National College of Ireland

Optimization of Actor Critic Policy in Continuous Action Space

MSc Research Project
MSc in Data Analytics

Simanta Sarkar
Student ID: x18201148

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Simanta Sarkar
Student ID:	18201148
Programme:	MSc in Data Analytics **Year:** 2019-2020
Module:	MSc Research Project
Lecturer:	Dr. Muhammad Iqbal
Submission Due Date:	17/08/2020
Project Title:	Optimisation of Actor-Critic model in Continuous Action space
Word Count:	6826 **Page Count: 20**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Simanta Sarkar

Date: 17/08/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Optimization of Actor Critic Policy in Continuous Action Space

Simanta Sarkar
18201148

Abstract

The implementation of Reinforcement learning algorithms has made a huge impact on various problems where no existing methodologies has succeeded in control task and make decision. In this paper we are implementing a hybrid algorithm to virtual self-driving car through collating the Actor-Critic and Proximal Policy Optimization (PPO) methods to introduce a continuous control tasks for locomotion of cars. Successful locomotion of a self-driving car can be achieved through angular movements of the steering by understanding the changes in environment where the actions like to take turns smoothly or throttle maps to continuous action space. The policy which maps input received from the sensors which causes change of action in cars is upgraded to achieve rewards. Due to these upgraded techniques the general policy-based methods have been improvised by the Actor-Critic method. The primary purpose of the research is to study the performance of the modified policy optimization techniques which enhances the interaction of the agent with the environment resulting in improved rewards in comparison with other policy-based methods. The testbeds used for the implementation of the modified algorithm are Cartpole and MountainCarContinuous. The modified actor-critic algorithm has yielded consistent policy update reducing the risk of learning a sudden irreversible bad policy.

Keywords: Reinforcement learning, Machine learning, Policy Gradient, Actor-Critic, PPO.

1 Introduction

The advancement in Deep Learning through extensive research work in Convolution Neural Networks (CNN) it is an easy task to detect or recognize the objects of the surroundings. However, the major challenge lies in learning the driving policy to imitate a human driver in steering, accelerating, applying brakes, taking turns, overtaking and many more sensing the objects in the environment.

In restraint task the reinforcement learning(Mnih et al., 2015) approach has proved success for several instances because of its proficiency in plaining the action in an environment. The action space for an agent in Reinforcement Learning is dissected into discrete and continuous. The discrete action space is a distinct finite set of actions. On the contrary, a real-valued vector represents the continuous action space. Application of discrete action space in Reinforcement Learning is constrained to simple video games(Mnih et al., 2013) or board games. Most of the control task problem results in outputs lying in the real-valued line of the continuous action space. It is a simplified task to map input to the

corresponding output space of continuous action. However, continuous control task demands for sophisticated treatment because of the infinite numbers of actions in the continuous range.

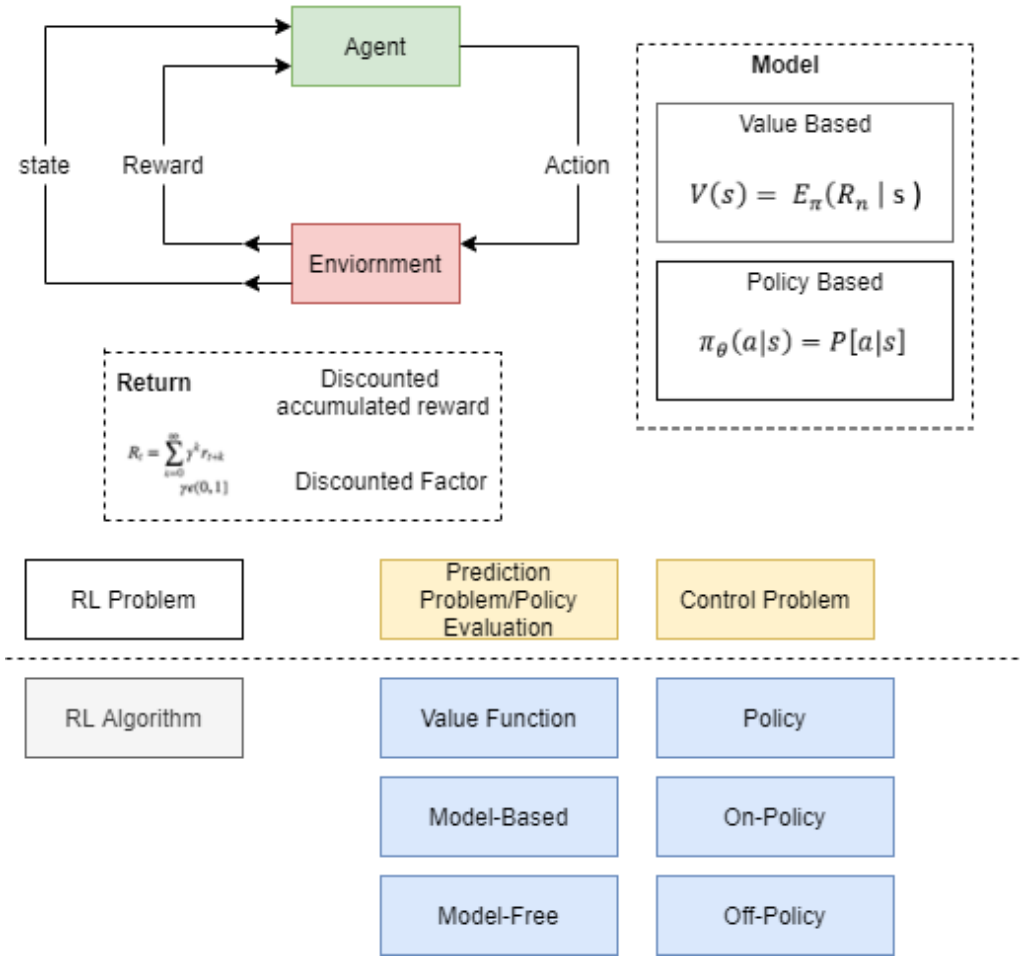


Figure 1: Reinforcement Learning cycle.

In Reinforcement Learning, specific terminologies require attention to understand the problem:

State and Observation – A depiction of the agent or environment is termed as the state. An observation is the resultant of the interaction between the state and agent.

Action space – The environment that showcases the observation through the synergy between the state and the agent is termed as action space. Action spaces can either be continuous with real dimensional or discrete with bounded vector spaces.

Policy – The behavioral aspect that drives the mapping of action to the state is policy. A policy can be either deterministic or stochastic in nature.

Trajectory – It is the correlation between state and action.

Reward and Return – Th evaluation given to an agent post execution of an action in each state is the reward. Over a path the agent majorly focuses in maximizing the reward.

Value function – The function takes the current state-action pair as the input values resulting in total expected reward.

The paper introduces a hybrid approach of Actor-Critic and Proximal Policy gradient to optimize the rate of policy update in Reinforcement Learning Cycle. In the Actor-Critic policy gradient algorithm the critic selects the appropriate action or state value function and thereby assist the actor to perform the policy update. The Proximal Policy Optimization algorithm adopts its policy of action in accordance with the stochastic policy by considering

sampling actions and thus decides on the optimum policy to ensure the highest expected return.

