Neural Network based Finite Time Guarantees for Continuous Sate MDPs with Generative Model

Harshita Arya[1]

Abstract

This work is based on the approach proposed by Sharma, et al. 2020 in [1], with the key difference being that we take a Neural Network based approach for fitting the function approximation over the state space.We derive from an Online Empirical Value Learning (ONEVaL), a 'quasi-model-free' reinforcement learning algorithm for continuous MDPs. It requires a generative model to sample from but not the full model specification. ONEVaL can compute near-optimal policies and comes with theoretical performance guarantees on sample complexity to achieve a desired level of performance.The algorithm relies on using a fully randomized policy to generate samples that have the -mixing property. The value function is estimated 'empirically' by taking several samples of the next state using the generative model. We finally try to contrast our work with the existing [2], where NFQ employs a Neural Network for the Q-function along with experience replay to enable effective and efficient training that requires fewer interactions with the environment. We will try to contrast both these approaches by evaluating on benchmark problems like CartPole Balancing and LunarLander.

1. Background

When applying reinforcement learning to real-world problems, an important challenge is finding an appropriate representation for the value function. Multi-layer perceptrons are an appealing choice due to their ability to approximate non-linear functions. However, despite successful applications, multilayer perceptrons also come with difficulties. A key issue is that the representation in multilayer perceptrons is global rather than local. This can lead to problems such as instability in learning, lack of generalization, and difficulty in scaling to large problems. To address the limitations of global value function approximations like neural networks, an alternative is to use more localized representations that can exploit the structure of a problem. Developing reinforcement learning algorithms that can effectively leverage both local and global value function representations remains an active area of research. In order to exploit only the good sides of the global approximation through a multi layer perceptron we address the issues with using multi-layer perceptrons for value function approximation is to constrain the influence of each update to the neural network. The core idea underlying our proposed method is simple:

- when making an update at a new datapoint, we also explicitly retain prior knowledge.
- We implement this concept by storing all previous state-action transition experiences in memory. This experience data is then reused every time the neural network representing the Q-function is updated. By reusing prior experiences, we can regularize the network updates to prevent drastic changes and preserve previously learned representations.
- We use β mixing sampling to get a random policy from our Replay Buffer.
- We finally use a Target Network for a fixed iteration to generate our baseline for comparison.

We then try to compare the results in the similar problem sets for NFQ, an algorithm for efficient and effective training of a Q-value function represented by a multi-layer perceptron. Based on the principle of storing and reusing transition experiences, a model-free, neural network based Reinforcement Learning algorithm.

2. Methodology

2.1. OneVAL Approach

We will consider a discounted MDP (X , A, P, r, γ) where X is the state space and A is the action space. The transition probability kernel is given by $P(\cdot|x, a)$ represents the probability of an event occurring given a specific state x and action a., i.e., if action a is executed in state x, the

[1].

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

probability that the next state is in a Borel-measurable set B is $P(x_{t+1} \in B \mid x_t = x, a_t = a)$ where x_t and a_t are the state and action at time t. The reward function is $r : X \times A \to \mathbb{R}$. We are interested in maximizing the infinite horizon expected discounted reward where the discount parameter is γ. Let π denote the class of stationary deterministic Markov policies mappings $\pi : X \to A$ which only depend on history through the current state. We only consider such policies since it is well known that there is an optimal MDP policy in this class. When the initial state is given, any policy π determines a probability measure P(π) . Let the expectation with respect to this measure be E . We focus on the infinite horizon discounted reward criterion. The expected infinite horizon discounted reward or the value function for a policy and initial state x is given as

$$\max_a \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid x_0, a\right]$$

The optimal value function is given as $v^*(x) = \sup_{\pi \in \Pi} v^\pi(x)$, and the policy which maximizes the value function is the optimal policy, π^*.We will assume that we have interactions $(x_1, x_2, \ldots, x_N, x_{N+1}, \ldots)$ generated by a randomized policy π^g, i.e., $x_{t+1} \sim P(\cdot|x_t, a_t)$ where $a_t \sim \pi^g(\cdot|x_t)$. We assume that for any a and x, $\pi^g(a|x)$ is strictly positive. We keep a window of size N which moves forward one step in every iteration. Now these N samples serve as the states for which we will compute our approximate value function and then use function approximation to generalize.

The orginal OneVAL paper performs function approximation using $F_{1:J}^{(\theta)}$. Instead we take a Neural Network based approach for our function approximation technique.

$$\hat{Q}(s, a; \theta) = \text{NN}(s, a; \theta)$$

Our Neural Network consists of one Hidden Layer and takes input the number of observation and gives output the number of states. We use ReLU as our Activation function. $f(x) = \max(0, x)$. β random sample transitions are picked from our Replay Buffer and ϵ greedy policy based action is picked. We then compare our model with the Target Network whose parameters are kept constant per epoch. We compute out Loss using SmoothL1Loss

$$\text{SmoothL1Loss(x)} = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

We compute the loss using the following function

$$\frac{1}{n} \sum_n \sum_j \min\left(\hat{Q}(s, a; \theta) - v_b(x_n)\right)$$

2.2. Neural Fitted Q Iteration Approach

The basic idea underlying NFQ is the following: Instead of updating the neural value function on-line (which leads to the problems described in the previous section), the update is performed off-line considering an entire set of transition experiences. Experiences are collected in triples of the form (s, a, s') by interacting with the (real or simulated) system. Here, s is the original state, a is the chosen action, and s' is the resulting state. The set of experiences is called the sample set D.

