Basics of reinforcement learning

Yixin Lin

Duke University

yixin.lin@duke.edu

February 15, 2017

Overview

Introduction

- Picking the right problem is *hard*
- Capturing the right abstraction level: all the relevant details, none of the irrelevant ones
- Three (extremely general!) types of tasks in machine learning
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

Introduction

- Reinforcement learning is the problem of *sequential decision making in dynamic environment*
- Goal: capture the most important aspects of an agent making decisions
  - Input (sensing the state of the environment)
  - Action (choosing to affect on the environment)
  - Goal (prefers some states of the environment over others)
- This is *incredibly* general
- Examples
  - Robots (and their components)
  - Games
  - Better A/B testing
  - Most importantly, **you**!

Difficulties

- Curse of dimensionality: too many choices!
- Stochasticity, and more generally, uncertainty: what do I do?
- Reward choice: what do I care about?
- *Credit assignment*: what did I do wrong?

Elements of reinforcement learning

- Policy: how I act in a given situation
- Reward signal: pain and pleasure at the moment
- Value function: total amount of happiness from now until I die

The Markov Decision Process (MDP)

- S: set of possible **states** of the environment
  - $p(s_0), s_0 \in S$: a distribution over initial state
  - Markov property: we assume that the current state summarizes everything we need to remember
- A: set of possible **actions**
  - $P_{sa}(\cdot)$: state transitions, for each state s and action a
- $R : S \to \mathbb{R}$: **reward**
  - $\gamma \in [0, 1]$: discount factor

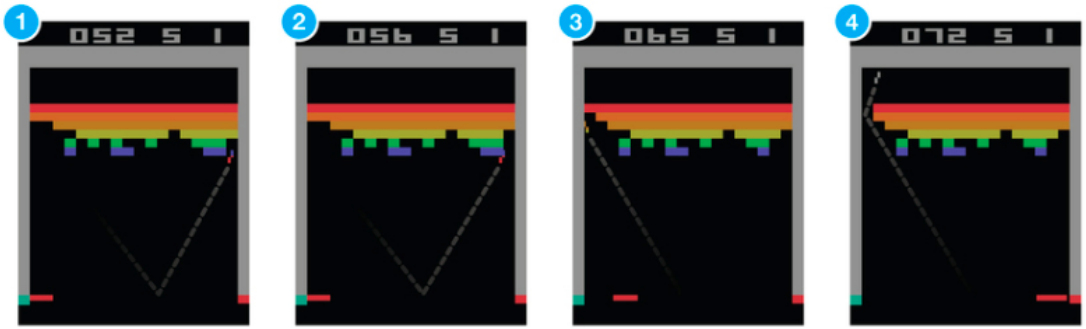
The Markov Decision Process (MDP)

- π: a policy (what action to do, given a state)
- $V^{\pi}(s) = E_{\pi}[\sum_{i=0}^{\infty} \gamma^i R(s_i)]$
  - How good is a state, given a policy?
- $Q^{\pi}(s, a)$
  - How good is an action at a state, given a policy?
- More realistic: Partially Observed MDPs (POMDPs), where state is not directly observed but affects observations

Techniques for solving RL problems

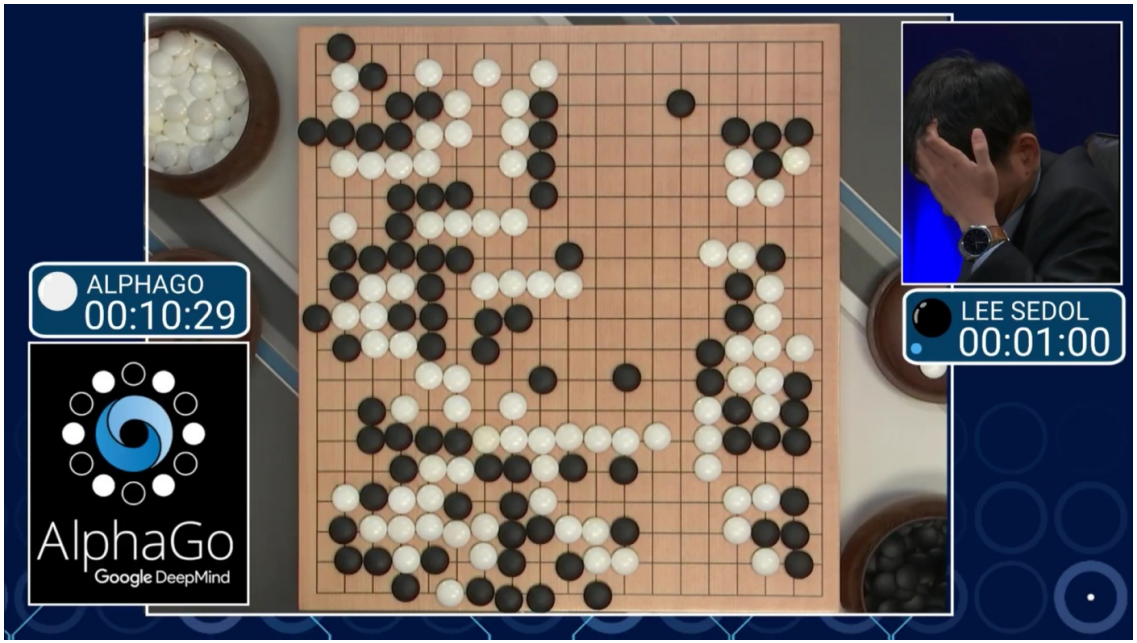
- Dynamic programming
- Temporal difference (TD) learning
- Q-learning: find the Q function (how good is the action?)
- Policy gradients: optimize the policy directly
- Deep RL: use deep neural networks as function approximators
  - Deep Q-learning
  - Policy gradients: learn a policy directly

Recent successes: Atari



- "Human-level control through deep reinforcement learning", Mnih et al.

Recent successes: AlphaGo



- "Mastering the game of Go with deep neural networks and tree search", Silver et al.

Failure modes

- Misspecifying reward functions (OpenAI blog)
- Concrete AI safety problems
  - Safe exploration (don't destroy the environment or itself)
  - Changing environments and data (fail gracefully when data changes)
  - Avoid negative side-effects (can't explicitly describe every bad behavior)
  - Avoid wireheading (reward signals are faulty)
  - Scalable oversight (when feedback and oversight is expensive)

AI safety problems

- Recent papers
  - "Concrete problems in AI safety", Amodei et al.
  - "Safely interruptible agents", Orseau and Armstrong
- Organizations
  - Machine intelligence research institute (MIRI)
  - Future of humanity institute (FHI)
  - Partnership on AI
    - AAAI, ACLU, Amazon, Google (DeepMind), Facebook, IBM, Microsoft, Apple, OpenAI)

Thanks!

Resources and references: yixinlin.net/intro-rl