The method is implemented on the control space of OpenAI Gym[1] like the Cartpole and MountainCarContinuous as the algorithms based on policy gradient have been tested to perform well in continuous action space. The implementation have resulted in consistent policy update to obtain a high reward compromising on the learning rate.

1.1 Motivation

The primary purpose of a self-driving car is to reach a destination without any mishap through learning the environment with the assistance of automated sensors without the human interference. Efforts have been made by the researchers to build a self-driving car. Successful contribution has been made only in regard to discrete action space. However, for real-world implementation it is important to train the car effectively and efficiently in continuous action space for successful driving process including taking turns, avoid obstructions, accelerate and decelerate when required. The need for this has motivated the research work to modify the existing algorithm which has improved learning performance in the continuous action space.

1.2 Research Objective

"Can a hybrid of actor critic and proximal policy gradient be used to reduce the rate of policy update in Reinforcement Learning cycles to optimize the consecutive policy update with reduced variance between the two policies in a continuous control environment?"

- To modify the actor-critic algorithm for consistent policy update to prevent the agent from learning sudden bad policy posing a threat to recover from the state.
- To assess the efficiency of the algorithm in continuous action space.

2 Related Work

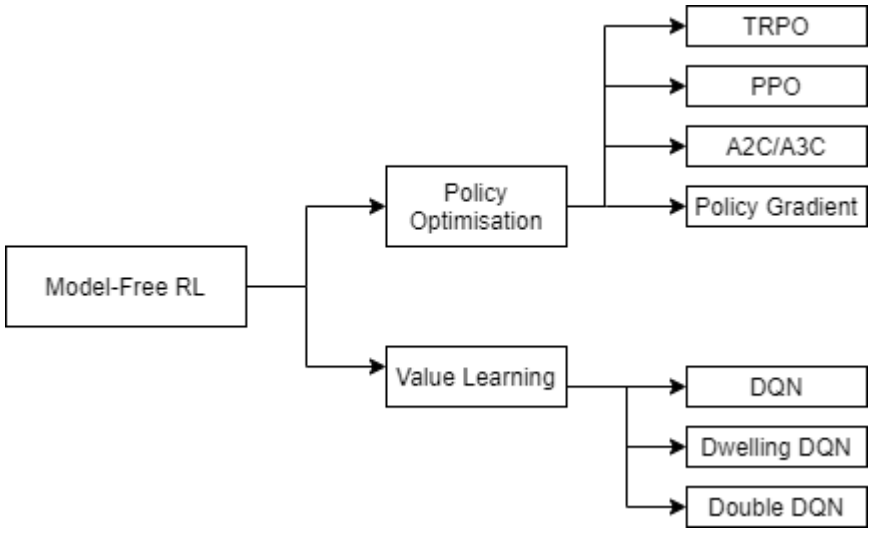


Figure 2: Reinforcement Learning algorithm classification.

The reinforcement learning algorithms can be explicitly classified into Model-based and Model-free algorithm(Com, n.d.; Nagabandi et al., 2018) depending on the knowledge of the

[1] https://gym.openai.com/envs/#classic_control

agent on the architecture of the environment. The algorithm referring to the agent's learning of optimal actions implicitly by following an activity pattern through selection of action from the set and experiencing subsequent state and the corresponding reward is termed as Model Based algorithm. The model can potentially forecast the consequence of an action facilitating in learning the optimal policies through interaction with the environment. On the contrary, in Model Free algorithm the agents hold a chance to master the environment through experience and the estimation of the optimal policy can be performed with no dependence on the estimation of the dynamics of the environment.
The model-based algorithm can effectively use fewer samples to learn the dynamics of the environment preventing overfit through conventional implementation of the Bayesian models.

The reduced complexity in practical application and the adaptability in modification of the Model-free algorithm outperforms the Model-based algorithm. Some of the widely used model free algorithms are discussed below.

2.1 Q-Learning

Q-learning can be explained as off-policy algorithm of reinforcement learning as the Q-function(Kappen, 2011) seeks to learn the policy from the action disjoint from the current policy with the purpose to maximize the total expected reward. The function inherits the properties of Bellman Equation. The algorithm proposed by Watkins is commonly used to explain the Markov Decision Process optimally. However, the algorithm overstates the action-value leading to deficient performance in stochastic Markov Decision Processes.

2.1.1 Deep Q-learning

Deep Q-Learning(Hester et al., 2018; Lillicrap et al., 2016; Mnih et al., 2013) aims at collating the Q-learning algorithm with deep neural network to approximate the Q-values for a state-action set. The algorithm has distinguished contribution in playing Atari games, policy search for robot engine control, strategic policies coupled with searching resulted in defeat of domain expert in Go game. The effectiveness of the algorithm lies in exploitation of scalability and deep learning efficiency (Ansehel et al., 2017). Algorithm for real world problems like self-driving cars, call centers, recommendation system etc is constrained as large number of iterations are required to learn the best policy.

2.1.2 Double Q-Learning

In order to prevent the estimation of same weights in target Q-value and expected Q-value leading to selection of overestimated action value, in Double Q-learning(Van Hasselt et al., 2016) bi-valued functions are learned resulting in two sets of weights w and w'. To train the network randomly experiences are assigned to refurbish either of the two value-functions. For every update, one set of the hyperparameter regulate the policy and the complementary set determines the value for the greedy policy. In (Van Hasselt, 2010) proved that Double Q-learning achieves improved policies.

2.1.3 Prioritized Experience Replay in Deep Q-Learning

In reinforcement learning the agents performs additive update of the parameters of the policy and value function while accounting for a series of experience. The algorithm renounces the data post every update resulting in two concerns a) the independently and identically distributed assumptions of algorithms based on stochastic gradient are disregarded due to high correlated updates, b) rampant forgetting of some rare learned experience leads to loss of potential information. Prioritized Experience(Schaul et al., 2016) Replay algorithm address this issue. The replay buffer reserves in tuples facilitates in disruption of the correlation through combining the recent and prior experiences as mentioned in (Horgan et al., 2018). The number of experiences is reduced by the experience replay compromising on computation and memory.

2.1.4 The Dueling Network Architecture

Reinforcement learning algorithms use classic neural networks mostly. The model-free reinforcement learning algorithms in the duelling network architectur(Hester et al., 2018; Wang et al., 2016) is proposed for modernization. The architecture explicitly de-couples the predicted Q-value estimator in state values and the results in two distinct estimates through fully connected two streams working as single Q-neural networks with shared convolutional feature learning component. So additional monitoring is not required. The state value function is updated with every action resulting in improved state-value estimation. Instinctively the network is probable to learn to prioritize the state value without knowing its impact. The architecture can be used to replace the unit-stream networks in popular algorithms like Deep Q-networks. Researches have evaluated duelling architecture implementation using Atari 2600.

2.2 Policy Gradient

The objective of the Policy gradient algorithm(Sehnke et al., 2008; Silver et al., 2014) is to formulate the probability distributions of the state dependent actions using neural network. Interaction of the agent with the environment decides best parameters. Policy gradient through iterative process gives best rewards. Algorithm divides into Deterministic and Stochastic Policy. The deterministic policy(Barth-Maron et al., 2018; Silver et al., 2014) utilizes a value-function to map the action with the corresponding state for a deterministic environment whereas a probabilistic state dependent action is generated by the stochastic policy for a partially perceptible environment. The major downside of the algorithm lies in high variance in estimation of policy gradient resulting in reduced convergence rate. Parameter-based exploration is one of the novel methods to reduce the gradient variance. Parameter-based exploration aims at generating low variance gradient and initiate stochasticity. The algorithm showed significant improved results over the others the real-world applications.