The consideration of the entire training information instead of on-line samples has an important further consequence: It allows the application of advanced supervised learning methods that converge faster and more reliably than online gradient descent methods. Here, we use Rprop , a supervised learning method for batch learning, which is known to be very fast and very insensitive with respect to the choice of its learning parameters.

$$v = \rho v + (1 - \rho) g^2$$
$$\theta = \theta - \frac{\eta}{\sqrt{v + \epsilon}} \cdot g$$

The latter fact has the advantage that we do not have to care about tuning the parameters for the supervised learning part of the overall (RL) learning problem. I have derived some ideas from the implementation https://github.com/seungjaeryanlee/implementations-nfq. This project visualizes the Cost by rewarding 1 or 0 vlaues. Therefore its visualizations differ from the previous approach.

2.3. My Contributions

We used the OpenAI Gym which is a widely-used toolkit for developing and comparing reinforcement learning algorithms. It provides a collection of environments where agents can interact and learn. We specifically used the Cartpole Balancing and Lunar Lander Environments in the Continuous setting. Our addition to the project were:

- Implementation of a different function approximation technique using Neural Network in the OneVAL paper.
- Implement the OneVAL approach using Continous Cart Pole and Lunar Lander Approach.
- Implement the Neuro Fitted Approach for Lunar Lander Setting. The original codebase was tuned for Cart-Pole setting.
- Compare the two approaches

2.4. Implementation

The OneVAL approach for our algorithm is sown below

Algorithm 1 ONEVaL Algorithm

```
1:  Input: Sample sizes $N$, $M$, $J \geq 1$
2:  Input: Initial seed $v_0$
3:  Input: Interactions from policy $\pi_g$: $(x_1, x_2, \dots, x_N, x_{N+1}, \dots, x_{2N-1}, x_{2N}, \dots)$
4:  repeat
5:      for $k = 0, 1, 2, \dots$ do
6:          Select samples $(x_n)_{n=k+1}$ from given interactions
7:          for each $n$ and action $a$ do
8:              Sample i.i.d. next states $x_{xi}^{n,a} \sim P(\cdot|x_n, a)$ for $i = 1, 2, \dots, m$
9:          end for
10:         Empirical value iteration: $v_{b,k+1}(x_n) = T_b^M v_k(x_n)$ for each $n$
11:         Function approximation: $v_{k+1} = \Pi F_b v_{b,k+1}(x_n)$
12:     end for
13: until termination condition is met
```

We first selects the samples at which we compute approximate value function. We then samples the next states for a given state and action.This is not difficult when we have a generative model for various learning tasks, for, instance, our Open AI Gym Simulation environment. Next we computes the approximate labels which gets generalized via function approximation.

2.5. Simulations

2.5.1. Cart Pole Environment

The CartPole environment in its classical form is a simple physics simulation provided by OpenAI Gym, typically defined in a discrete setting where actions are discrete values (pushing the cart left or right). However, in a continuous setting, the environment parameters change to allow for a continuous action space. Observation Space:In the classical discrete setting, observations usually consist of four values: cart position, cart velocity, pole angle, and pole angular velocity.In the continuous setting, these observations might remain the same but could be represented as continuous values within a certain range.Action Space:In the classical discrete CartPole, the action space typically consists of two actions: push the cart left or right.In the continuous setting, this space becomes continuous, allowing for a range of actions rather than discrete choices. For instance, applying force to the cart with varying intensities rather than binary left/right actions.Environment Dynamics:The physics of the CartPole environment might remain consistent in terms of balancing the pole atop the moving cart.However, the continuous action space allows for a more nuanced control strategy, potentially leading to more stable pole balancing.

It is observed that OneVAL algorithm gives a stable result over the given iterations and as we go through more iterations such as in upto 10k. We see the total average reward also increases. For the Neuro Fitted Q iteration our algorithm seems to give a result in a speedy fashion and the over all cost also seems to be minimized.

2.5.2. Lunar Lander Environment

The Lunar Lander environment, available in OpenAI Gym, is a classic reinforcement learning environment where an agent controls a lunar lander attempting to land on the moon's surface. The goal is to navigate the lander to a safe landing zone while conserving fuel.Observation Space:Observations typically include information about the lander's position, velocity, orientation, angular velocity, and information about the legs' contact with the ground.Action Space:The action space usually consists of discrete actions such as firing the left engine, right engine, main engine, or doing nothing. A continuous version of the Lunar Lander also exists, offering a continuous action space where the agent controls the throttle of two engines and the direction of the lander's thrust.Rewards:The lander receives rewards based on its actions and position. Positive rewards are given for successfully landing in the target zone, while penalties are incurred for crashing or using excessive fuel.

