A Brief Introduction to Reinforcement Learning

Michele Lombardi
DISI/ALMA-AI – University of Bologna
Three Words for Three Types of Learning

Supervised Learning
...is about learning from someone
Three Words for Three Types of Learning

Supervised Learning
...is about learning from someone

Unsupervised Learning
...is about looking at patterns
Three Words for Three Types of Learning

Supervised Learning
...is about learning from someone

Unsupervised Learning
...is about looking at patterns

Reinforcement Learning
...is about learning by doing
Some Notable Applications

Learning how to play Atari games

https://www.youtube.com/watch?v=eG1Ed8PTJ18
Some Notable Applications

Beating world masters at Go

https://www.youtube.com/watch?v=_OV0Hlj8Fb8
**Some Notable Applications**

Beating professionals at team-based esports

![Game Screen](https://www.twitch.tv/videos/410533063?t=01h32m02s)

<table>
<thead>
<tr>
<th>Human View</th>
<th>AI View</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.006</td>
<td>-1.386</td>
</tr>
<tr>
<td>-0.3154</td>
<td>-0.5425</td>
</tr>
<tr>
<td>3.11</td>
<td>-1.36</td>
</tr>
<tr>
<td>-2.324</td>
<td>2.863</td>
</tr>
<tr>
<td>3.037</td>
<td>-1.361</td>
</tr>
<tr>
<td>-1.387</td>
<td>2.951</td>
</tr>
<tr>
<td>3.023</td>
<td>-0.9395</td>
</tr>
<tr>
<td>2.951</td>
<td>-0.5747</td>
</tr>
<tr>
<td>2.963</td>
<td>-1.303</td>
</tr>
<tr>
<td>2.834</td>
<td>-3.164</td>
</tr>
<tr>
<td>3.127</td>
<td>-1.368</td>
</tr>
<tr>
<td>3.088</td>
<td>-1.366</td>
</tr>
<tr>
<td>2.984</td>
<td>-1.393</td>
</tr>
<tr>
<td>3.037</td>
<td>-1.391</td>
</tr>
<tr>
<td>3.076</td>
<td>-1.438</td>
</tr>
<tr>
<td>-2.412</td>
<td>2.846</td>
</tr>
</tbody>
</table>
Some Notable Applications

Robotics

[Image of a robot performing a task]

Grasp C

real time

autonomous execution

https://www.youtube.com/watch?v=Q4bMcUk6pcw&feature=emb_logo
Ingredients of Reinforcement Learning

In RL there's no dataset, but an environment to play with

At each step $t$:

- We observe the environment state
- We perform an action

Then:

- We receive a reward
- The environment moves to the next state

The goal is to define a policy:

- I.e. what action to choose, based on the observed state
A Simple Example

A food inventory control problem

• Producing yields one 🍏 in the next step
• Producing costs one 💰
• Max two 🍏🍏 storage
• Stored 🍏 spoils in 2 steps 🍏 → 🍏 → 🙁
• Orders require one or more 🍏
• Orders always deplete the inventory
• Un-met orders cost three 💰💰💰
• Orders are not a priori known
A Simple Example

A possible representation
• Steps = opportunities to produce
• Actions = whether to produce or not
• State = inventory situation + current step

Here's a problem instance

How do we determine an optimal policy?
Dynamic Programming

Let's consider all possible choices
Dynamic Programming

...And then go backwards

- For the last step we know the quality of each choice...
Dynamic Programming

...And then go backwards

- For the last step we know the **quality** of each choice...
- ...And therefore the **value** best possible outcome
Dynamic Programming

...And then go backwards

• Then we can do the same for the qualities in the previous step
• ...and the same for the values
Dynamic Programming

Now we just need to pick always pick the best Q

• The Q values define something called a Q-function
• ...And this approach is known as dynamic programming
From Enumeration to Sampling

In practice, enumeration is not viable
But we can circumvent it:
• Every optimal Q function satisfies a certain mathematical relation

\[ Q(s, a) = r(s, a) + \max_a Q(T(s, a), a) \]
From Enumeration to Sampling

In practice, enumeration is not viable
But we can circumvent it:

• Every optimal Q function satisfies a certain mathematical relation

\[
Q(s, a) = r(s, a) + \gamma \max_a Q(T(s, a), a)
\]

- state & action
- reward
- usually: discount factor to handle infinite horizons
- the best Q of the next state
From Enumeration to Sampling

In practice, enumeration is not viable
But we can circumvent it:

• Every optimal Q function satisfies a certain mathematical relation

\[
Q(s, a) = r(s, a) + \gamma \max_a Q(T(s, a), a)
\]

\(Q(s, a)\) is the reward for state & action \(s, a\), \(r(s, a)\) is the immediate reward, \(\gamma\) is the discount factor, and \(T(s, a, \cdot)\) is the best Q of the next state.

• Now the trick is starting from random Qs...

• ...Sample actions and observe rewards...

• ...And adjust the Q function until it satisfies the relation
From Enumeration to Sampling

Since we sample, we can now deal with uncertainty!

Think of:

• Unpredictable orders
• Machine failures
• Uncertain power supply...
Deep Reinforcement Learning

Storing Q values for each state & action is also impractical
...But we can replace them with a Neural Network!
• Way more scalable
• Can handle sensorial information (e.g. images)
...Of course is not as trivial as this ;-)
(D)RL: The Good, the Ugly, and the Bad

We can evaluate some of the advantages of RL:

- Has no need to generate labels
- Can adapt to dynamic conditions

Like humans
(D)RL: The Good, the Ugly, and the Bad

We can evaluate some of the advantages of RL:

- Has no need to generate labels
- Can adapt to dynamic conditions
- Can devote absurd computational resource to learning
- Can process huge amount of data
- Can live for (virtual) hundreds of years

Like humans

Unlike humans
(D)RL: The Good, the Ugly, and the Bad

We can the evaluate some of the advantages of RL

• Has no need to generate labels
• Can adapt to dynamic conditions
• Can devote absurd computational resource to learning
• Can process huge amount of data
• Can live for (virtual) hundreds of years

My favorite:

• If we can provide an environment to play with...
• ...We can think of building a RL system

Like humans

Unlike humans
(D)RL: The Good, the Ugly, and the Bad

There are several caveats:

RL systems need to experiment
- I.e. they must be allowed to make mistakes
- Exploration vs exploitation trade-off

A good problem understanding is necessary
- Choosing the state has an impact on the policy
- Representing states may be practically impossible (hence: approximation)
- How to represent actions? Which DRL variant?
(D)RL: The Good, the Ugly, and the Bad

...And of course, there are plain drawbacks

- RL systems often require very large training times
- They don't require supervision... but what about surveillance?
- RL systems can be (very) opaque!
- They often don't play along well with other decision support techniques

Can some of these spur interesting research? Of course!
A Brief Introduction to Reinforcement Learning

Questions?

michele.lombardi2@unibo.it