Reinforcement Learning: A Practical Introduction

Joe Booth
Orions Systems, Inc / Microsoft
September 2020
Distributed, Hierarchical, AI and human compute, video analytics platform for enterprise and intelligence agencies.

• v1: 100% human compute to annotate sports in near real time
  • Distributed = many humans in parallel
  • Hierarchical = tag start/end of play -> label play, touches -> intent (forced/unforced error)
  • 100k games per year

• v2 added AI and human compute
  • Automated training data, real time QA
  • Supports scenarios beyond today’s AI capabilities

Acquired by Microsoft, July 2020
  • Team, tech joined Dynamics Connected Store group
Introduction to Reinforcement Learning (RL)
RL vs. Supervised, Un-/semi-supervised Learning

1. RL = an agent **actively searching** a dataset
   - The agent’s action at timestep $t$, impacts observations at future timesteps
   - ... In chess, each action changes the environment
   - ... In multi-armed bandit, each action changes the information we have about the environment

2. RL is for searching truly massive search spaces
   
   Go has more valid states than atoms in the universe.

   **RL is a tool for exploring massive search spaces**
Examples of Real-world Deployments

Self driving (Wayve, etc.)

Industrial Control (Microsoft, Google, etc)

Robotics (Covariant, etc.)

+ Financial Tech
+ Search
+ Route Optimization
Technical Overview of Reinforcement Learning
Technical Overview of RL

- Deep Learning
- Approximation Methods
- Tabular Methods
Technical Overview of RL

Deep Learning

Approximation Methods

Tabular Methods

• Multi-arm bandits
• Dynamic Programming
• Markov Decision Process
• TD-Learning, Bootstrapping
• Planning
Multi-armed Bandits

Don’t know payout ratio.

Each action costs $1

Exploration = search (choose a random arm) vs
Exploitation = choose best arm
Multi-armed Bandits

Don’t know payout ratio.
Each action costs $1
Exploration = search (choose a random arm) vs
Exploitation = choose best arm

Epsilon Greedy
Example: $e=0.1$
If random value > $e$, Exploit else Explore

Note: Used in DQN and many RL algos
## Multi-armed Bandits

<table>
<thead>
<tr>
<th>Arm</th>
<th>Reward</th>
<th>Average Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Don’t know payout ratio.

Each action costs $1

Exploration = search (choose a random arm) vs Exploitation = choose best arm

### Upper Confidence Bounds

Choose arm with highest potential return (based on confidence)

or Random If 2+ arms have same score

Note: Used in AlphaZero / MuZero
Finite Markov Decision Processes

Figure 3.1: The agent–environment interaction in a Markov decision process.
Finite Markov Decision Processes – Blackjack

Figure 3.1: The agent-environment interaction in a Markov decision process.
Finite Markov Decision Processes – Blackjack

200 States:
- Player Hand 12 to 21
- Dealer Showing A to 10
- Player Ace / No Ace
  - \(10 \times 10 \times 2\)

2 Actions:
- Hit or Stick

Figure 3.1: The agent-environment interaction in a Markov decision process.
Finite Markov Decision Processes – Blackjack

200 States:
- Player Hand 12 to 21
- Dealer Showing A to 10
- Player Ace / No Ace = 10x10x2

2 Actions:
- Hit or Stick

Figure 5.1: Approximate state-value functions for the blackjack policy that sticks only on 20 or 21, computed by Monte Carlo policy evaluation.

Figure 3.1: The agent-environment interaction in a Markov decision process.
Technical Overview - Tabular Solution Methods

Rich body of academic work spanning ~40 years

- Monte Carlo Methods
- TD Learning
- n-step Bootstrapping
- Planning and Learning
  - Dyna:
  - Monte Carlo Tree Search

Dyna – learns in real time
Technical Overview of RL

Deep Learning
- DQN / Atari – Deep Learning
- AlphaZero – self play
- AlphaStar(craft)/ OpenAI Five - scale

Approximation Methods
- Value Function Methods
- On-Policy vs Off-Policy
- Policy Gradient & Actor-Critic Methods
- Nonlinear Function Approximation

Tabular Methods
- Multi-arm bandits
- Dynamic Programming
- Markov Decision Process
- TD-Learning, Bootstrapping
- Planning
How to deal with large state spaces:

Why Tabular to Function Approximation?