2.2.1 Actor Critic

The critic only and actor only method is improvised through the actor critic method by uniting the two. The substantial variance is removed by including critic factor(Grondman et al., 2012). The main purpose is to update the value function by analyzing optimum policy

fooled by estimating it and update policy parameter through value function. Off policy variant modified by on policy and learning process is limited thus providing an opportunity to learn other policies (Degris et al., 2012).This paper is using off policy and it has actor and critic as the two learning agents. Approximation and updating of policy parameters are done by actor and uses function value for updating the policy. The critic has requirement of four parameters: λ and three step parameters where the actor has the requirement of αu. (Mnih et al., 2016; Xiao et al., 2019)has proved that has standard and minimized errors. Asynchronous Advantage Actor Critic(A3C) and Advantage actor critic (A2C) could be applied on actor critic agent.

Advantage actor critic concentrate on synchronized updating of global network and asynchronous advantage actor critic concentrated on parallel training. In parallel training learning process is done all together in AC3 and it helps in exploring many parts of environment. In timeframe, training of various threads leads to updated parameters correlation. Parallel training helps in minimization of training time as it has a linear relation with parallel thread. The methodology can be applied with policy based, discrete and continuous and value based has be proved successfully. Implementation of asynchronous in Atari domain resulted in faster training. A2C has been used to resolve inconsistency in A3C and due to synchronous method A2C results in accelerated convergence and cohesive in training.

The drawback of Deep Q-learning is overestimating the standard policies and normal value which gets from error function approximation task. To resolve the overestimation (Fujimoto et al., 2018)has proposed an algorithm which gain minimum value among set of critics. To introduce update on lagged policy overestimation of target and bias is build and this result in minimized errors and better performance. The discrete action in actor critic is observed due to overestimation as the updating of policy is performed by using a gradient descent.

The algorithm in actor critic has failed in many instances even though the method was successful in many dimensions. Due to divergence of model without any target networks the results have shown huge variance in expected value and results in bad policy. The critic and actor's transaction results in learning process failure. So, the policy network must be updated at decreased frequency than value-based network for reducing the error previous to policy updates till it is decreased to maximum.

2.2.2 Continuous Control with Deep Reinforcement Learning

Deep Q-network successfully address problems having inputs with high-dimension space that maps to low-dimension discrete action space. Discretizing the continuous action space to implement the algorithm is opposed with curse of dimensionality making it difficult to train the network(Chou et al., 2017). The Model-free actor-critic off-policy algorithm has proven success in learning policies belonging to the continuous high-dimension action space implementing deep function estimators. The asynchronous advantage actor-critic (A3C) permits for non-synchronously training and updating the network policies with multiple parallel CPU cores. The Gaussian Policy used in continuous stochastic task control problems induce inescapable estimation error because of boundary of the action space resulting in slow training process. But there are limitations exists for this known as beta policy. The

advantages of Beta policy are that it does not induce any bias and have high converging speed. It is compatible with algorithms like TRPO and actor- critic, respectively.

2.2.3 Deterministic Policy Gradient Algorithms

Deterministic Policy Gradient(Silver et al., 2014) is explained as off-policy actor-critic methodology which is an improved replacement of stochastic policy. The major advantage of the functional form is its powerful potential of estimation. This algorithm is ideal to enhance the continuous control task is that the actor network update is dependent on the learning process of critic. The functional mode is the predicted gradient of the action value function. Without integrating the actor space, the actor update can be improved through enhanced learning procedure of critic. The deterministic policy ($\mu\theta$: S $\rightarrow$ A) and the variance σ can be varied to get the parametric stochastic policy ($\pi\mu\theta,\sigma$). The stochastic policy can be compared with the deterministic policy when $\sigma = 0$, whereas when σ is limiting to 0 the stochastic policy gradient gets transformed to a deterministic gradient. The rampant change of policy gradient around the mean challenges the estimation of the stochastic policy gradient. In this regard the distributional adaptation of the critic update is proved to obtain a stable critic learning procedure as the randomness due to inherited factors can be modelled in continuous environment. The Distributional Deep Deterministic policy gradient algorithm (Barth-Maron et al., 2018) collated with prioritized experience replay and N-step returns results in improved performance when implemented on difficult obstacle-oriented locomotion tasks.

2.2.4 Proximal Policy Optimisation Algorithms

Proximal Policy Optimisation algorithm(Schulman et al., 2017) overrides the TRPO with respect to simplicity in implementation. Additionally, the practicality in application is enhanced due to the capped ratio between the old and current policy distribution constraining the distance between the old and current parameters θnew and θold respectively. The algorithm maintains two neural networks for the prior policy and the current policy respectively. The prior policy is used for sampling purpose. On the other hand, the current policy is subjected to precision. The method monitors for the difference between the two consecutive policies through the capped estimated advantage function. Additionally, the advantage function can be optimised through derivation of first order. Experiments have illustrated that PPO have outperformed several novel algorithms when applied on Atari games and simulated mobile robots.

Implementing two estimators in Prioritized experience replay the mentioned challenges are traded off by Double Q-learning. Parametric stochastic policy (Chou et al., 2017)algorithm can also perform unbiased estimation of total expected reward. However, the limitation due to high variance remains undressed because of the immediate actions of the current action. Novel actor-critic algorithm addresses the problem of high variance between the consecutive policies through injecting a bias which might hinder the convergence in case of large sample. PPO algorithm performs the policy update through introduction of a capped ratio between the old and current policies. Several research works have illustrated significant contribution of actor-critic and PPO when implemented on Atari games.

3 Research Methodology

Reinforcement learning algorithm aims at learning an environment through the agent's interaction. The agent is trained to learn the maximised rewarding policy (Henderson et al., 2017) from the consequences of the past actions explained as exploitation and through exploring new possibilities. Through assignment of the numerical reward to the agent it learns to select the actions that results in maximized expected reward. The reinforcement algorithm is based on the Temporal Difference methodology(Degris et al., 2012) where successive states are considered followed by resulting rewards.

$$s_n, r_{n+1}, s_{n+1}, \ldots\ldots, r_N, s_N \quad (1)$$

The expected total reward for the agent's attainment of a state say s_n can be mathematically expressed as,

$$R_n = r_{n+1} + \gamma^1 r_{n+2} + \ldots\ldots\ldots + \gamma^{N-n-1} r_N \quad (2)$$

Where $\gamma < 1$ is explained as *Discount Factor*. The numeric value of the reward is directly proportionate to the importance of the action taken. Additionally, the algorithm assumes that the state value $V(s)$ can be equated to the expected total return.
Mathematically,

$$V(s) = E_\pi(R_n \mid s_n = s) \quad (3)$$

Where the π is the action policy. The parametric policy function (parameter Θ) formulates the probability distribution of action conditioned on the state which can be explicitly written as,

$$\pi_\theta(a|s) = P[a|s] \quad (5)$$

The implemented modified algorithm is based on the Policy Optimisation methodology. The Q-learning algorithm is juxtaposed with the Policy optimisation algorithm which aims at directly optimising the policy overcoming the convergence and stability issue of the former algorithm(Sehnke et al., 2008). The relative instability of Q-learning algorithm is because of collated impact of bootstrapping and approximation function.

