A Guide to Probably Approximately Correct Bounds for ~~Neural Networks~~ Reinforcement Learning Algorithms

Thomas Walker

Summer 2023

Contents

1 Introduction

So far we have dealt exclusively in the supervised learning setting. Where we are using the empirical training error as a proxy for network performance on the underlying distribution. In the reinforcement learning case (RL) there is an agent who is learning to make decisions that optimize some reward function. This specified reward function is also a proxy to instil desirable performance into the agent's actions. There are two questions that we would like to be able to answer if we are training an RL agent. Firstly, we would like to ensure that the training process directs the learned policy of the agent to the optimal policy defined under reward. However, this is desirable behaviour only if the optimal policy defined under our proxy reward is similar to the desired behaviour we want from the agent. The PAC RL framework we will discuss can only help us answer the first question. The second question is still open and an active area of research.

2 Preliminary PAC RL Algorithm

Reinforcement learning is formulated using Markov Decision Processes (MDP), which define a trainable agent through a stochastic process. There are different ways to set up the MDP depending on the types of investigations one intends to conduct. As we describe the framework set out in [1] we will concur with their implementation of the MDP. Which involves a discrete time MDP with a finite state space. Formally, our environment is characterized by the tuple

$$\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, i_0),$$

where

- $\mathcal{S} = \{1, \dots, N\}$ is our finite state space,
- $\mathcal{A} = \mathcal{A}_1 \times \dots \mathcal{A}_N$ is the finite policy space,
- $\mathcal{P}$ is the set of transition probabilities,
- $\mathcal{R}$ is the set of expected immediate rewards, and
- $i_0 \in \mathcal{S}$ is the initial state.

At time step t using the above information the MDP functions in the following way,

1. the agent observes the state it is in,
2. based on this the agent takes an action from $a \in \mathcal{A}_i$ using its policy. The state i is the observed state of the agent.
3. The agent then transitions to a new state $j \in \mathcal{S}$ according to $p_a(i,j) \in \mathcal{P}$, and the agent also receives the reward R_t.

We assume that $|R_t| \leq r_{\max}$ for all t, and we are interested in the total discounted reward

$$R_0 + \beta R_1 + \beta^2 R_2 + \dots$$

for $\beta \in [0,1)$. Our intention is to learn a policy that maximizes the total discounted reward. Formally, a policy is a function that assigns an action to a state at each time step. That is, a policy π is a sequence of functions

$$\pi = (\pi_0, \dots, \pi_t, \dots)$$

where at time step t if we are in state i we use $\pi_t(i)$ to determine our action. The policy is called stationary if $\pi_0 = \pi_1 = \dots = \pi_t = \dots$.

Definition 2.1. *For a policy π the expected total return from state i if action $a \in \mathcal{A}_i$ is made and then policy π is followed thereafter is,*

$$W^{\pi}(i,a) = \sum_{j=1}^{N} p_a(i,j) \left(r_a(i,j) + \beta V^{\pi^+}(j) \right),$$

where $\pi^+ = (\pi_1, \pi_2, \dots)$

Definition 2.2. *For a policy π the expected total discounted reward for starting in state i is*

$$V^{\pi}(i) = W^{\pi}(i, \pi_0(i)).$$

We shorten terminology by referring to V^{π} as the value of the policy and $W^{\pi}(i,a)$ as the action value of a at i under π.

Theorem 2.3. *For a given discount factor β, the stationary policy π^* given by $\pi^*(i) = a$ where*

$$W^{\pi^*}(i,a) = \max_{a' \in \mathcal{A}} \left(W^{\pi^*}(i,a') \right)$$

is optimal. Meaning that for any other policy π and $i \in \mathcal{S}$ we have that

$$V^{\pi^*}(i) \geq V^{\pi}(i).$$

This policy is not directly computable as the transition probabilities $p_a(i,j)$ and the expected immediate rewards $r_a(i,j)$ are unknown. It can only approximate these through experimentation. Therefore, one can see how a link to neural network generalization may arise. There is some unknown underlying process that we aim to capture through a learning algorithm which can only utilize proxies of the underlying process. Similar to the supervised case we can call a learning algorithm a PAC RL learner if for any environment $\mathcal{E}$, any $\epsilon > 0$, any $\delta > 0$ and $0 \in \beta[0,1)$ it produces a policy π in time polynomial in

