ESE-680-005 Reinforcement Learning

Rebecca Li

Fall 2019

Organization

- Instructors: Miguel and Santiago
- TAs: Clark, Kate, Arbaaz (Monday 5-7 GRASP conf room, Wednesday 9-11 452C walnut 3401, Arbaaz on demand)
- Homeworks: 50%, groups of 2 students
- Midterm: Oct 17th (MDP, policy gradient)
- Take-home final: Dec 5th: Theoretical, implementation
- Textbook: "Reinforcement Learning: an introduction" by Sutton and Barto
- Textbook: "Algorithms for Reinforcement Learning" Csaba Szepesvari, https://sites.ualberta.ca/ szepesva/RLBook.html

---

1 Lecture 1: Overview

1.1 What is Reinforcement Learning (RL)

A **Model-free** framework to formalize **sequential** decision making. We have an agent at time t that interacts in the environment:

- Actions $A_t \in \mathcal{A}(s)$ (possibly state dependent)
- States $S_t \in \mathcal{S}$
- Reward R_t

Our goal is to learn the best policy $\pi^*(s)$ to find the best action in the world. Some particulars:

- No supervision, only reward signal
- feedback is delayed (not instantenous)
- time matters, sequential, data not i.i.d.
- Agent picks new data

The future of RL:

- RL and robotics
- Autonomous driving
- Safe learning

1.2 Why use RL?

In the case of the cart-pole, we have a fairly simply problem which we already know how to control. This is nice because we can benchmark against the designed controller. However, in practice we should do something else, like a PID controller because it will be faster to use. Despite this, it makes a good test problem.

Another problem that RL could be good at is a problem with delayed/sparse reward, such as making it to the top of a mountain.

1.3 Markov Decision Process

Definition 1.1. *Markov Decision Process*
A **memory-less process** that given a state s_t and action a_t, will transition to new s_{t+1}. It also receives a reward r_t after taking the action.

Our goal is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes reward:

$$\pi^*(s) = \arg\min_a \mathbb{E}[r(s, a)]$$

1.4 Successes in RL

- "Playing Atari with Deep Reinforcement Learning" V.Mnih et.al. (2013)
- Go success. Part of the problem with Stockfish or Elmo have hardcoded features, while AlphaGo Zero has no such features. This is great for sparse reward problems. AlphaZero learns value of the states, and then selects action with the best option.
- AlphaStar. Some problems are imperfect information, need for long term planning, large action space, and still requires 200 years of gameplay.
- OpenAI Five: multiagent problems. 45000 years of self-play over 10 months.
- Aggressive helicopters
- Grasping tasks, arm farms.