The policy gradient algorithm utilizes the gradient ascent termed as *Policy Score Function* (Schulman et al., 2017) to obtain the maximised parameter value Θ resulting in optimum policy. The Policy score function is equivalent to the expected return for the particular policy. Mathematically the score function can be defined as,

$$J(\theta) = E_{\pi\theta}[\sum \gamma r] \quad (5)$$

Steps to obtain the optimum policy:

1. Formulating the Policy score function.
2. Maximisation of the parameter Θ through gradient ascent resulting in improved sample of good actions.

Based on the research aim and the environment for implementation the ideal policy optimisation methodology is selected amongst the three novel optimisation technique. Here, a detailed discussion of the methodology involving average state value and average reward value at each time stamp will be explained as continuous environment is selected for the implementation of the research work.

Average State-Value

The Policy Score function is expressed in terms of the weighted state value. Each state values are weighted by the probabilistic recurrence of the respective state.

$$J(\theta) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s)$$
$$= \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} \pi_{\theta}(a|s) Q^{\pi}(s,a) \quad (6)$$

Here, $d^{\pi}(s)$ explains the stationary distribution of the state following the Markov Chain for the policy π_{θ}. After a time point the state gets stabilized hence the term *stationary distribution*. Mathematical expression,

$$d(s) = \frac{N(s)}{\sum_{s'} N(s')} \quad (7)$$

$N(s)$ represents the occurrence frequency of respective states and $\sum_{s'} N(s')$ is the cumulative frequency of all the explored states.

Average Reward

The policy score function is defined in terms of average reward at each time stamp aiming to obtain the maximum expected reward.

$$J_{avR}(\theta) = E_{\pi\theta}(r)$$
$$= \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} \pi_{\theta}(s,a) R_s^a \quad (8)$$

Here, $\sum_{a \in A} \pi_{\theta}(s,a) R_s^a$ is the probability distribution of the action taken where $\pi_{\theta}(s,a)$ is the conditional probability of the action 'a' given a state 's' and R_s^a is the resulting reward for the action taken.

The Policy score function is formulated to assess the value of the policy followed by maximisation of the policy parameter Θ to acquire the optimum policy. The gradient ascent is employed on the policy score function for computing the policy gradient ascent to update the parameter. For the computational purpose the score function is equated to the cumulative expected return for the given policy.

$$J(\theta) = E_{\pi}[R(\tau)] \quad (9)$$

where $\tau = (s_0, a_0, s_1, a_1, s_2, a_2 \ldots)$ *and* $R(\tau)$ *is the cummulative expected return.*

However, the averaging methodology is posed with several short comings. There is a combined effect of change in action and the resulting reward and the following state on the variation of the parameter Θ. In this course the parameter change impacts the distribution of the action and the resulting state. The challenge prevails in the approximation of the policy gradient with respect to the parameter when the distribution of the state impacted by the parameter change. On the contrary computation of the effect on the selection of action is comparatively easy. The Policy gradient theorem successfully addresses the limitation as the differentiation of the score function is independent of state distribution. The gradient ascent can be mathematically expressed as follows:

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \sum_{\tau} \pi(\tau;\theta) R(\tau)$$
$$= \sum_{\tau} \nabla_{\theta} \pi(\tau;\theta) R(\tau) \quad (10)$$

The above-mentioned gradient is challenged by probability distributional outcome of the stochastic policy to perform differential calculus. Further modification is performed on the gradient through logarithm transformation to simplify the mathematical computation. The transformed expression is given below:

$$\nabla_{\theta} J(\theta) = \sum_{\tau} \pi(\tau;\theta) \nabla_{\theta} (log \pi(\tau;\theta)) R(\tau) \quad (11)$$

Since, in the research work continuous environment is considered where methods involving average value can perform efficiently the policy gradient is expressed in terms of the expectation.

$$\nabla_{\theta} J(\theta) = E_{\pi}[\nabla_{\theta} (log \pi(\tau \mid \theta)) R(\tau)] \quad (12)$$

The rate of parameter Θ update is expressed below:

$$\Delta\theta = \alpha * \nabla_{\theta} (log \pi(s,a,\theta)) R(\tau) \quad (13)$$

Consolidation of Q-learning and policy optimization resulted in the novel Actor-critic algorithm to achieve improved policy estimation. In actor-critic algorithm simultaneously two parallel networks for Actor and Critic respectively are updated at every point. The update

at every step facilitates in approximating the maximized expected reward for an action and the resulting state.

The actor can be explained as the policy structure as it is used to choose the actions of the agent dictating its behavior.

$$\pi(s, a, \theta) \tag{14}$$

The critic is the estimated value function used to evaluate the actions taken by the agent.

$$\hat{q}(s, a, w) \tag{15}$$

The training of two independent networks demands for optimization of two parameters θ and w for policy and value update respectively.

$$\Delta\theta = \alpha \nabla_\theta (\log \pi_\theta(s, a)\, \hat{q}_w(s, a)) \tag{16}$$

Here α is the learning rate of policy update and the action value is $\hat{q}_w(s, a)$.

$$\Delta w = \beta(R(s, a) + \gamma \hat{q}_w(s_{t+1}, a_{t+1}) - \hat{q}_w(s_t, a_t)) \nabla_w \hat{q}_w(s_t, a_t) \tag{17}$$

The time difference error is: $R(s, a) + \gamma \hat{q}_w(s_{t+1}, a_{t+1}) - \hat{q}_w(s_t, a_t)$.

The action value gradient is: $\nabla_w \hat{q}_w(s_t, a_t)$ with β as the learning rate.

The Trust Region Policy optimization (TRPO) assists in consistent policy change at each step implementing KL divergence constraint on parameter update in consecutive steps. In off-policy algorithm there stands a chance where policy π and the action-value learning rate β to optimize and obtain the courses of agent are distinctive. Importance sampling optimizer addresses the predominant gap in the policy score function between the distribution of the training data and the current policy state. The modified policy score function is formulated below.

$$\begin{aligned} J(\theta) &= \sum_{s \in S} \rho^{\pi_{\theta_{old}}} \sum_{a \in A} \pi_\theta(s|a)\, \hat{A}_{\theta_{old}}(s, a) \\ &= \sum_{s \in S} \rho^{\pi_{\theta_{old}}} \sum_{a \in A} \beta(s|a) \frac{\pi_\theta(s|a)}{\beta(s|a)}\, \hat{A}_{\theta_{old}}(s, a) \end{aligned} \tag{18}$$

Here, θ_{old} is the known parameter preceding the policy update. Estimated advantage function substitutes for the true value of the reward as the actual reward is unknown.

