

Reinforcement Learning

Jordan Boyd-Graber

University of Maryland

Introduction / Foundations



Steve Gorton and Tim Ridley, Alexander Hafemann/Getty Images



From iScoop

Control Learning

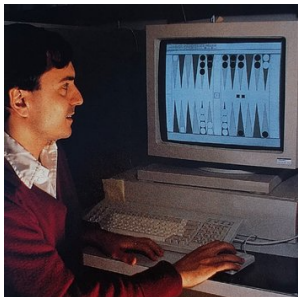
Consider learning to choose actions, e.g.,

- Roomba learning to dock on battery charger
- Learning to choose actions to optimize factory output
- Learning to play Backgammon

Problem characteristics:

- Delayed reward
- Opportunity for active exploration
- Possibility that state only partially observable
- Possible need to learn multiple tasks with same sensors/actuators

Early Example: TD-Gammon



Learn to play Backgammon
[Tesauro, 1995]

- +100 if win
- -100 if lose
- 0 for all other states

Trained by playing 1.5 million games against itself
Approximately equal to best human player

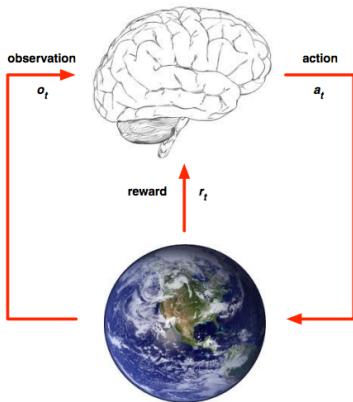
Where RL is Now

- Language Model Alignment
- Machine Translation
- Question answering
- Starcraft
- Go
- Atari
- Robotics

The Problem of Delayed Reward

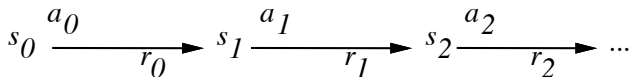
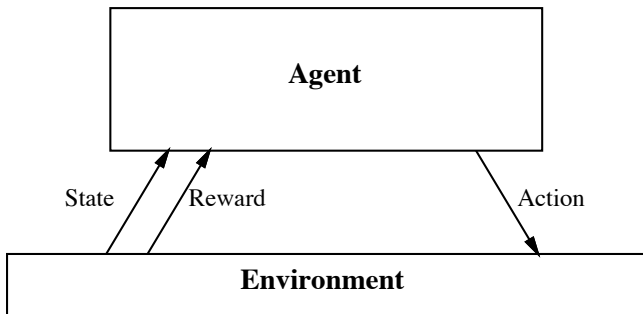
- Mistakes now could have a big cost in the future
- You need to set up an opportunity now for a payoff in the future
- Hard to know which is which

Reinforcement Learning Problem



- At each step t the agent:
 - ▶ Executes action a_t
 - ▶ Receives observation o_t
 - ▶ Receives scalar reward r_t
- The environment:
 - ▶ Receives action a_t
 - ▶ Emits observation o_{t+1}
 - ▶ Emits scalar reward r_{t+1}

Reinforcement Learning Problem



Goal: Learn to choose actions that maximize

$$r + \gamma r + \gamma^2 r + \dots^9, \text{ where } 0 \leq \gamma < 1$$

Markov Decision Processes

Assume

- finite set of states S
- set of actions A
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- then receives immediate reward r_t
- and state changes to s_{t+1}
- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
 - ▶ i.e., r_t and s_{t+1} depend only on current state and action
 - ▶ functions δ and r may be nondeterministic
 - ▶ functions δ and r not necessarily known to agent

State

- Experience is a sequence of observations, actions, rewards

$$o_1, r_1, a_1, \dots, a_{t-1}, o_t, r_t \quad (1)$$

- The state is a summary of experience

$$s_t = f(o_1, r_1, a_1, \dots, a_{t-1}, o_t, r_t) \quad (2)$$

- In a fully observed environment

$$s_t = f(o_t) \quad (3)$$

Agent's Learning Task

Execute actions in environment, observe results, and

- learn action policy $\pi : S \rightarrow A$ that maximizes

$$\mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots]$$

from any starting state in S

- here $0 \leq \gamma < 1$ is the discount factor for future rewards

Note something new:

- Target function is $\pi : S \rightarrow A$
- but we have no training examples of form $\langle s, a \rangle$
- training examples are of form $\langle \langle s, a \rangle, r \rangle$

What makes an RL agent?

- Policy: agent's behaviour function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment

Policy

- A policy is the agent's behavior
 - ▶ It is a map from state to action:
 - ▶ Deterministic policy: $a = \pi(s)$
 - ▶ Stochastic policy: $\pi(a | s) = p(a | s)$

Value Function

To begin, consider deterministic worlds . . .

For each possible policy π the agent might adopt, we can define an evaluation function over states

$$\begin{aligned} V^\pi(s) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \\ &\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

where r_t, r_{t+1}, \dots are from following policy π starting at state s

Example: Imitation Learning

- Take examples of experts $\{(s_1, a_1) \dots\}$
- Learn a classifier mapping $s \rightarrow a$
- Create loss as the negative reward

Example: Imitation Learning

- Take examples of experts $\{(s_1, a_1) \dots\}$
- Learn a classifier mapping $s \rightarrow a$
- Create loss as the negative reward
- What if we diverge?

Likelihood Ratio Policy Gradient

Let τ be state-action $s_0, u_0, \dots, s_H, u_H$. Utility of policy π parametrized by θ is

$$U(\theta) = \mathbb{E}_{\pi_{\theta}, U} \left[\sum_t^H R(s_t, u_t); \pi_{\theta} \right] = \sum_{\tau} P(\tau; \theta) R(\tau). \quad (4)$$

Our goal is to find θ :

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} p(\tau; \theta) R(\tau) \quad (5)$$

Approaches to RL

Value-based RL

- Estimate the optimal value function $Q^*(s, a)$
- This is the maximum value achievable under any policy

Policy-based RL

- Search directly for the optimal policy π^*
- This is the policy achieving maximum future reward

Model-based RL

- Build a model of the environment
- Plan (e.g. by lookahead) using model

