Evaluating Model-Free Algorithms for Learning World Value Functions

Abigail Naicker
Supervised by Dr Steven James and Dr Benjamin Rosman
School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg
2350973@students.wits.ac.za

***Abstract*— Machine learning has a method called, Reinforcement learning (RL), that is a framework for developing intelligent agents that are capable of making decisions in difficult environments. In RL, value functions provide an estimate of the expected cumulative future reward for an agent in a certain state. The pursuit of creating adaptable and goal-oriented agents in the field of RL has led to the exploration of World Value Functions (WVFs). World value functions learns the value of reaching an action at each state attempting to reach each and every goal and they can be evaluated using model-free algorithms. Recent work has demonstrated the importance of learning a particular type of goal-oriented value function for reinforcement learning. This learning can be done with any normal reinforcement learning algorithm, but no analysis and research has been done to determine which algorithm is better to use from all the model-free algorithms. We focus in this paper, on evaluating model-free algorithms on different environments aiming to determine the most suitable algorithm to use, evaluating the performance for learning these type of functions, WVF's, by using OpenAI Gym environments to simulate different environments and vary the reward function to evaluate how it affects the performance of the algorithms. In this paper we have found that SARSA generally performs better than Q-learning and Dyna q-learning on an environment and that soft-actor critic algorithm can handle complex environments, as well as showing DQN's fluctuating performance and futher work can be done on this. This paper mentions some advantages and disadvantages of the algorithms, which can contribute and be helpful to the development of more effective and more efficient reinforcement learning algorithms for many different applications and to add to the field of machine learning, reinforcement learning.**

***Index Terms*—Reinforcement Learning, Model-Free Algorithms, World-Value Functions**

I. Introduction

Machine learning has a sub-field part of it that is called and known as Reinforcement Learning which deals with the development of algorithms that allow an agent to learn how to decide on decisions in a setting where some idea of reward is maximized. Reinforcement learning has become popular and is being used in many different applications. One application where reinforcement learning is used is in game play as researched in Barto *et al.* [2017] paper which AlphaGo developed by DeepMind, used reinforcement learning techniques to defeat human champions in the complex game of Go. Another application is to personalize web services, batch-mode reinforcement learning was employed, and the result was a policy that encourages users to interact with a website for extended periods of time. This policy is now a part of the Adobe Marketing Cloud as evaluated in Barto *et al.* [2017]. Reinforcement learning has even stretched to be used in the healthcare field for dynamic treatment regime and medical imaging analysis and diagnosis [Coronato *et al.* 2020]. Another popular and captivating application of reinforcement learning is used in robotics [Garaffa *et al.* 2021]. It enables robots to learn complex tasks and control. Robots can learn to navigate environments and has been taught to walk [Haarnoja *et al.* 2018b], run, and perform other tasks.

The ability to estimate the value of actions in an environment is critical for decision making and optimal behaviour in reinforcement learning. In reinforcement learning, value functions provide an estimate of the expected cumulative future reward for an agent, and they can be learned using model-free algorithms and a world value function learns that value of reaching an action at each state attempting to reach each and every goal.

Recently Tasse *et al.* [2022] proposed a goal-oriented version with properties of that is can be learned using any suitable reinforcement learning and such as the Q-learning algorithm which can be seen in figure 2.1 and that it can encode knowledge of how to solve the task in which it was learned, but there has never been an analysis of how best to learn them. We have done a thorough empirical analysis on different stimulated environments by testing how all the different model-free algorithms performance is.