Proximal Policy Optimization (PPO) is an improvement over TRPO with respect to the complexity in implementation however confining to the alike implementation procedure. PPO substitutes the current policy with ratio of the current is to old policy in TRPO. The ratio is defined as:

$$r(\theta) = \frac{\pi_\theta(s|a)}{\pi_{\theta_{old}}(s|a)} \tag{3}$$

Additionally, the process bounds the ratio within a small closed interval $[1 - \varepsilon, 1 + \varepsilon]$ imposing a limit on the divergence on the two consecutive policies. The algorithm defines the score function as follows:

$$J^{CLIP}(\theta) = E[\min r(\theta) \hat{A}_{\theta_{old}}(s, a),\ clip(r(\theta), 1 - \varepsilon, 1 + \varepsilon) \hat{A}_{\theta_{old}}(s, a)] \tag{20}$$

4 Design Specification

Actor-critic algorithm involves two parallel neural networks that addresses successfully the high variance involved in the gradient approximation compromising on the inherited bias. PPO at core a policy gradient algorithm is an improvement over the TRPO with respect to minimised implementation complexity. The methodology proposes closed boundary ratio between the consecutive old and current policy distribution aiming to reduce the between variance. The paper has combined the above-mentioned algorithms to implement on the experiment involving continuous action space.

The policy update as expressed in equation (16) in actor-critic algorithm uses the logarithmic transformation of the current policy. The implemented algorithm substitutes the current

policy in the logarithmic function of the actor-critic with the clipped policy ratio expressed by PPO in the equation (19) in the logarithm function. The modified policy update function facilitates in consistent policy update with minimised variance between the consecutive policies. This results in significant small and optimised gradient ascent at each iteration.

Mathematically the modified policy gradient can be expressed as:

$$\Delta\theta = \alpha\nabla_\theta(\log\frac{\pi_\theta(s|a)}{\pi_{\theta_{old}}(s|a)}\hat{q}_w(s,a)) \quad (21)$$

4.1 Algorithm

Step 1: *Initialize parameters θ,w and state s randomly; draw sample action a following probability distribution $\pi_\theta(a|s)$*

Step 2: *For t in $\tau = (s_0, a_0, s_1, a_1, s_2, a_2 \ldots.)$:*

Step 3: *Draw sample reward r_t following state s' and following action a'*

Step 4: *Compute the ratio:* $\frac{\pi_\theta(s|a)}{\pi_{\theta_{old}}(s|a)}$

Step 5: *Compute:* $\Delta\theta = \alpha\nabla_\theta(\log\frac{\pi_\theta(s|a)}{\pi_{\theta_{old}}(s|a)}\hat{q}_w(s,a))$

$$\Delta w = \beta(R(s,a) + \gamma\hat{q}_w(s_{t+1}, a_{t+1}) - \hat{q}_w(s_t, a_t))\nabla_w\hat{q}_w(s_t, a_t)$$

Then update the parameters of action and critic: $\theta \leftarrow \theta + \alpha\Delta\theta$,$w \leftarrow w + \beta\Delta w$

Step 6: *Update $a \leftarrow a'$ and $s \leftarrow s'$*

Note: α , β is the learning rate.

5 Implementation

The formulated hybrid algorithm of actor-critic and PPO is aimed to implement on the virtual self-driving car. Here, the challenge lies in gauging the performance of the formulated algorithm as the self-driving car is an independent environment. To assess the credibility of the algorithm it is implemented on Gym environments like Cartpole, MountainCar and MountainCarContinuous. Policy gradient algorithms have proven to be successful for instances where the state belongs to only real-valued continuous space and the action space can be discrete or continuous. Thus, to full proof the efficiency of the algorithm the environment selected as the testbed belongs to discrete (Cartpole and MountainCar) and continuous (MountainCarContinuous) action space, respectively.

5.1 Environment: Open AI Gym

A research company with no profit and is focused on developing AI that would benefir everyone is OpenAI. The founders of OpenAI(Brockman et al., 2016) are Sam Altman and Elon Mask. According to their website the mission of the OpenAI is to bulild sage AGI and assure the benefits are available for everyone.

A toolkit for comparing and building reinforcement algorithms are known as OpenAI Gym. It guides through everything including from walking to play different games to teaching agents. For learning reinforcement tasks gym performs as an open source interface. The developer can decide on any reinforcement algorithms in the environment which Gym provides. Using numerical computation library which exits like Theano or Tensorflow

developers can write agent. It provides around 700 opensource environment while writing. An environment is known as the universe of agent where state of agent changes with different actions performed. A system which distinguish environment through sensors and complete actions with actuators are called Agents.

5.1.1 Cartpole

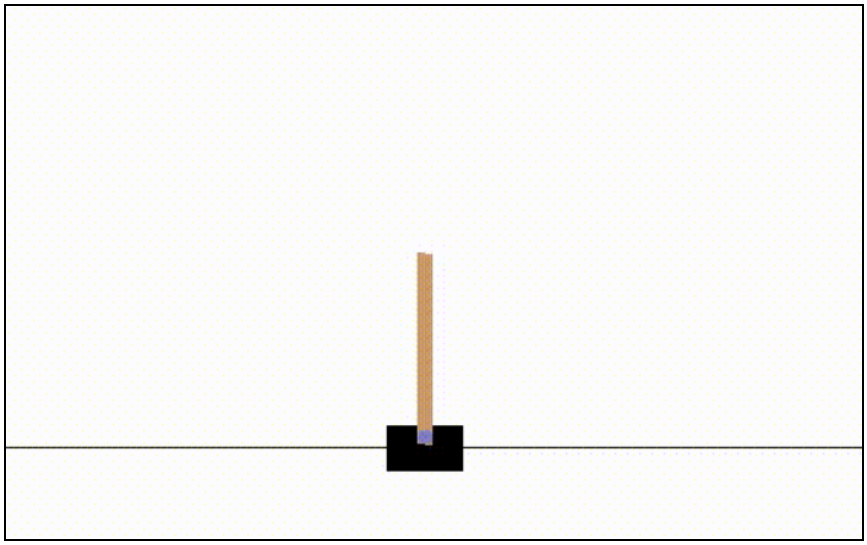


Figure 3: Balancing the pole on the cart.

In gym, cartpole is one of the famous games available. In this game, a cart which is associated with a pole must be in balanced state, so that this will not fall. If the pole bend more than 15 degrees or it moves more than 2.4 units apart from the centre the game will end.

Specifications
Name: Cartpole-v0[2]
Category: Classic Control
Description (Cartpole-v0): In CartPole-v0, a pole has been connected by a joint to the cart which is unattached, and it is moving across a frictionless track. A force of +1 or -1 on to the cart controls the system. When the pendulum goes vertical, so the aim is to prevent it from falling. Each time the ole is standing upright a reward of +1 is given. If the pole bend more than 15 degrees or it moves more than 2.4 units apart from the centre the game will end.
Source: The environment compares to the cart-pole problem according to Sutton, Barto, Anderson.
Observation

Number	Observation	Maximum	Minimum
0	Cart Position	2.4	-2.4
1	Cart Velocity	Inf	-inf
2	Pole Angle	41.8	-41.8
3	Pole Velocity at Tip	Inf	-inf

[2] https://gym.openai.com/envs/CartPole-v1/

Actions

If the number is 0, then push the cart to left side and if its 1 push the cart to the right side.

Reward

1 is the reward for every step which includes the termination step, and the threshold is 475 for v1.

