notes set 1

Super Simple Q-Learning Game

from "Teaching an AI to play a simple game using Q Learning" by Soren D

The game is the player 'P' must move to catch the chese 'C' without falling into the pit 'O'.

```
# O = = P = = = = = = C = #
  0 1 2 3 4 5 6 7 8 9 10 11 12
```

- Our strategy will be to make a program that allows the user to play the game, next we will add a random play mode, and finally we will add in the q learning algorithm

- As previously discussed q learning is a reinforcement learning algorithm. It works by allowing an agent to learn to reach a goal by taking random actions, however actions that bring the agent closer to goal are rewarded.

① From an implementation point of view, Q learning works by keeping a table of all game states, and the actions available at each of these states. For each state S and action A pair, a numerical value Q representing the possible reward for taking action A at state S is kept

- The Q table is initialized with random values.
- The Q table is updated over time using an algorithm that balances exploration of the state space with use of the table to realize the agent's goals

The table is updated using the equation below

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

- $Q(s_t, a_t)$: new value
- $Q(s_t, a_t)$: old value
- α: learning rate
- r_t: instantaneous reward
- γ: discount factor
- $\max Q(s_{t+1}, a)$: reward available by taking best action from state s_t which lands agent in s_{t+1}
- $Q(s_t, a_t)$: old value

at s_t we take optimal action, we land in s_{t+1} giving agent max reward possible at that state

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left(r_t + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t)\right)$$

- this looks different from what we have seen in Mitchell's book, but similar to what is in Sutton & Barto
- to clarify, let $\alpha = 1$

$$Q(s_t, a_t) = Q(s_t, a_t) + 1 \cdot (r_t + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t))$$

$$= \cancel{Q(s_t, a_t)} + r_t + \gamma \cdot \max Q(s_{t+1}, a) - \cancel{Q(s_t, a_t)}$$

$$Q(s_t, a_t) = r_t + \gamma \cdot \max Q(s_{t+1}, a)$$

↑ that is what is seen in Mitchell

so

$$Q(s_t, a_t) = r_t + \gamma * \max\left(Q(s_{t+1}, a)\right)$$

↑

new value = immediate reward + discounted best available action

(3) The steps of the algorithm are:

- Initialize the Q table with random values / or 0

  for each s_t in S

  for each a_t available at s_t

  $Q(s_t, a_t) = \text{random / or } 0$

- While (playing) {

  chance = random(1, 100)

  if (chance < threshold) {

  select random action - a_t

  }

  else {

  select a_t such that it is max a_t for $Q(s_t, a_0 \cdots a_n)$

  }

  do action a_t, meaning

  next State = q Matrix (a_t)

  similar to adjacency matrix, here we have a table that tells us what the next state given an action a_t at the current state is.

  Update Q table

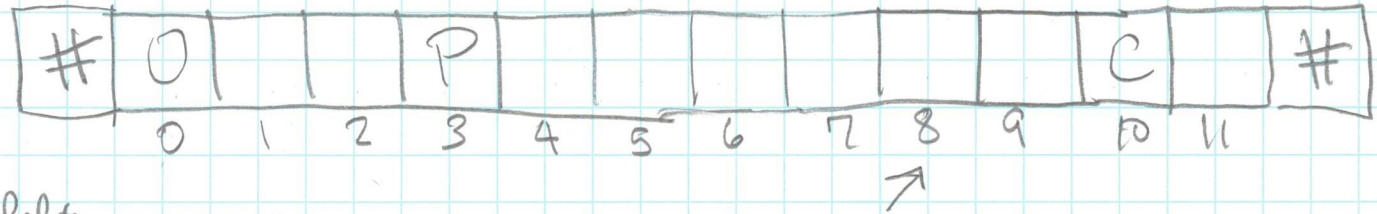
  $Q(s_t, a_t) = r_t + \max Q(\text{next State}, a_i)$

  a_i is highest reward a at next state

  } s_t = nextstate

(4)

The Q-matrix or Qadj identifies the action / actions needed to move from state to state



going left will not take P off game board

left right

here is the game board

qMatrix = [[0, 1], [0, 2], [1, 3], [2, 4], [3, 5],
[4, 6], [5, 7], [6, 8], [7, 9], [8, 10]
[9, 11], [10, 11]]

state 0

in last state, going right will not take you off the game board

* As an aside, in graph learning, the game states are not known. It is up to the exploration to find them.

- In q learning the states are known, the actions that move the agent from one state to another are known, they are recorded in Q-matrix, or Q-adj.

- The goal of q learning is to find the most efficient way to get to the goal state.

(7) Generally in Q-learning / reinforcement learning the Q table is used to train a neural network.