#1 Intractability / memory

#2 Samples needed to fill large tables
Technical Overview - Approximation Methods

Rich body of academic work spanning ~40 years

• Value Function Methods
• Policy Gradient Methods
• Actor-Critic Methods
• Nonlinear Function Approximation (Deep RL)
Technical Overview - Approximation Methods

Nonlinear Function Approximation – DQN Atari

Network:
- 32 filters of 8 x 8, stride 4
- 64 filters of 4 x 4, stride 2
- 64 filters of 3 x 3, stride 1
- 512 fc
- 4 actions

Experience Replay:
- State, action, reward.
- Size = 1,000,000

States:
- $84 \times 84 \times 4 = 28,224$

4 Actions:
- NoOp, Fire, Left, Right

Figure 3.1: The agent-environment interaction in a Markov decision process.
Technical Overview - Approximation Methods

**Model Free**
- OpenAI Five
- AlphaStar

**Model Based**
- AlphaZero
Technical Overview of RL

- **Deep Learning**
  - MuZero – planning without access to dynamics
  - SuperDyna – approximation + partial observability + temporal abstraction + non-stationary
  - DQN / Atari – Deep Learning
  - AlphaZero – self play
  - AlphaStar(craft)/ OpenAI Five - scale

- **Approximation Methods**
  - Value Function Methods
  - On-Policy vs Off-Policy
  - Policy Gradient & Actor-Critic Methods
  - Nonlinear Function Approximation

- **Tabular Methods**
  - Multi-arm bandits
  - Dynamic Programming
  - Markov Decision Process
  - TD-Learning, Bootstrapping
  - Planning
Problem with Model Based

• We need the model – AlphaZero has the simulation of chess, Go, etc

• How could we do this for Atari?

Answer: MuZero, (Schrittwieser, et. al. 2019)
Problem:

4k ROM vs A Google Data Center

Proposed solution: SuperDyn (working title) Rich Sutton 2019

• Learns subproblems, learns solutions (policies), learns state features, learns models of the world

Reality:

World is much bigger than Agent
How to Structure Problems to Use Reinforcement Learning Effectively
Example RL Problem

Camera on Edge device:

$\$$ service
(say person detection)

Goal: minimize cost

don’t request same person twice
don’t miss person

Figure 3.1: The agent-environment interaction in a Markov decision process.
Example RL Problem

Camera on Edge device:
- $$$ service
  (say person detection)

Goal: minimize cost
- don’t request same person twice
- don’t miss person

States: 7,056 x n
84 x 84 x n
n=number of history frames

2 Actions: Skip, Send

Reward:
- +1 new person,
- -0.1 same person, no one
- -1 missed person

Figure 3.1: The agent-environment interaction in a Markov decision process.
Example RL Problem

Camera on Edge device:

```plaintext
$$\text{service}$$
(say person detection)
```

Goal: minimize cost
don’t request same person twice
don’t miss person

---

States: $2 \times n$
% change $\times n$
$n=$number of history frames
+ previous actions

2 Actions: **Skip**, **Send**

Reward:
+1 new person,
-0.1 same person, no one
-1 missed person

---

![Diagram](image.png)

Figure 3.1: The agent-environment interaction in a Markov decision process.
Conclusions

Reinforcement Learning:

• A tool for exploring massive search spaces
• In use today
• Richly researched domain with continued innovation
• May be easier to use than you think
Courses

Coursera: Reinforcement Learning Specialization (U. Alberta)
coursera.org/specializations/reinforcement-learning

Udacity: Deep Reinforcement Learning
udacity.com/course/deep-reinforcement-learning-nanodegree--nd893

OpenAI: Spinning Up (free)
spinningup.openai.com/en/latest/

Books

Reinforcement Learning (Sutton & Barto) (free)
incompleteideas.net/book/the-book-2nd.html

Contact Joe

Joe.Booth@Microsoft.com
@iAmVidyaGamer
github.com/Unity-Technologies/marathon-envs