Review of Reinforcement Learning

- Reinforcement learning problems are similar to MDPs. You have:
  - a set of states \( s \in S \) and actions \( a \in A \)
  - one start and possibly many terminal states
  - an UNKNOWN set of transition probabilities \( T(s, a, s') \) and rewards \( R(s, a, s') \)

In MDPs the transition probabilities and the rewards are known, in reinforcement learning problems they are unknown.

- Passive Learning:
The agent does not get to choose which action to take. It is as if the policy, \( \pi \), is given. It only observes the sequence of states, actions, and rewards.

  - Model Free Learning:
The agent only learns the values for the states (or Q-states) not the transition probabilities.
    * Direct Estimation
      The agent learns values of states via:
      \[
      V^\pi(s) = \frac{\sum_i \text{total reward of episode } i}{\sum_i \# \text{ of times agent visited } s \text{ in episode } i}
      \]
      Alternatively, the agent can learn values of Q-states:
      \[
      Q(s, \pi(s)) = \frac{\sum_i \text{total reward of episode } i}{\sum_i \# \text{ of times agent took action } \pi(s) \text{ from state } s \text{ in episode } i}
      \]
      Con: have to wait for many episodes to learn anything at all
    * Running Average Approaches:
      Average old values with new values weighted by \( \alpha \), the “learning rate”.
      - Temporal Difference Learning (TD Learning)
        Learns values of states via:
        \[
        V^\pi(s) \leftarrow (1 - \alpha) [V^\pi(s)] + \alpha [R(s, \pi(s), s') + \gamma V^\pi(s')]
        \]
        Con: computing policy is impossible since \( T \) is unknown
      - Q-Learning
        Learns values of Q-states via:
        \[
        Q(s, a) \leftarrow (1 - \alpha) [Q(s, a)] + \alpha \left[ R(s, \pi(s), s') + \gamma \max_{a'} Q(s', a') \right]
        \]
        Con: no generalization happens; all states look equally different. This means the agent must do much more exploring.
      - Feature representation of Q-states
        \[
        Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \ldots + w_n f_n(s, a)
        \]
        Learn the weights via:
        \[
        w_i \leftarrow w_i + \alpha \left[ R(s, \pi(s), s') + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] f_i(s, a)
        \]
- Model Based Learning:
The agent learns the model \((T(s, a, s') \text{ and } R(s, a, s'))\) rather than the values.

  * Empirical Model Learning
    - record \(R(s, a, s')\) upon experiencing it
    \[
    T(s, a, s') = \frac{\text{# of times agent sees } (s, a, s')}{\sum_x \text{# of times agent sees } (s, a, x)}
    \]

  * Adaptive Dynamic Programming (ADP)
    - learn \(T\) and \(R\) through Empirical Model Learning
    - use policy iteration to find \(\pi^*\)
    - refine the model based on experience
    - repeat

- Action Selection
  Balance “exploration” and “exploitation”, the agent must balance using the policy it has learned with exploring the state space.

  - \(\epsilon\)-Greedy Action Selection
    With probability \(\epsilon\), take a random action.
    With probability \((1 - \epsilon)\) act optimally according to the current knowledge.
  
  - “Time Decaying” \(\epsilon\)-Greedy Action Selection
    Decay \(\epsilon\) over time, since we expect the agent to eventually have explored most of the state space.