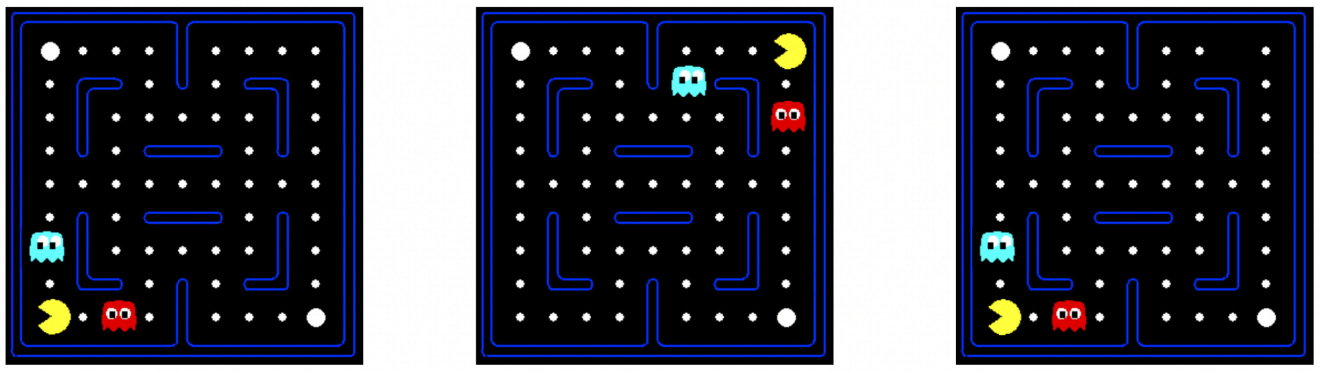
21. Based on your answer in question 20, can we still use q learning to learn a policy for the pacman in this environment?

- No

22. If your answer in question 21 is no, what should we do to make a policy for the pacman at this size?

    - Approximate Q-learning. No need to exactly solve that Q-table.

23. Given three different states below, even though they are all different in term of the state representation, what is the similarity between them?



    - Pacman in corner trapped by 2 ghosts
    - Power pellets in the same position

24. Based on your answer to question 22, can we give all three states the similar q value?

    - Yes, they should have similar q value because pacman is stuck and cornered by 2 ghosts in all 3 states

25. We call the idea in 22 and 23 as **feature** representation of the states. Basically, instead of estimate the q value from s and a directly, we extract features from the s and a, and then evaluate the features as an approximation to the actual q value.

Suppose we summarize the s and a using 3 features: f0, f1, and f2. Each feature is a number. How do we combine 3 features as one total evaluation of s and a?

    - weighted sum

26. Justify your answer in 24 with following formula:

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- Each feature will have input s and a. output will be a number. we give each feature a weight and then take the weighted sum of all of the features.

27. What does the value of w mean when it is positive, negative, or zero?

- positive is most important
- negative is least important
- zero means we don't take that feature in count (useless)

28. Please give at least three other features of s and a for the pacman game?

- number of food around us
- closeness of ghost agents
- closeness of power pellet
- number of seconds left for ghost to be trapped

29. We will use gradient descent to iteratively solve the values of weights. We skip the mathematics calculation steps and give you the update rule below. The term *difference* in the formula is same as TD error we learned before. Please write down the difference of updating between exact/classical Q learning and approximate Q learning.

$$\text{transition } = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s, a)$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha\,[\text{difference}] \qquad \textbf{Exact Q's}$$

$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s, a) \qquad \textbf{Approximate Q's}$$

- Updating exact Q learning requires just looking at the Q-value and adding the learning rate and the difference.
- Whereas, updating the approximate Q learning requires a weight and features given a state and action instead of Q-values. The weights will add importance to the features.

30. In what kind of environments, even using approximate Q learning is very challenging?

- When the state action pairs are very expansive, lots of choices or actions in the game.