Any previous understanding regarding the environment's dynamics is not required for the type of algorithms we are evaluating (model-free). Usage of the right selection, and evaluation for the models and as well the right selection for the algorithms is important in many aspects of machine learning as stated in Raschka [2018] . These algorithms learn the world value function that is optimal through encountering with the environment after this it is on the basis of the noticed rewards and states, it updates the world value function. The model-free approach's greatest advantage is its fast computation. Model-free and model-based algorithms are different and the reason is that, in a model-free algorithm it often supports a larger representation than a model-based approach when the two algorithms are given the same amount of computing due to a low computational demand. The model-free approach can perform better asymptotically. There are different model-free algorithms for learning this function, namely some are State-Action-Reward-State-Action (SARSA) [Sharma *et al.* 2021],

Deep-Q Networks, Q-learning [Tasse *et al.* 2022], Soft-Actor Critic (SAC) [Haarnoja *et al.* 2018a].
However, choosing the right model-free algorithm for a specific reinforcement learning task can be challenging. Previous researchers have evaluated the performance and analysed a specific algorithm specifically, such as in Clifton and Laber [2020], the Q-learning algorithm performance was evaluated. No research has been done to compare these algorithms against one another and to determine which algorithm is better to use in terms of convergence time and accuracy. In the research Tasse *et al.* [2022], world-value functions were evaluated but only on the Q-learning algorithm specifically. What about all the other algorithms? This research aims to evaluate the performance of different model-free algorithms for learning world value functions in a simulated environment and to determine which algorithm will be better to use. In the research Sutton and Barto [2018] introduces some reinforcement learning algorithms and explores their strengths and weaknesses of these different algorithms. This research can contribute to the field of machine learning and help other researchers and users in selecting an algorithm to use for a task.
Firstly, we define and understand the difference between the value functions and the one we are evaluating, world value functions, of which we have seen that value functions aim to assess the future rewards for specific states or state-action pairs and world-value functions focus on evaluating the overall performance and effectiveness of an agent's policy across the entire environment.
Secondly, we address the question of what is the impact of different goal-reward functions on the performance of model-free algorithms in learning these world value functions. We find that by using the world-value function idea, model-free algorithms demonstrate a more comprehensive understanding of the underlying environmental dynamics and can consequently adapt to complex scenarios.
Finally, we address the question of what is the performance of the different algorithms for learning world value functions in a simulated environment. We find that the performance of these algorithms is significantly influenced by the environment's complexity, as well as the chosen exploration-exploitation strategy, highlighting the significance of customizing the algorithmic decisions to the unique features of the simulated world.

II. Background and Related Work

A. Reinforcement learning

A type of machine learning training method that trains an agent how to interact with the world around it in order generate decisions that maximize a cumulative reward signal. One of the key components of reinforcement learning is the estimation of value functions. This method of which is often modelled through a Markov Decision Process (MDP) is used solve a RL problem [Barto *et al.* 2017].
The way in which the agent and environment interact—the agent acting in it, and the environment observing and rewarding it is the focus of this reinforcement learning problem. As it is depicted in figure 1. After the agent performs an action A_t at time t, as a result, the environment provide the agent feedback regarding the state S and reward R. This is a cyclical process. So the RL problem is to discover a policy that is the optimal action, a, for each state s to maximise total rewards R. The agent's objective is to learn the optimal policy, which determines the best course of action in each state, in order to maximize its cumulative reward over time.

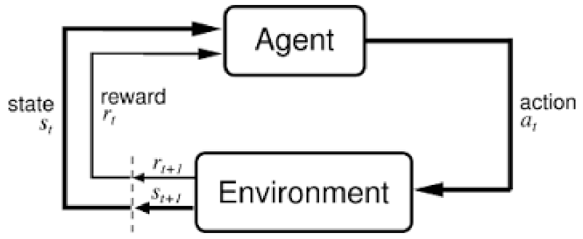


Fig. 1. Agent Environment Interaction [Barto *et al.* 2017]

B. Markov Decision Processes

This is a framework that is used in reinforcement learning that is used for modelling decision making problems, being a 5 tuple: $<$S, A, P, R, $\gamma>$ Barto *et al.* [2017], where:

