# Report RL Self Study

Jens Hartsuiker

February 2024

## 1 Introduction

The aim of this project is to familiarize ourselves with reinforcement learning one slightly covered by our study programs and a field closely related to our favourite one: mathematical optimization.

We studied the field at the hand of this Huggingface tutorial [?]. From which we implemented the following algorithms from scratch ourselves: Q-learning, the Reinforce algorithm, actor critic methods and finally the PPO algorithm.

Our final goal is to apply a learning algorithm to the MsPacman Atari environment included in the gymnasium project [?].

## 2 Problem statement

It is when we implemented the bare PPO algorithm from said tutorial when we ran into our problem. MsPacman moved itself in a corner, to be stuck there until she got eaten by a ghost: we are stuck in a local optimum. We varied various hyperparameters: learning rate, number of model parameters, number of epochs, number of parallel learners, gamma, etc. However this was all to no avail.

## 3 Solution Hypothesis

At this point in time we went back to the drawing board. The bare PPO algorithm learns the average value for a state. We presumed that, especially for the MsPacman environment, this did not allow for enough expressive power within the model. Hence we constructed an algorithm that estimates the values of state actions pairs, adjusting the loss accordingly.

In pursuit of better results still, we also tried modifying the critics' loss function such that at low value estimation discrepancies the total loss does not get outweighed by the actors' loss.

Note that in our algorithms our actors'- and critics' loss share values. We tried severing this link by detaching the advantage vector, but this resulted in worse performance.

**Bare algorithm**

The mathematical formulas for the basic algorithm are the following:

$$Critic\,loss = (\gamma * V(s_1) - (V(s) - reward(a)))^2 \tag{1}$$

Here $V(s)$ denotes the value of state $s$, and $reward(a)$ denotes the reward that taking action $a$ gives while going from state $s$ to $s1$.

$$Advantage = \gamma * V(s_1) - (V(s) - reward(a)) \tag{2}$$

**Action pair version**
The mathematical formulas for the state action pair version are the following:

$$Critic\,loss = (\gamma * max_{a'} Q(s_1, a') - (Q(s, a), -rewards(a)))^2 \tag{3}$$

Here $Q(s, a)$ denotes the value of taking action $a$ in state $s$, and $reward(a)$ denotes the reward that taking action $a$ gives while going from state $s$ to $s1$.

$$Advantage = \gamma * Q(s_1, a_1) - (Q(s, a), -rewards(a)) \tag{4}$$

Substituting the critic loss into the advantage gives us:

$$Advantage = \gamma * Q(s_1, a_1) - (\gamma * max_{a'} * Q(s_1, a') + reward(a) - reward(a)) \tag{5}$$

Which simplifies to:

$$Advantage = \gamma * (Q(s_1, a_1) - max_{a'} * Q(s_1, a')) \tag{6}$$

**Action pair version - own twist**
For this model we kept the advantage the same as the action pair version, however we modified to critic loss as follows:

$$\begin{aligned} Critic\,loss = Max(C * (\gamma * max_{a'} Q(s_1, a') - (Q(s, a), -rewards(a)))^2, \\ C * abs(\gamma * max_{a'} Q(s_1, a') - (Q(s, a), -rewards(a))) \end{aligned} \tag{7}$$

For $C$ we chose a value of 1.25. We also tried values 1.01 and 2, but these did not provide better results.

## 3.1 Other changes

Besides the already existing features from our first trial such as parallel learning, we added new features to our new trials. Examples are kernel initialization, logarithmic subtraction instead of division of probabilities.

## 4 Results

Here we test these algorithms on the Lunar Lander environment, as this environment is less computationally expensive than the MsPacman environment. Thereafter we will train the MsPacman

|  | Average | Min | Max |
|---|---|---|---|
| Bare algorithm | 65.5 | -310.0 | 288.9 |
| State action | 91.8 | -328.0 | 302.1 |
| Own | -21.2 | -296.6 | 261.1 |

Table 1: Reward scores after following optimal policy for 250 epochs.

environment on the best approach.

We present the training curve of the models described above. All these trials are executed with the same hyperparameters:

- Number of episodes: 1000
- Number of epochs: 5
- Number of parallel learners: 10
- Learning rate: 0.0001
- Eps: 0.2

This means that we sample 5 times, update our models and repeat a 1000 times, resulting in a total of 5000 samples.