5.1.2 MontainCarContinous- v0:

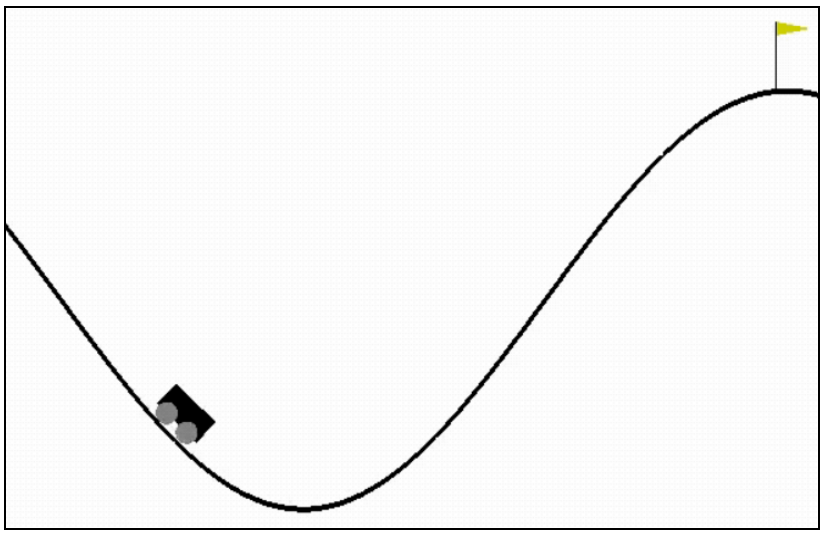


Figure 4: Car trying to reach the mountain top.

MountainCarContinuous addresses the similar reinforcement learning problem as of the MountainCar with a more generalised and realistic real-valued continuous action space.

Specifications

Name: MountainCarContinous-v0[3]

Category: Classic Control.

Description: A car which has less power must climb a hill in one-dimension to reach the top of the hill. In MountainCarContinous the engine force applied to the car is continuous not discrete. The goal is to reach the top of the mountain on the right-hand side of the car. This episode will be terminated if the car has reached or if it goes beyond that. There is another hill on the left-hand side. The car can earn potential energy and can move further towards the target if it climb this mountain. When the car reaches the top of the hill, the car cannot mover further or can move to a position which is equal to -1. Penalty will not be applied after reaching this position.

Source: The continuous version of mountain car is developed by Andrew Moore during PhD.

Observation

Number	Investigation	Maximum	Minimum
0	Car Position	0.6	-1.2
1	Car Velocity	0.07	-0.07

Here the velocity is to aid exploration.

Actions

[3] https://gym.openai.com/envs/MountainCarContinuous-v0/

The action will be continuous as the number is 0 and the car can be push towards left side or right side, which is a negative or positive value, respectively.

Reward

For reaching the target on the top of hill the reward is 100 after subtracting the squared sum of actions from initial point to goal. An exploration challenge is raised by the reward function when the target is not reached soon as it will decide to not to move further and target will not be able to find.

6 Evaluation

The reinforcement learning algorithms primary aims at enhancing the action taken by the agent based on the rewards obtained through interaction with the environment. The efficiency of an algorithm is assessed depending on the deployment of the agent through either by quality of the policy or the obtained reward. For a comparatively less model training time it is a challenge for the agent to explore all possible options to achieve the best policy. In such instances agent targets at maximising the reward values while in the learning phase. Thus, formulating the appropriate the reward function facilitates in accelerated learning rate. Rewards can be positive or negative encouraging to accumulate the rewards or attain the terminal state at an accelerated speed to eliminate addition of penalties, respectively.

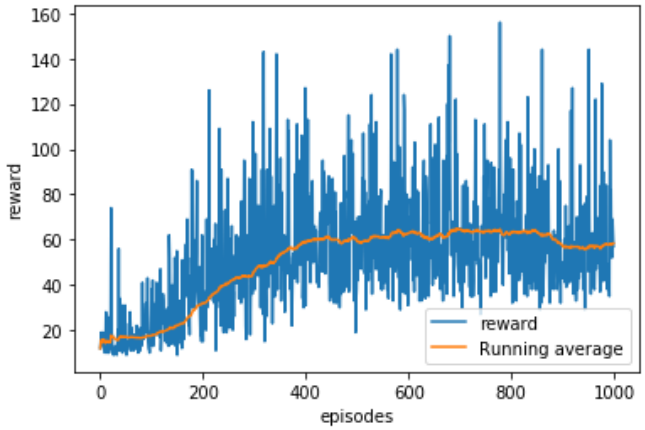
Evaluation of the proposed hybrid algorithm has been performed empirically using the Cartpole, MountainCar (discrete and continuous) as the benchmark environment prior to its implementation on the virtual self-driving car. For every environment multiple episode each consisting maximum of 1000 steps are execute based on the stability obtained in the achieved reward. However, it is expected to observe fluctuations in the reward curve as the agent learns the environment through the “trial an error” procedure. Thus, moving average 2 is performed on the total rewards of the consecutive episodes to denoise the reward curve and facilitate in clear understanding of the trend and expected total reward.

6.1 Cartpol:

The pole and the cart are attached through a moveable joint which facilitates the pole to shift along a friction-free track. The aim is to prevent the pendulum from falling by controlling the velocity.

The agent is trained for 1000 episodes in the Cartpole environment. The graph plotted with the y-axis as the reward value and the x-axis as the episodes illustrates the behavior of the reward values. The adjacent table demonstrates the statistical computations for every 100 episodes to support the requirement in solving the Cartpole environment in Gym. The environment is conditioned to be solved if the agent scores on an average equal to or over 195 for 100 consecutive trials.

6.1.1 Actor-Critic

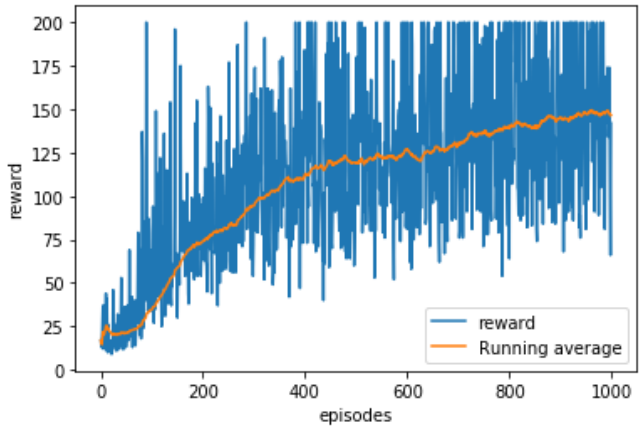


episode	total reward (per episode)	avg reward (last 100)	SD (last 100)
0	12.0	12.0	0.0
100	38.0	17.7	9.9
200	15.0	31.9	17.1
300	47.0	46.0	21.7
400	31.0	58.8	25.7
500	57.0	59.1	20.5
600	70.0	63.0	24.1
700	42.0	64.5	24.4
800	74.0	63.3	20.6
900	44.0	57.2	21.2
Avg reward for last 100 episodes			58.49
SD			25.84
Mean			52.04
CV			0.50

Figure 5: Actor-critic in Cartpole environment.

