Exam Prep Discussion 6

Reinforcement Learning!

- Typically, we navigate MDPs by using transition and reward functions to determine our optimal policy and then act on it: **offline planning**
- NOW, we don't know our transition and reward functions, so we use repeated exploration to learn those functions: **online planning**
  - Each time we explore, we get a sample: (s, a, s', r)

Reinforcement Learning!

- Typically, we navigate MDPs by using transition and reward functions to determine our optimal policy and then act on it: **offline planning**
- NOW, we don't know our transition and reward functions, so we use repeated exploration to learn those functions: **online planning**
  - Each time we explore, we get a sample: (s, a, s', r)
- **Model-based learning:** first estimate transition and reward functions, then use to determine policy

$$T(s,a,s') = \frac{\text{number of } (s,a,s')}{\text{number of } (s,a)} \qquad R(s,a,s') = r$$

- **Model-free learning:** estimate values *V* or Q-values *Q* directly, without needing to find transition or reward functions

Model-Free Learning with V (value of a state)

- **Passive RL / on-policy learning:** following a policy to state values

Model-Free Learning with V (value of a state)

- **Passive RL / on-policy learning:** following a policy to state values
- **Direct evaluation:** for each state s, learn V(s) by taking the average reward achieved by moving FROM s
  - Each state's value is calculated separately
  - Transition probabilities are ignored

Model-Free Learning with V (value of a state)

- **Passive RL / on-policy learning:** following a policy to state values
- **Direct evaluation:** for each state s, learn V(s) by taking the average reward achieved by moving FROM s
  - Each state's value is calculated separately
  - Transition probabilities are ignored
- **Temporal difference (TD) learning:** update each state's estimated $V^{\pi}(s)$ with each new sample, using exponential moving averages

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha[R(s,\pi(s),s') + \gamma V^{\pi}(s')]$$

  - α is **learning rate**
    - Bigger α: higher weight on *sample* V(s)
    - Smaller α: higher weight on *current* V(s)

Model-Free Learning with Q (value of a (state, action) pair)

- **Active RL / off-policy learning:** learning a policy

Model-Free Learning with Q (value of a (state, action) pair)

- **Active RL / off-policy learning:** learning a policy
- **Q-learning:** also uses exponential moving averages, but with Q-values

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

  - Will converge to the optimal policy even if there are suboptimal actions

Model-Free Learning with Q (value of a (state, action) pair)

- **Active RL / off-policy learning:** learning a policy
- **Q-learning:** also uses exponential moving averages, but with Q-values

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[R(s,a,s') + \gamma \max_{a'} Q(s',a')\right]$$

  - Will converge to the optimal policy even if there are suboptimal actions
- **Approximate Q-learning:** uses a feature vector to represent state-actions and a weight vector to represent reward

$$Q(s,a) = w^{\mathrm{T}} f(s,a) = \langle w, f \rangle = \sum_{i=1}^{n} w_i f(s,a)_i$$

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

$$w \leftarrow w + \alpha \cdot \text{diff} \cdot f(s,a)$$

Exploration vs. Exploitation

- **ε-greedy:** choose randomly among ALL actions w.p. ε; choose best action w.p. (1 - ε)
  - Bigger ε: more exploration
  - Lower ε: more exploitation

Exploration vs. Exploitation

- **ε-greedy:** choose randomly among ALL actions w.p. ε; choose best action w.p. (1 - ε)
  - Bigger ε: more exploration
  - Lower ε: more exploitation
- **Exploration function:** maximize over "custom-defined" function instead of Q-values while Q-learning

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} \mathbf{f(s', a')}]$$

$$f(s,a) = Q(s,a) + \frac{k}{N(s,a)}$$