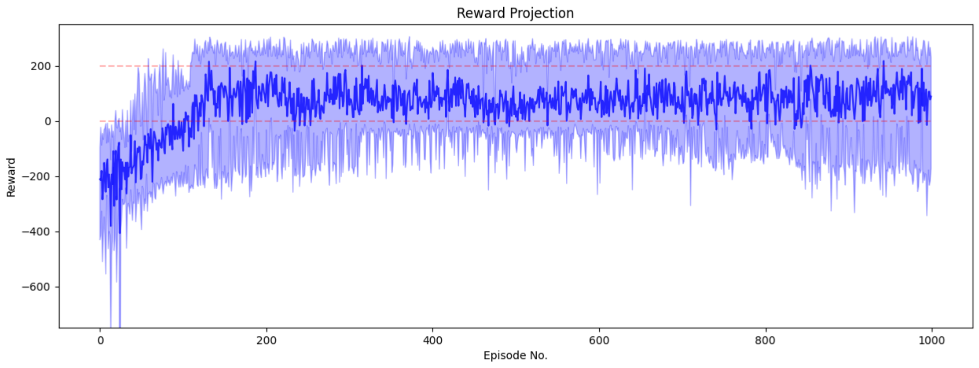
In this plot (1), we show for each trial a plot of the minimum, average and maximum reward of the parallel learners, for each epoch. A score of 200 or higher is considered a success.
From both the learning plots in figure 1 and table 1 we see that the standard state action pair approach works better than our algorithm.

## 5 Conclusion

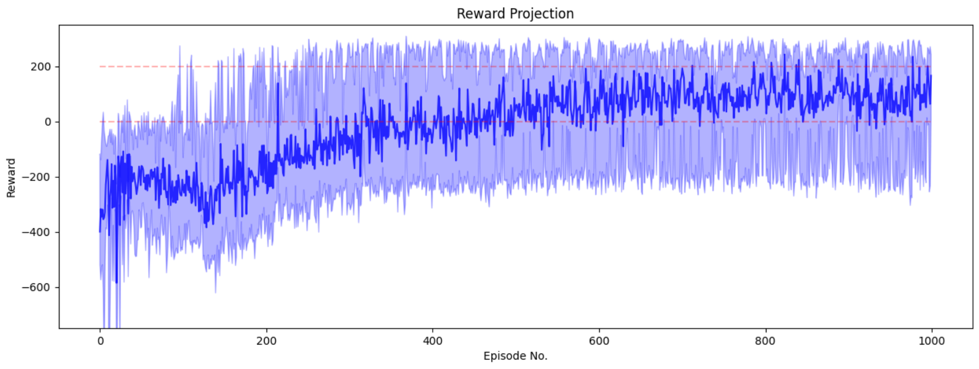
We've seen that our improvements made a significant difference and that our algorithms are capable of learning. Our idea to implement a state action pair value estimation proved more successful than a mere state estimation. Our idea to modify the critic function was to no avail however.

## 6 Future work

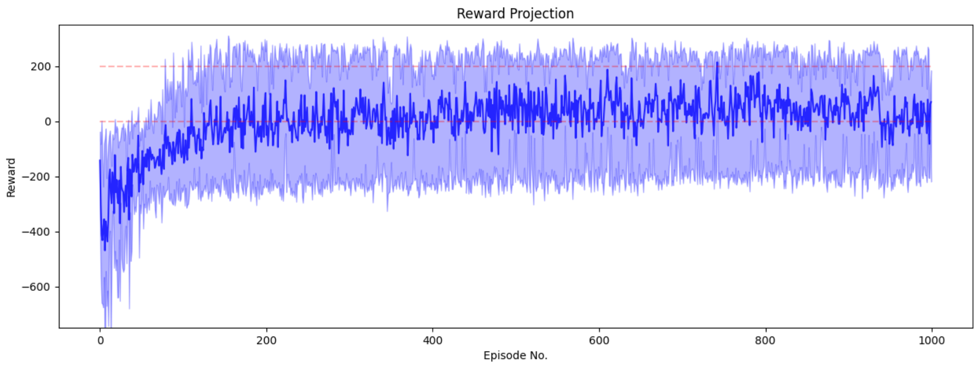
Currently we are running various trials on the MsPacman environment and results will be published here as soon as possible.



(a) The state estimation only trial



(b) The state action pair estimation trial



(c) Our modified loss trial.

Figure 1: A plot of the minimum, average and maximum reward of the parallel learners for each epoch, for each trial.