Learning Objectives - Reinforcement Learning

At the end of the class you should be able to:

- Explain the relationship between decision-theoretic planning (MDPs) and reinforcement learning
- Implement basic state-based reinforcement learning algorithms: Q-learning

Reinforcement Learning

What should an agent do given:

- Prior knowledge possible states of the world
  possible actions
- Observations current state of world
  immediate reward / punishment
- Goal act to maximize accumulated (discounted) reward
- Like decision-theoretic planning, except model of dynamics and model of reward not given.

Reinforcement Learning Examples

- Game - reward winning, punish losing
- Dog - reward obedience, punish destructive behavior
- Robot - reward task completion, punish dangerous behavior

Experiences

- Assume there is a sequence of experiences:

$$state, action, reward, state, action, reward, \ldots.$$

- The sequence of experiences up to the time the agent has to choose its action is its history
- The agent has to choose its action as a function of its history.
- At any time it must decide whether to do.
  - explore to gain more knowledge
  - exploit knowledge it has already discovered

Why is reinforcement learning hard?

- What actions are responsible for a reward may have occurred a long time before the reward was received.
  - The dog is expected to determine that eating the shoe at the start of the day is what was resposible for it being scolded at the end of the day.
- The long-term effect of an action depend on what the agent will do in the future.
  - It might be okay for a robot to create a mess as long as it cleans up after itself.
- The explore-exploit dilemma: at each time should the agent be greedy or inquisitive?

Reinforcement learning: main approaches

- search through a space of policies (controllers)
- learn a model consisting of state transition function $P(s'|a, s)$ and reward function $R(s, a)$; solve this an an MDP.
- learn $Q^*(s, a)$, use this to guide action.

Recall: Asynchronous VI for MDPs, storing $Q[s, a]$

(If we knew the model:)

Initialize $Q[S, A]$ arbitrarily
Repeat forever:

- Select state s, action a
- $Q[s, a] := R(s, a) + \gamma \sum_{s'} P(s'|s, a) \left(\max_{a'} Q[s', a'] \right)$

Asynchronous VI for Deterministic RL

initialize $Q[S, A]$ arbitrarily
observe current state s
repeat forever:

select and carry out an action a
observe reward r and state s'
$Q[s, a] := r + \gamma \max_{a'} Q[s', a']$
What do we know now?
$s := s'$

Computing Averages: Temporal Differences

- Suppose we have a sequence of values:

  $$v_1, v_2, v_3, \ldots$$

  and want a running estimate of the average of the first k values:

  $$A_k = \frac{v_1 + \cdots + v_k}{k}$$

Temporal Differences (cont)

- Suppose we know A_{k-1} and a new value v_k arrives:

$$
\begin{aligned}
A_k &= \frac{v_1 + \cdots + v_{k-1} + v_k}{k} \\
&= \frac{(k-1)A_{k-1} + v_k}{k}
\end{aligned}
$$

  Let $\alpha_k = \frac{1}{k}$, then

$$
\begin{aligned}
A_k &= (1-\alpha_k)A_{k-1} + \alpha_k v_k \\
&= A_{k-1} + \alpha_k(v_k - A_{k-1})
\end{aligned}
$$

  "TD formula"

- Often we use this update with α fixed.
- We can guarantee convergence to average if $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$.
- E.g., $\alpha_k = 10/(9+k)$ treats more recent experiences more, but converges to average.

Q-learning

- Idea: store $Q[State, Action]$; update this as in asynchronous value iteration, but using experience (empirical probabilities and rewards).
- Suppose the agent has an experience $\langle s, a, r, s' \rangle$
- This provides one piece of data to update $Q[s, a]$.
- An experience $\langle s, a, r, s' \rangle$ provides a new estimate for the value of $Q^*(s, a)$:

$$r + \gamma \max_{a'} Q[s', a']$$

which can be used in the TD formula giving:

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

Q-learning

initialize $Q[S, A]$ arbitrarily
observe current state s
repeat forever:
    select and carry out an action a
    observe reward r and state s'
    $Q[s, a] := Q[s, a] + \alpha \left(r + \gamma \max_{a'} Q[s', a'] - Q[s, a]\right)$
    $s := s'$

Properties of Q-learning

- Q-learning converges to an optimal policy, no matter what the agent does, as long as it tries each action in each state enough.
- But what should the agent do?
  - exploit: when in state s, select an action that maximizes $Q[s, a]$
  - explore: select another action

Problems with Q-learning

- It does one backup between each experience.
  - Is this appropriate for a robot interacting with the real world?
  - An agent can make better use of the data by
    — remember previous experiences and use these to update model (action replay)
    — building a model, and using MDP methods to determine optimal policy.
    — doing multi-step backups
- It learns separately for each state.