$$|\mathcal{S}|,\ \max_{i \in \mathcal{S}} |\mathcal{A}_i|,\ \frac{1}{\epsilon},\ \frac{1}{\delta},\ \frac{1}{1-\beta},\ r_{\max}$$

such that

$$\mathbb{P}\left(\left| V^{\pi^*}(i_0) - V^{\pi}(i_0) \right| \leq \epsilon \right) \geq 1 - \delta.$$

The learning algorithm proposed in [1] works in the following way,

1. A sequence is initialized in some initial state,
2. the algorithm then explores using a sequence of decision-action steps which we can an episode.

   - An episode consists of M decision-action steps.

3. Once the algorithm has completed a sufficient number of episodes it produces the learned optimal policy.

Let T denote the exploration time, and t be the step number of a particular episode. The algorithm is designed to maximize information capture at each step of the exploration phase. If we let H_T denote the information the algorithm has collected so far, then our exploration policy, $\tilde{\pi}^e$, will take the step that maximizes information capture in relation to H_T.

- Let $m_t[H_T](i,a)$ be the number of times action a has been tried in state i at time t of an episode.
- Let $n_{a,t}[H_T](i,j)$ be the number of observed transitions from state i to state j under action a at time t of an episode.
- Let $R_{a,t}[H_T](i,j)$ be the total reward received on transitioning from state i to j under action a at time t of an episode.

Algorithm 1 PAC RL Learning Algorithm

```
for $i, j \in S, a \in \mathcal{A}_i, t = 0 \to M-1$ do
    $m_t(i,a) \leftarrow 0$
    $n_{a,t}(i,j) \leftarrow 0$
    $R_{a,t}(i,j) \leftarrow 0$
    $\tilde{\pi}^e_t(i) \leftarrow 1$
end for
while $\hat{d}^{\tilde{\pi}^e}_0(i_0) > \frac{2}{1-\beta}$ do
    $t \leftarrow 0$
    $i \leftarrow i_0$
    while t<M do
        Perform $a = \tilde{\pi}^e_t(i)$, receive reward $R_t$, observe the transition to state $j$.
        $m_t(i,a) \leftarrow m_t(i,a) + 1$
        $n_{a,t}(i,j) \leftarrow n_{a,t}(i,j) + 1$
        $R_{a,t}(i,j) \leftarrow R_{a,t}(i,j) + R_t$
        $t \leftarrow t + 1$
        $i \leftarrow j$
    end while
    Compute $\tilde{\pi}^e$ and updated values $\hat{d}^{\tilde{\pi}^e}$.
end while
return Policy $\tilde{\pi}^*$.
```

The Algorithm 1 using an accuracy measure $\hat{d}^{\tilde{\pi}^e}$ to determine when sufficient information has been gathered in the exploration phase to then compute the approximation to the optimal policy. The returned policy is computed as

$$\tilde{\pi}^*_t(i) = a$$

for

$$\tilde{W}^{\tilde{\pi}^*}_t(i,a) = \max_{a' \in \mathcal{A}_i} \tilde{W}^{\tilde{\pi}^*}_t(i,a').$$

Where $\tilde{W}^{\pi}_t$ can be thought of as the empirical expected value under the policy and with information H_T. These are defined in exactly the same way as above except that the transition probabilities are estimated by

$$\tilde{p}_{a,t}[H_T](i,j) = \frac{n_{a,t}[H_T](i,j)}{m_t[H_T](i,a)}$$

and the expected immediate rewards are estimated by

$$\tilde{r}_{a,t}[H_T](i,j) = \frac{R_{a,t}[H_T](i,j)}{n_{a,t}[H_T](i,j)}.$$

References

[1] Claude-Nicolas Fiechter. “Efficient Reinforcement Learning”. In: *Proceedings of the Seventh Annual Conference on Computational Learning Theory*. COLT ’94. Association for Computing Machinery, 1994, pp. 88–97.