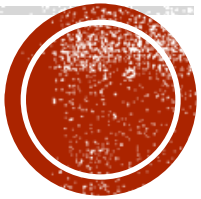
Other Settings and Variants of Reinforcement Learning



CS229

Tengyu Ma

RL Settings Covered in This Class

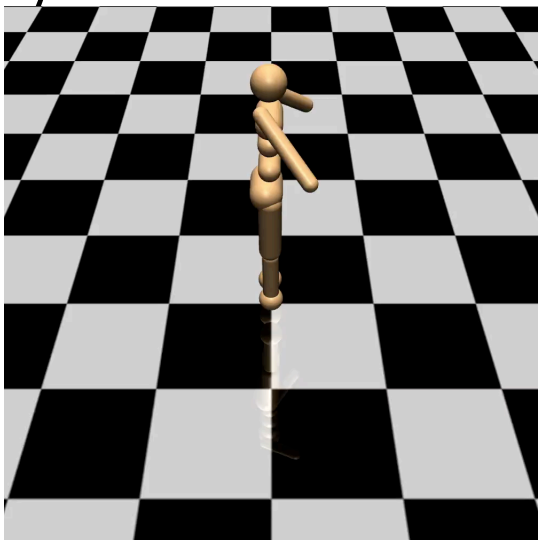
Learn from trials and errors

- Start "from scratch" (no training data)
- Interact with the environments
  - Go, chess, and Atari: trials in computers because dynamics/transition probabilities is known
  - Robotics, autonomous cars: trials in real-world
    - samples are more precious than runtime
    - sample efficiency: model-based vs model-free RL
- Observe the reward
- Observe the states
- Discrete (tabular) or continuous state space

Where do the rewards come from?

Rewards are defined by us (as algorithm designers)

- "Sparse" reward:
  - example: $R(s) = 1$ if it reaches a position with x coordinate > 10; and $R(s) = 0$, otherwise.
  - issue: if we always obtain 0 (or constant) reward, then no information is gained
- "Dense" reward:
  - e.g., $R(s) =$ x velocity



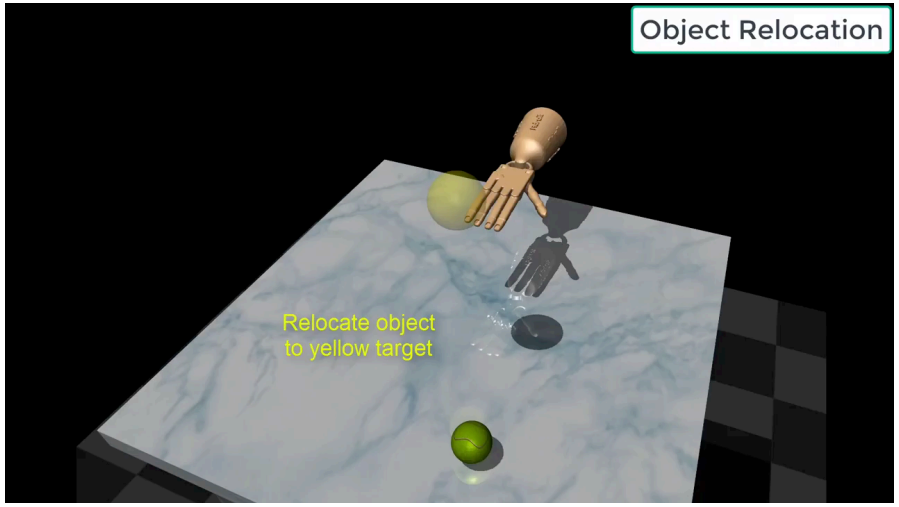
Rewards Shaping (Designing Dense Rewards)

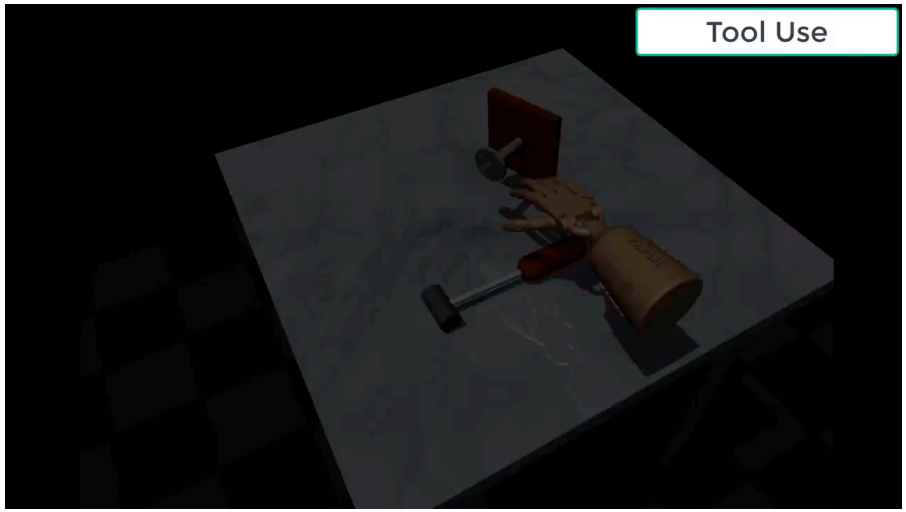
Reward = distance between current rotation angle and the desired angle

Note that the rotation angle is not part of the states, and need to be predicted by a neural net

Video credit: openAI blog
https://openai.com/blog/learning-dexterity/
Learning Dexterous In-Hand Manipulation

Reward Shaping may be Difficult for Complex Tasks





Reward = ?

- negative distance between hand to the goal location ✗
- negative distance between the ball and the goal location
  - no informative reward before the hand accidentally touches the ball
- A complicated reward may work

- dist(hand, ball) if the hand doesn't touch the ball; - dist(ball, goal) o.w.

