Smashing the Bottleneck in EDRL*

a Research Plan

Matthew Farrugia-Roberts

Master of Computer Science
The University of Melbourne

*It stands for:
Expectile
Distributional
Reinforcement
Learning
(not examinable)

[Note: Per-slide timing and *transcript* under each slide, full-talk timing at end**]**

[Timing: 45 seconds (98 words at 130wpm)**]**

[Start of talk]

> *A child learns to grasp, and, eventually, to walk.*
>
> *An algorithm achieves grand-master rating in an ancient board game.*
>
> *Artificial intelligences compete for the high-score in a simulated arcade.*
>
> *A robot learns to grasp, and, eventually, to walk.*

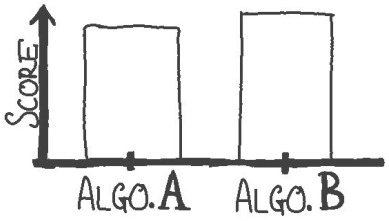
What do all these situations have in common?

They all could involve ***reinforcement learning*****,** a major subfield of modern machine learning that's widely used in practical applications.

Hi! I'm Matt.

I'm studying AI at the University of Melbourne, and today I'd like to share an idea I've been working on for improving the state of the art in reinforcement learning.

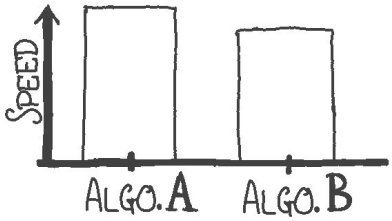
Goals of Reinforcement Learning



Efficacy

Does the AI actually win games?

How far and fast can the robot walk?



Efficiency

How long does it take to learn to win?

How long does it take to walk?

*synthetic data

[Timing: 40 seconds (87 words at 130wpm)**]**

First up, let's agree on what we mean by 'improving'.

Most people measure the success of a reinforcement learning algorithm by the quality of the decisions it makes (judged by their consequences).

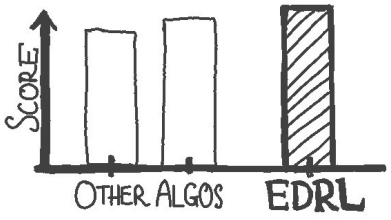
> *Does the AI actually win the board game?*
> *Does the robot actually manage to walk?*

I call this the algorithm's **efficacy**.

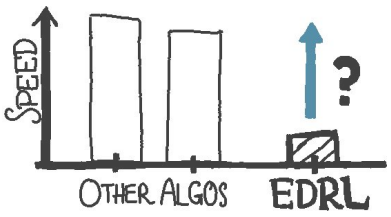
But practical reinforcement learning algorithms also need to be **efficient**.

It's no good if the AI will eventually win the board game every time, *if it takes ages to play a single training game.*

Can We Smash Rowland et al.'s Bottleneck?



Efficacy



Efficiency

EDRL: **E**xpectile **D**istributional **R**einforcement **L**earning (skillful, but *slow!*)

Bottleneck: Solving a statistical sub-problem with an expensive/naive algorithm

Question: *Can we smash this sub-problem bottleneck, and improve EDRL?*

*synthetic data

[Timing: 51 seconds (107 words at 130wpm)**]**

[Note: The transition emphasises the added columns in the plots on this slide**]**

Sometimes these two dimensions are in trade-off.

For example, right now, the main drawback of one of the *most efficacious* reinforcement learning algorithms---an algorithm we'll call EDRL---is that it's *too slow*:

This is an algorithm that schools any human (and most AIs) on the Atari 2600, *but training it pushes even DeepMind's computing resources.*

What's *taking* so long?

EDRL involves a statistical sub-problem which it solves using an expensive numerical optimisation algorithm.

A natural question is (and this is essentially my research question): *Can we smash this bottleneck* if we switch to a faster algorithm for the sub-problem? Can we make EDRL *efficient* as well as efficacious?

My Proposed Study: Overview

Three algorithms (for the sub-problem):

Baseline
Rowland et al., 2019

v.

Alternative 1
Schnabel & Eilers, 2013

v.

Alternative 2
Newey & Powell, 1987

Two stages of investigation & analysis:

- **Stage 1:** Study the algorithms for the sub-problem in isolation, in simple, small-scale proof-of-concept experiments
- **Stage 2:** Combine these algorithms with EDRL and run large-scale, standard reinforcement learning experiments

[Timing: 1 minute and 14 seconds (161 words at 130wpm)**]**

It turns out we don't have to look very far to find alternative algorithms for this sub-problem:

- First, I realised an equivalent problem has been studied before in regression---we have one potential replacement algorithm from there.
- Then, there's a theorem from the eighties which I think we might be able to turn into a second algorithm.
- So, along with our baseline (the numerical optimisation routine), that's three algorithms...