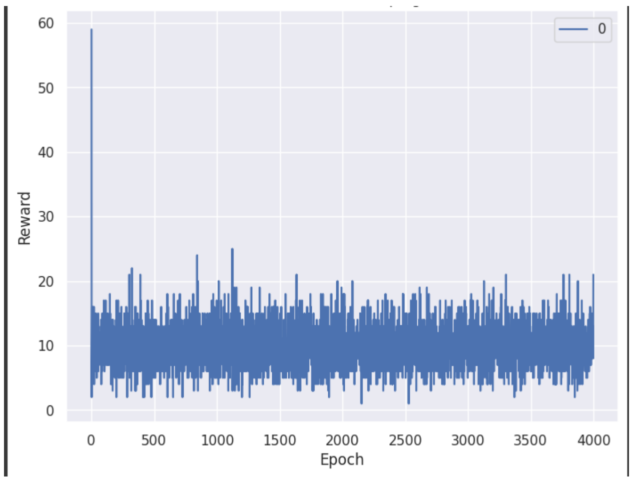

Figure 1. Graph of the OneVAL Algorithm for getting the Rewards for Lunar Lander problem and for 4000 epochs

Both Algorithms see to give promising results as per Figure 1 2 and 3. But Similar trend is observed that OneVAL is Stable over time. As in the graph oscillated around the optimal reward over the further epochs. Whereas the Neuro Fitted Algorithm gave fast result.

3. Conclusion

We therefore conclude that OneVAl is a stable online Learning algorithm that gives promising result for tasks like

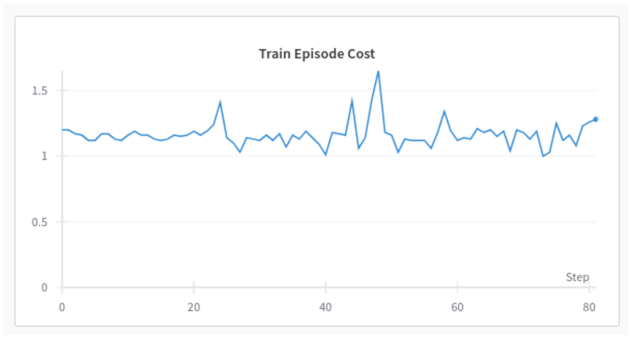

Figure 2. Graph of Neuro Fitted Q Iteration for Lunar Lander for the Training Episode Cost

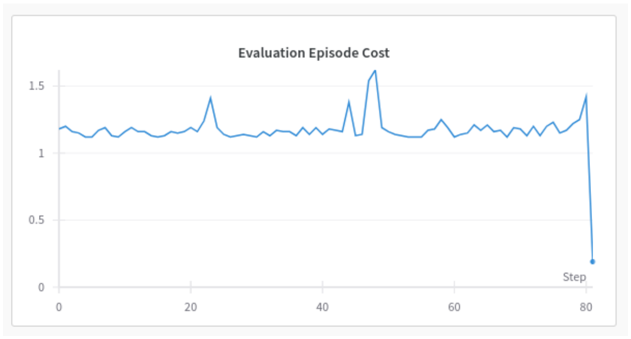

Figure 3. Graph of Neuro Fitted Q Iteration for Lunar Lander for the Evaluation Episode Cost

Robotics Control where stability is a big issue.Online RL is crucial for robots and autonomous systems to adapt to dynamic environments, learn from real-time sensor data, and make decisions in real-world scenarios.These algorithms are also used in control systems for adaptive control, where systems need to adapt to changing dynamics or uncertainties without requiring pre-collected data. Whereas Neuro Fitted ALgorithm is suited for Offline Learning. Storing and reusing all transition experinces, the neural learning process can be made very data efficient and reliable.Additionally, by allowing for batch supervised learning in the core of adaptation,advanced supervised learning techniques can be applied that provide reliable and quick convergence of the supervised learning part of the problem. NFQ allows to exploit the positive effects of generalisation in multi-layer perceptrons while avoiding their negative effects of disturbing previously learned experiences.

4. Citations and References

[1] Hiteshi Sharma and Rahul Jain. Finite Time Guarantees for Continuous State MDPs with Generative Model. In 2020 59th IEEE Conference on Decision and Control (CDC).

[2]Martin Riedmiller. Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In ECML 2005.

[3] Hiteshi Sharma, Mehdi Jafarnia-Jahromi and Rahul Jain. Approximate Relative Value Learning for Average-reward Continuous State MDPs.

[4] Hiteshi Sharma, Rahul Jain and Abhishek Gupta. An Empirical Relative Value Learning Algorithm for Non-parametric MDPs with Continuous State Space.

[5] Boyan and Moore. Generalization in reinforcement learning: Safely approx-imating the value function. Advances in Neural Information Processing.