Improving the Q Learning Algorithm with Rudimentary AI

Burke Brockelbank
(Dated: August 4, 2018)

Neural networks have effectively been taught to play video games. These neural nets have historically been trained through a method of reinforcement learning called Q learning. One of the major challenges with this method occurs in the early stages of training when the quality function is most inaccurate. Here we show a method for skipping past this initial training phase, decreasing the time to convergence. In this method, the neural net first trains on labelled data generated by a sub-optimal, but theoretically simple, artificial intelligence until it is capable of playing the game to a similar effectiveness. It is then trained with Q learning to refine the performance of the AI. This method is applicable to any Q learning scenario where it is easy to concieve an AI to prepare labelled data.

I. INTRODUCTION

The method of training explored in this report is Q learning. Q learning has always had the problem of slow convergence near the beginning of its training, an issue that is inherent its formulation. Q learning is shown to theoretically converge in infinite time with some restrictions on the learning rate in [1] and has proven capable of converging with no assistance in practice, however the amount of computation power and time needed to do this may not always be feasible. Additionally, it is better to avoid the problems with Q learning rather than trust in the theory in case a specific implementation makes it impossible to use.

Another issue with reinforcement learning is that its behavior is highly dependent on how the reward function is specified. Poor choices of reward functions can lead the model to make decisions which are not desired. In [2] the authors describe an interesting example where the specific reward function they used lead to a difference between how they wanted their player to act and how their player ended up maximizing reward.

A. The Quality Function

The formulation of DQ begins with specifying the reward of an action. Suppose that there is a player in a state s who takes an action a. This action will move the player from the state s to s' and also give the player some immediate reward $r(s,a)$. For example, if the player is a robo-advisor that is trading stocks then the state is the current price of all stocks and the action is either to wait, sell a stock, or buy a stock. The reward may be specified as the amount of value gained in moving from the state s to s'. After this the player will take the action a' from the state s', followed by taking the action a'' from the consequent state s'' and so on. The cumulative reward R will include all of these rewards, but to reflect the uncertainty about future actions, a discount factor $\gamma \in (0,1)$ is introduced.

$$R = r_{s,a} + \gamma r_{s',a'} + \gamma^2 r_{s'',a''} + \dots, \tag{1}$$

Suppose we had some ideal policy function π^* that we could consult at every step to give us the ideal action to maximize total reward given a state. Then we define our quality function as,

$$Q^*(s,a) = r_{s,a} + \gamma r_{s',\pi^*(s')} + \gamma^2 r_{s'',\pi^*(s'')} + \dots. \tag{2}$$

Thus, $Q(s,a)$ represents the maximal cumulative reward we can get from the state s given that we are going to perform the action a from that state. We can rewrite this as,

$$Q^*(s,a) = r_{s,a} + \gamma Q^*(s', \pi^*(s')). \tag{3}$$

which is known as the Bellman equation. Additionally, π^* can then be written in terms of Q^* as,

$$\pi^*(s) = \arg\max_a Q^*(s,a). \tag{4}$$

The objective of Q learning is to use a neural net to model this quality function. Let the approximation be Q. Define a policy function analogously to (4),

$$\pi(s) = \arg\max_a Q(s,a). \tag{5}$$

Then there will be some difference between the two sides of (3) called δ.

$$Q(s,a) = r_{s,a} + \gamma Q(s', \pi(s')) + \delta. \tag{6}$$

The loss is then calculated as a function of δ.

In order to train the neural net effectively, some degree of exploration is used. In the case of this report, π is replaced with π_ϵ where $\epsilon \in [0,1]$. π_ϵ has a probability ϵ of returning a random action and a $(1-\epsilon)$ probability of returning π.

B. Monkey Want Banana

This report considers the performance of

C. Challenges with Q Learning

One of the main issues with Q learning is evident in (6). Calculating the loss for performing the action a from the state s, it depends on the next state. Hypothetically, δ could be very small or even zero even if Q does not accurately model Q^* as long as $Q(s, a)$ and $\gamma Q(s', \pi(s'))$ differ from their respective Q^* predictions by the same amount. If Q is inaccurate, like at the beginning of its training, it will converge slowly.

II. METHODS

III. RESULTS

IV. DISCUSSION

[1] R. S. Sutton, A. G. Barto, *et al.*, *Reinforcement learning: An introduction* (MIT press, 1998).
[2] J. Clark and D. Amodei, "Faulty reward functions in the wild," (2017).