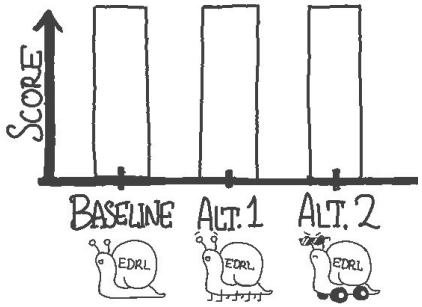
...and my question? *which is fastest* when you plug it into EDRL?

Now. **[pause]**

Comparing these algorithms would be expensive, since standard reinforcement learning experiments use *lots* of computation for all of the training the algorithms need to go through. So I'm going to split this up into two 'stages':

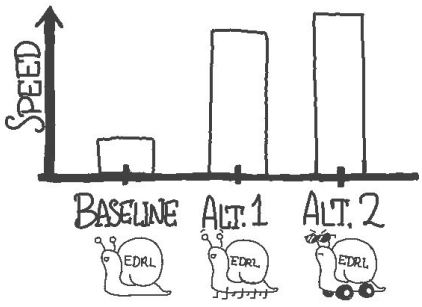
- In the first stage, we'll just make sure that the alternatives work well and are actually faster in isolation.
- Then, if we get a clear result, we'll go ahead to a full-scale reinforcement learning evaluation to see if they work in the full context.

Stage 2: Establishing Improvements Over EDRL



Does efficacy *stay the same*?

Train and evaluate algorithms in standard benchmarks, make sure that they still work.



Does efficiency *improve*?

***Separately**, time* the algorithms (after some training/evaluation) to see if they are indeed faster.

*synthetic data

[Timing: 51 seconds (111 words at 130wpm)**]**

The details are actually pretty straight-forward. But I want to tell you a bit about that second stage.

We're looking for *two things* in our analysis:

- our alternatives should be faster in a training experiment, and
- they shouldn't lose any efficacy.

One problem is, if we want to measure both by timing the *whole training process*, we can't use any tricks like parallelism to speed up the baseline (the original authors did that, because it's *very slow*). We wouldn't get a fair efficiency comparison.

I thought of separating the investigation instead:

We can parallelise the baseline for a feasible *efficacy* experiment, and *then* control for hardware to time the baseline's *efficiency*.

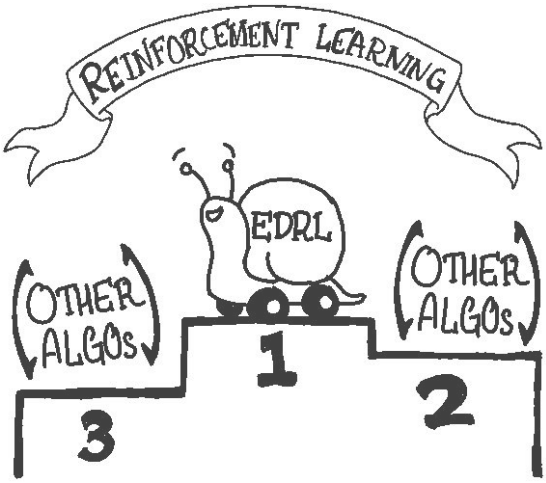
Two Main Contributions

Smashing the bottleneck

> Giving us a state-of-the-art and practical/scalable reinforcement learning algorithm.

Highlighting this part of the statistics/regression literature

> It could be relevant to future reinforcement learning work.



[Timing: 39 seconds (85 words at 130wpm)**]**

Anyway, there are lots of other interesting bits. Just read the plan!

But I'll close by reiterating what I think are two major contributions of this work, if we smash the bottleneck:

- We turn EDRL from *impractically slow* to one of the leading reinforcement learning algorithms *in the world*.
- Also, we highlight the relevance of some work from this corner of the statistics and regression literature, which I think hasn't been acknowledged by the reinforcement learning field, but could have more useful insights to share.

Thank You For Listening

References:

Newey and Powell, "Asymmetric least squares estimation and testing," *Econometrica: Journal of the Econometric Society*, 1987.

Schnabel and Eilers, "A location-scale model for non-crossing expectile curves," *Stat*, 2013.

Rowland, Dadashi, Kumar, Munos, Bellemare, and Dabney, "Statistics and samples in distributional reinforcement learning," in *International Conference on Machine Learning*, 2019.

Image credits:

Accessed 13th November, 2020:

Infant hand by RitaE
(via Pixabay, Pixabay license)
Chess by PublicDomainPictures
(via Pixabay, Pixabay license)
Space Invaders by Coentor
(via Wikimedia Commons, CC BY-SA 3.0)
Robot hand by Greenhill and Elias
(via Wikimedia Commons, CC BY-SA 3.0)

All other images by author

[Timing: 1 second, 1 word**]**

Thanks!

[End of talk]

[Total timing: 5 minutes and 0 seconds, exactly 650 words at approx. 130wpm**]**