Video credit: Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations, Rajeswaran et al.'17. https://www.youtube.com/watch?time_continue=53&v=jJtBll8l_OM

RL with Sparse Rewards

Generally very hard without additional structure

- Consider a bit-flipping environment:
  - $\mathcal{S} = \{0,1\}^d$ with very large d
  - $\mathcal{A} = \{1, \dots, d\}$
  - Taking action i at state s will flip s_i
  - $g \in \mathcal{S}$ is an unknown goal state
  - observed reward $R(s) = 1(s = g)$
- We got zero information (reward = 0) unless we are lucky to hit g

Hindsight Experience Replay [Andrychowicz et al.'17]

Learning with Sparse Rewards

What if we also know the goal g?

- Reward shaping: $R(s) = dist(s, g)$
  - (but this used the prior knowledge of the problem)
- Another idea: hallucinating goals
  - Solve a more general question: for any pair of (s_0, h), how to reach the state h from state s_0?
  - Goal-conditioned value function/Q functions $V(s, h)$, $Q(s, a, h)$
    - $V(s, h)$ the value of the state s if the goal is h
  - Design/hallucinate many other goal states $h's$, for which we can get positive rewards, so that we can train V and Q
    - The value or Q will be correct on g as well, because of the extrapolation of neural networks

Hindsight Experience Replay [Andrychowicz et al.'17]

RL Settings Covered in This Class

Learn from trials and errors

- Start "from scratch" (no training data)
- Interact with the environments
- Observe the reward
- Observe the states
- Discrete (tabular) or continuous state space

Partially Observable Markov Decision Process (POMDP)

- The state s is not fully observed
  - observe o_t which is a random variable depending on s_t
- Example 1: contract force in robotics may be hard to measure
- Example 2: opponent's information when playing action real-time strategy games (such as starcraft, dota, and league of legends)
- A classic but hard-to-scale method: maintain a "belief state" (posterior of states given all previous observations), and design stochastic policy that takes the belief state as input

Very related setting: observations are from the camera (in pixel space)

- Idea: use unsupervised learning to infer the latent states

RL Settings Covered in This Class

Learn from trials and errors

- Start "from scratch" (no training data)
- Interact with the environments
- Observe the reward
- Observe the states
- Discrete (tabular) or continuous state space

Imitation Learning (Learning from Demonstrations)

Additional access to a set of demonstrations --- successful trajectories by an expert policy

- Suppose an expert policy π_E generates n i.i.d trajectories $\{\tau^{(i)}\}_{i=1}^{n}$
  - The expert is assumed to have near-optimal rewards
- Goal: learn a policy π from $\{\tau^{(i)}\}_{i=1}^{n}$ that imitates π_E

Behavioral cloning (BC): treat it as supervised learning problem

- Limitation: cascading errors (see the board)

RL Settings Covered in This Class

Learn from trials and errors

- Start "from scratch" (no training data)
- Interact with the environments
- Observe the reward
- Observe the states
- Discrete (tabular) or continuous state space

Imitation Learning with Environment Interactions

- Paradigm: learn some policy π_0 from $\{\tau^{(i)}\}_{i=1}^{n}$ first, and then use an RL algorithm that interacts with the environment to refine the policy π_0
- Simplest approach: initialize the RL algorithm with π_0
- Better approach: in the RL algorithm, constrain the policy to be somewhat close to π_0
- An active research area

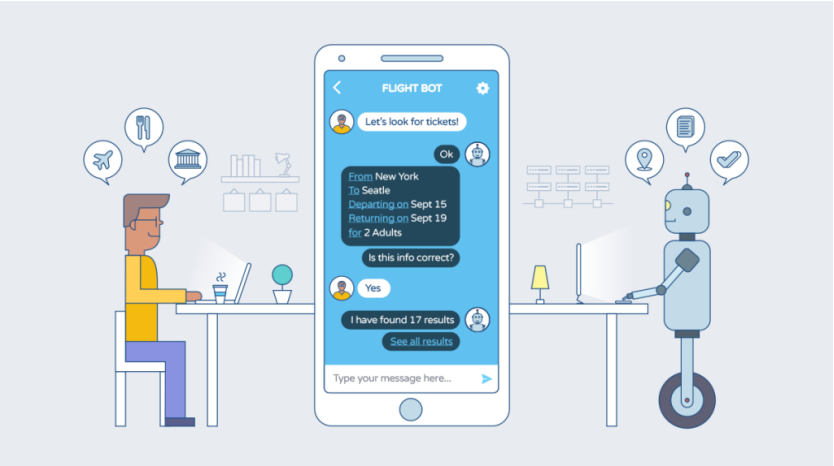
Overcoming Exploration in Reinforcement Learning with Demonstrations [Nair et al.'18]
Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards [Vecerik et al.'18]
Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. [Rajeswaran et al'18]
Deep q-learning from demonstrations. [Hester et al.' 18]

RL Settings Covered in This Class

Learn from trials and errors

- Start "from scratch" (no training data)
- Interact with the environments
- Observe the reward
- Observe the states
- Discrete (tabular) or continuous state space

Applications of RL in NLP



- Training/controlling a chat bot
- Actions: sentences that the bot say
- States: all past messages
- Goal: the bot completes some tasks, e.g., books a ticket for the human
- Challenges
  - action space and state space are huge discrete space (all possible sentences)
  - large-scale interactions with humans are hard to obtain or simulate

RL Settings Covered in This Class

Learn from trials and errors

- Start "from scratch" (no training data)
- Interact with the environments
- Observe the reward
- Observe the states
- Discrete (tabular) or continuous state space

Bottom line: choose the most suitable setting and algorithms for you applications