The initial episodes resulted in low reward values. Training the agent for number of episodes resulted in significant high reward in the later phase. The presence of fluctuations in the absolute reward values hinder the understanding of the trend and expected cumulative reward. The moving average curve illustrates the reward value trend. The gradual increasing trend in the slope of the reward curve between 200-400 episodes followed by the sudden flattening of the curve until 900th episode reflect the accelerated learning rate of the agent. This is to note that the agent fails to satisfy the predefined threshold average reward values for consecutive 100 episodes. The agent was able to achieve approximately only 60 as average reward value even in the state of stability failing to solve the cartpole environment in Gym. Additionally, the total variance of the reward is in line with the variance of the 400th and most of the later episode where the curve attains its stability hinting at stabilization of the exploration rate. The early stabilization of the exploration implies that further training of the agent will not result in improved reward values.

6.1.2 Modified Actor-Critic



episode	total reward (per episode)	avg reward (last 100)	SD (last 100)
0	17.0	17.0	0.0
100	69.0	34.9	28.8
200	99.0	74.3	31.1
300	113.0	94.4	34.1
400	149.0	112.7	37.4
500	147.0	118.9	42.7
600	81.0	126.3	38.0
700	179.0	130.3	41.5
800	64.0	138.9	41.9
900	169.0	144.7	42.0
Avg reward for last 100 episodes			146.68
SD			50.73
Mean			112.10
CV			0.45

Figure 6: Modified Actor-critic in Cartpole environment.

In the early training phase starting with low reward values few spikes in reward values (approx. 200) are noticed with significant fluctuations. However, for the implemented modified actor-critic algorithm the agent achieved a total reward value of approximately 200

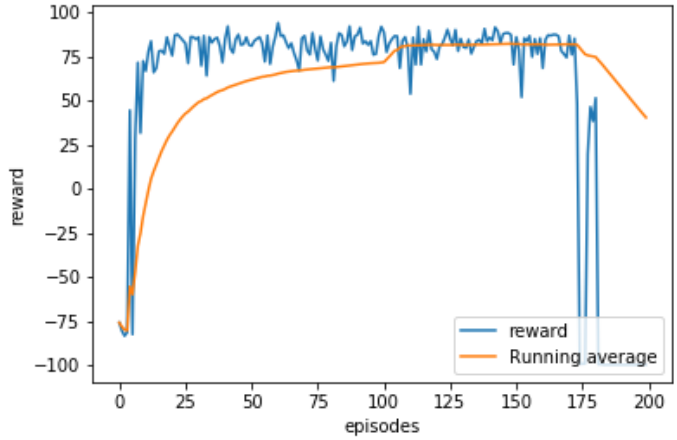
in most of the episodes staring from 400th episode with reduced fluctuations. Thus, it is probable to have an average reward value in the proximity of 195 for consecutive 100 episodes satisfying the requirement to solve the environment. A gradual increasing trend is observed in the moving average curve reflecting a comparatively lowered learning rate of the agent. The difference in total variance of the rewards and the variance of each episode explains that there is a room for the agent to still maximize the reward to attain the optimum policy.

6.2 MountainCarContinuous

Here, the agent is a low-powered car which is aimed at climbing a one-dimensional mountain to attain the target. The actions taken into account are mapped to real-valued continuous space unlike MountainCar-v0.

The agent is trained for 200 episodes in the MountainCarContinuous environment. To visually evaluate the behaviour of the rewards, gain over the episodes graph is plotted with y-axis representing the reward against the episode as x-axis. The adjacent table of statistical computations over the rewards values for every 20 episode provides a checkpoint to ensure if the algorithms satisfy the requirements for the MountainCarContinuous environment in Gym. A benchmark reward value of 90 is assigned to solve the environment.

6.2.1 Actor-Critic



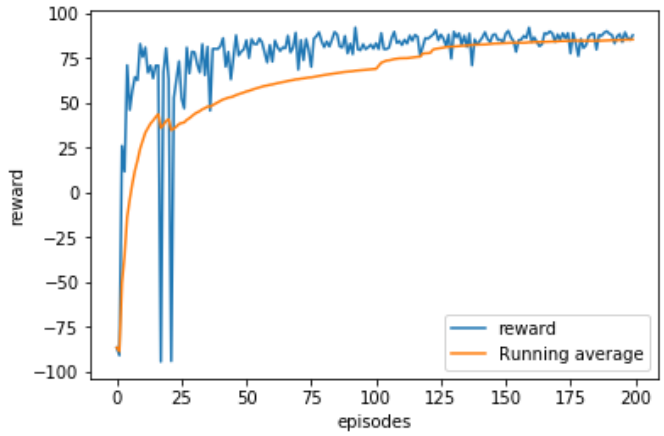
episode	total reward (per episode)	avg reward (last 100)	SD (last 100)
0	-76.0	0.0	0.0
20	75.4	30.2	65.9
40	85.0	55.8	53.4
60	93.9	64.7	45.4
80	82.6	68.6	40.1
100	88.6	71.5	36.4
120	73.4	81.5	7.0
140	85.6	81.8	6.6
160	68.3	81.7	7.3
180	51.3	74.8	32.7
199	17.5	39.8	22.6
SD			61.0
Mean			55.6

Figure 7: Actor-critic in MountainCar environment.

The reward attains a remarkable high positive value and stabilizes at a very initial training phase (around the 5th episode). Moreover, it is important to notice that the fluctuations in the flattened region of the reward values are not diversified that is a high reward value is maintained until a significant number of episodes (around 175th episode). However, in the later phase a noticeable dip observed accompanied by significant fluctuations between positive and negative rewards. in the values. This is to mention that after the steep fall in the curve it fails to attain stability till the training phase terminates. To understand the trend in the reward values moving average is plotted. The graph illustrates a stationary curve with a steep increase in the initial episodes. The steep increase in the curve signifies very high learning rate. The negative shift in the curve post the 175th episodes falls in line with the total reward values of the respective phase. The reward values lying in the close proximity of the threshold value of 90 to solve the environment. This apparently shows that the environment is solved by the agent otherwise challenged by the steep fall which fails to recover. In addition to this, the significant flattened curve between 105-175 episode approximately supported by

dramatic low and stable standard deviation values in the respective episodes validates stable exploration rate is probable to restrain the agent in recovering from a bad policy.

6.2.2 Modified Actor-Critic



episode	total reward (per episode)	moving avg till last 100	SD till last 100
0	-86.5	-86.5	0.0
20	63.3	41.0	57.4
40	81.8	51.1	49.6
60	73.1	60.3	42.9
80	81.2	65.6	38.4
100	83.4	69.0	35.1
110	82.3	74.9	25.7
120	85.8	77.8	19.3
140	84.6	82.5	5.3
160	84.2	83.9	4.6
180	80.7	84.7	4.0
199	87.2	85.4	4.2
SD			26.3
Mean			77.2

Figure 7: Modified Actor-critic in MountainCar environment.

The initial phase of the training is prevalent with spikes in the reward values followed by gentle increase in the values till the 20th episode before it fairly stabilizes with a slight hint of increasing rate at the terminal episodes. The gradual increment in the slope of the curve signifies lower learning rate. Moreover, there is no reverse trend observed in the curve. The rewards values centering around 85 for significant number of episodes with a gentle increasing trend facilitates in inferring that the applied modified algorithm supports the agent in successfully solving the MountainCarContinuous environment. The imbalance in the total variance and the per episode variance illustrates that the agent holds exploration chance to maximize reward to attain improved optimum policy.

