Reinforcement Learning for the real world

Harshad Khadilkar
Tata Consultancy Services Ltd.
Motivation: Why RL?
This is about letting an ecosystem of machines teach itself superhuman capabilities.

Why?

“Because it’s there”

- George Mallory (1923), when asked why he wanted to climb Mt. Everest
2 Motivation
RL in the optimization space

<table>
<thead>
<tr>
<th>Easy</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>How many eggs for breakfast</strong></td>
<td><strong>How much down-payment on car loan</strong></td>
</tr>
<tr>
<td><strong>Which lane to choose at the toll booth</strong></td>
<td><strong>Packing irregular boxes arriving on a conveyor belt</strong></td>
</tr>
</tbody>
</table>

- Slow
- Fast

- Difficulty increases
- Pressure increases
Motivation
RL in the optimization space

- Linear programming and its variants
- Rule-based planning
- Supervised deep learning
- Meta-heuristics
- Reinforcement learning

Thinking, Fast and Slow
Daniel Kahneman
Winner of the Nobel Prize

Pressure increases
Difficulty increases
Motivation

When to use RL

Necessary conditions: Answer YES to all of the following

Use for tasks that humans find hard to do (or to do well) → No ideal reference

When time is short → Can’t search or solve in real-time

When the system is hard to define, or complex → No analytical relationships

“The most important training in Unseen University [for wizards] wasn’t how to do magic, but to know when not to use it” - Terry Pratchett
This is about letting an ecosystem of machines teach itself superhuman capabilities

Why?

“Because it’s there”
- George Mallory (1923), when asked why he wanted to climb Mt. Everest

How?

Let the algorithm explore the environment on its own, while learning from experience

Reinforcement learning
Briefly: What is it?
Learning to maximise long-term reward through interaction with the environment

Action at time $t$

<table>
<thead>
<tr>
<th>RL Agent</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reward at time $t - 1$

State at time $t$
How RL works
Context: Existing work

Is this a new idea? Not at all.

Aerospace: Adaptive control
Ops Research: Dynamic programming
Computer science: Reinforcement learning
How RL works

Anatomy of an RL problem

Strictly speaking, must be a Markov Decision Process defined by

(States, Actions, Rewards, Transitions, Discount factor)
How RL works

Anatomy of an RL problem

State at $t$

Action at $t$

Reward at $t$

State at $t+1$

Action at $t+1$

Reward at $t+1$

State at $t+2$

Action at $t+2$

Reward at $t+2$

I want to maximise long-term return from $t$ to infinity

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$$
How RL works

Anatomy of an RL problem

State at $t$

Action at $t$

Reward at $t$

Can be any function from $(state, action) \rightarrow scalar$

Use neural network, with RHS of equation providing labels

Value-based Deep RL

This is unknown at $t$, but is ‘known’ at $t+1$

All value-based approaches

\[
Q(s_t, a_t) = R_t + \gamma Q(s_{t+1}, a_{t+1})
\]
How RL works

Anatomy of an RL problem

- **State at** $t$
- **State at** $t+1$
- **State at** $t+2$
- **Action at** $t$
- **Action at** $t+1$
- **Action at** $t+2$
- **Reward at** $t$
- **Reward at** $t+1$
- **Reward at** $t+2$

$\pi_\theta : S_t \rightarrow A_t$

**Policy-based Deep RL**

Use neural network, with gradients driving the training

Gradient of expected reward with respect to each element of

Goal: Compute parameters $\theta$ that maximize reward

Alternative approach

All policy based approaches
How RL works

Practical challenges

The bad news
These ideas work brilliantly in games, but not in real life

Why not?

1. Large scale
2. Variable scale
3. Complexity
4. Limited compute
5. Explainability requirement
RL in the real world
RL in the real world

One-slide summary of past work

Many systems, many planners, one holy grail

System

Efficiency

Planner

Problem: Systems do not operate in silos ... ... but planners/controllers do

Goal: Build optimal planning & control algorithms that,
1. Operate in real-time (online)
2. Work without human-labelled historical data
3. Adapt automatically to changes in the environment

Example: Port planning
Verma et al., AAMAS 2019

Example: Supply chain
Baniwal et al., ACC 2019

Example: Railway rescheduling
Khadilkar, IEEE ITS 2019
4. RL in the real world
Key takeaways from past work

1. Use domain knowledge
   - to divide the problem into a sequence of tasks
   - to define how system performance is measured

2. Define tasks that can be repeatedly performed to achieve goals (constant I/O size)

3. Build the right fidelity of simulation to compute the effect of actions on the system

4. Use RL only for decisions where the ‘correct’ ones are not obvious

5. Wherever feasible, speed up RL training by seeding with existing heuristics
Goal: Minimise knock-on effects along the railway line, when recovering from a delayed state

Solution: Divide the problem into a sequence of moves
Current work
Planning for robotic parcel loading

Current work

Goal: Maximise the volume packed in containers, using boxes appearing on a conveyor belt

Stream of incoming boxes

ONLINE 3D BIN PACKING

Current box

Stable arrangement

Robot stackable

Rotate the box?

Where to place?

Skip current box?

Container
Current work
Supply chain replenishment

Goal: Minimise supply chain operating costs while maximising key performance indicators

State of the art: Requirements flow from right to left (upstream), while products flow from left to right (downstream)

Solution: Multi-agent reinforcement learning at each node of the supply chain, for automated adaptive response to demands

OPTIMAL NETWORK OPERATION

- Diverse vendors
- Heterogeneous transportation
- Regional warehouses
- Local warehouses
- Reduce wastage
- Avoid stock-out
- Hundreds of stores
- Replenishment decisions
- Transport & labour planning
Multi-Agent Reinforcement Learning (MARL)

Generalisation of Markov Decision Processes to **Stochastic Games**

**Homogeneous Stochastic Games:**
- Number of agents
- Transitions
- States
- Actions
- Rewards
- Discount factor

**Heterogeneous Stochastic Games:**

Can this set of participants in a *system of systems* collaborate effectively?
Concluding remarks

Reinforcement learning = Use of machine learning for decision-making problems

Should be used when it is the best tool for the job:
1. Fast response
2. Systems simulatable but not analytically describable
3. Unknown ‘optimal’ decisions
4. Sequence-dependent rewards

Making RL work for you in real life:
1. Make sure you can simulate your problem, for training
2. Divide large problems into a sequence of repeated tasks
3. Use domain expertise rather than throw it away
4. Build solutions with explanations, not black boxes