6.3 Discussion

6.3.1 Cartpole

The accelerated learning rate due to implantation of conventional actor-critic algorithm (Thomas, 2014) triggers a dramatic policy update rate resulting in aggressive moves to attain high rewards quickly hindering to map the policy and parameter space. On the contrary, slow learning rate in the modified actor-critic algorithm restrains from inconsistent policy update addressing the challenge in conventional actor-critic algorithm. This is to mention that in conventional method of policy update there is a chance that due to unconstrained learning rate might result in bad policy from where it is difficult for the agent to recover from the resulting state which is significant from the negative shift in the moving average curve even after attaining stability. The modified algorithm juxtaposed with the actor-critic have lower policy update rate constraining the agent from taking aggressive moves aiming at maximization of expected reward. Here, positive rewards are obtained thus coefficient of variation (CV) is used to compare the conventional and modified method. The relative lower value CV for the modified algorithm validates that the variation in the consecutive policy updates is reduced.

6.3.2 MountainCarContinuous

The parity between the conventional and modified actor-critic algorithms is that both are close to solving the environment having the reward values in a close proximity of the benchmark value. However, for the conventional method there is a negative shift post the stability phase in reward signifying the attainment of bad policy posing a challenge to reverse the situation. This is falls in line with the downside of the traditional policy gradient algorithms. In other words, the accelerated unconstrained learning rate leads to aggressive moves of the agent to achieve high rewards and sudden stability in the reward values at a very initial training phase. This stepwise update of parameters leads to inconsistent policy update dictating the agent to obtain a bad policy. The modified algorithm lowers the learning rate by imposing the constraint on the parameter update. This resulted in gradual and consistent policy update aiming to maximize the reward. Moreover, the actions taken by the agent is controlled to prevent any aggressive moves leading to irreversible bad policy update. However, the major downside of the modified actor-critic is the reduced learning rate compromises on the time taken to attain stability followed by optimum policy.

There is a distinct trade-off between the conventional and the modified actor-critic algorithm. The conventional method has an accelerated learning rate which facilitates in achieving high reward value allowing a risk factor. On the contrary, the modified algorithm updates policy at a consistent and slow rate minimizing the agent's chance of sudden learning of bad policy from where it is difficult to recover. However, the low learning rate significantly increases the training time for the agent which might be a challenge to apply in real-world problems.

7 Conclusion and Future Work

The conventional actor-critic algorithm updates the parameters through unconstrained learning rate. This results in dramatic parameter update often triggering an exploded inconsistent policy update to achieve the maximum reward values. Thus, the exploration scope for the agent is minimized dictating the agent learn an irreversible bad policy in some instances. This downside of the conventional algorithm is addressed by the modified actor-critic algorithm. The algorithm bounds the learning rate to prevent rapid and inconsistent policy update. The consistent update refrains the agent from learning a bad policy. The enhanced learning policy of the agent through implementation of the modified algorithm is consistently proved in the Cartpole and MountainCarContinuous environment. However, the low learning rate increases the time taken to achieve the high reward or the optimum policy. This proportionally increases the training time of the model. Thus, to implement the algorithm on real-world problem demands for compromising on time and cost.

A self-driving car is an enduring aim of Artificial Intelligence. This demands for computers to acquire human skills, understanding and experience to drive a vehicle autonomously without impediment and damage. The goal can be achieved through object detection of the environment and learning the driving policy. The desired results of the modified algorithm encourage to implement the algorithm on virtual self-driving car. As the timeframe of the project do not permits for deployment the implementation on virtual self-driving car as the future work of the project.

Acknowledgement

I would like to take this opportunity to acknowledge my supervisor Prof Dr. Muhammad Iqbal for extending his constant guidance, support and encouragement throughout the course of dissertation.

References

Ansehel, O., Baram, N., & Shimkin, N. (2017). Averaged-DQN: Variance reduction and stabilization for Deep Reinforcement Learning. *34th International Conference on Machine Learning, ICML 2017*, *1*, 240–253.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Dhruva, T. B., Muldal, A., Heess, N., & Lillicrap, T. (2018). Distributed distributional deterministic policy gradients. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 1–16.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym*. 1–4. http://arxiv.org/abs/1606.01540

Chou, P. W., Maturana, D., & Scherer, S. (2017). Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. *34th International Conference on Machine Learning, ICML 2017*, *2*, 1386–1396.

Com, C. G. (n.d.). *Continuous Deep Q-Learning with Model-based Acceleration*.

Degris, T., White, M., & Sutton, R. S. (2012). Off-policy actor-critic. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, *1*, 457–464.

Fujimoto, S., Van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *35th International Conference on Machine Learning, ICML 2018*, *4*, 2587–2601.

Grondman, I., Busoniu, L., Lopes, G. A. D., & Babuška, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *42*(6), 1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595

Henderson, P., Chang, W.-D., Shkurti, F., Hansen, J., Meger, D., & Dudek, G. (2017). *Benchmark Environments for Multitask Learning in Continuous Domains*. 1–6. http://arxiv.org/abs/1708.04352

Hester, T., Schaul, T., Sendonaris, A., Vecerik, M., Piot, B., Osband, I., Pietquin, O., Horgan, D., Dulac-Arnold, G., Lanctot, M., Quan, J., Agapiou, J., Leibo, J. Z., & Gruslys, A. (2018). Deep q-learning from demonstrations. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 3223–3230.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., & Silver, D. (2018). Distributed prioritized experience replay. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 1–19.

Kappen, H. J. (2011). Optimal control theory and the linear Bellman equation. *Bayesian Time Series Models*, *9780521196*, 363–387. https://doi.org/10.1017/CBO9780511984679.018

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Mnih, V., Badia, A. P., Mirza, L., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *33rd International Conference on Machine Learning, ICML 2016*, *4*, 2850–2869.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*. 1–9. http://arxiv.org/abs/1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Nagabandi, A., Kahn, G., Fearing, R. S., & Levine, S. (2018). Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. *Proceedings - IEEE International Conference on Robotics and Automation*, 7579–7586. https://doi.org/10.1109/ICRA.2018.8463189

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 1–21.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. 1–12. http://arxiv.org/abs/1707.06347

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., & Schmidhuber, J. (2008). Policy gradients with parameter-based exploration for control. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5163 LNCS*(PART 1), 387–396. https://doi.org/10.1007/978-3-540-87536-9_40

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. *31st International Conference on Machine Learning, ICML 2014*, *1*, 605–619.

Thomas, P. S. (2014). *Bias in Natural Actor-Critic Algorithms*. *32*, 1–8. papers://d471b97a-e92c-44c2-8562-4efc271c8c1b/Paper/p261

Van Hasselt, H. (2010). Double Q-learning. *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, NIPS 2010*, 1–9.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-Learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2094–2100.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Frcitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. *33rd International Conference on Machine Learning, ICML 2016*, *4*(9), 2939–2947.

Xiao, Y., Hoffman, J., Xia, T., & Amato, C. (2019). *Learning Multi-Robot Decentralized Macro-Action-Based Policies via a Centralized Q-Net*. 2094–2100. http://arxiv.org/abs/1909.08776