- States (S):
  These finite states represent the different configurations or situations in which the agent can find itself during the decision-making process.
- Actions (A):
  The actions represent the possible choices given to the agent in a certain state.
- Transition function (P):
  This is the function of how the world evolves under actions. When an action is performed by the agent in a particular state, it transitions to a new state based on transition probabilities, as seen in equation 1 .
- Rewards (R):
  This is the feedback signal to the agent. The goal of the agent is typically to maximize over time the cumulative total rewards as seen in equation 2 .
- Discounted Factor (γ):
  This real-valued discount factor indicates our current valuation of rewards in regards to future rewards.

$$P(s'|s,a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a] \tag{1}$$

$$r(s'|s,a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \tag{2}$$

The reward is employed to assess the value of the state, aiding in the determination of the matching value function for every state, where return is the entire cumulative reward as expressed in equation 3. Now we would like dense rewards as it allows the agent to learn faster as you getting feedback more.

$$G(t) = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3}$$

The dynamics of a MDP can be modelled as a Markov process, which presumes the Markov property. According to the Markov property, which states that a system's behavior in the future only depends on its existing state as seen in equation 4. This property allows the agent to decide on decisions based solely on the current state, disregarding the prior actions and states.

$$\mathbb{P}[S_{t+1}|S_1, A_1, S_2, A_2, ..., St, At] = \mathbb{P}[S_{t+1}|St, A_t] \tag{4}$$

The goal of MDP is to identify the best course of action that will maximize the projected cumulative reward as time goes on. [Sutton and Barto 2018]. This can be done using various algorithms such as value or policy iteration, or reinforcement learning methods like the Asynchronous Advantage Actor-Critic (A3C) algorithm or the Q-learning.

C. *Value Function and Policies*

These functions provide an estimate of the expected cumulative future reward for an agent in a certain state. Value functions are used to direct the agent's decision-making process by estimating the long-term consequences of taking different actions. The state-transition probability and the action policy's probabilistic character both influence the value function. The Bellman equation takes these probabilities into consideration (Equation 5). Anyone who studies machine learning and specifically reinforcement learning, will know that the Bellman Equation is an important part of reinforcement learning. The Bell man equation is a recursive function that informs us the future rewards we can anticipate knowing the current situation.

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_\pi(s')] \tag{5}$$

A policy maps states to the chances of choosing each potential course of action.The function π(a|s), the policy, determines the likelihood that an agent in state s will select action a, and this is known as the policy [Barto *et al.* 2017]. The agent wants to maximize the total amount of rewards it gets in the long run [Sutton and Barto 2018]. As you can see the value function and policy is connected terms to each other. Value functions can be estimated using various methods like temporal difference learning or, Monte Carlo methods [Sutton and Barto 2018].The combination of value functions and policies forms the foundation of reinforcement learning algorithms. By iteratively improving value functions and updating policies, an agent can learn to make better decisions and maximize the cumulative rewards in a MDP.

Value functions and policies work together. The value function considers the rewards that will be received in the future from the current situation, as per a set of rules (policy). We get an an action value function (equation 7) and a state-value function (equation 6) as expressed below:

$$v_\pi(s) = \mathbb{E}\{G_t|S_t = s\} \tag{6}$$

$$q_\pi(s, a)_\infty = \mathbb{E}\{G_t|S_t = s, A_t = a\} \tag{7}$$

Deep networks of machine learning algorithms, specifically supervised and unsupervised are used in deep learning in an effort to model high-level abstractions in data and learn from various levels of abstraction [Mousavi *et al.* 2018]. Recurrent neural networks (RNNs) and Convolutional neural networks (CNNs) recently had the most use in combination with reinforcement learning [Mousavi *et al.* 2018].

D. *World-Value functions*

World-value functions extend the concept of value functions in reinforcement learning by incorporating a broader notion of value, including not just the state-action pairs but the entire environment. They provably encode how to reach all achievable goals [Nguyen *et al.* 2018]. WVFs capture the overall desirability of different configurations of the environment, reflecting the significance of these configurations in achieving the agent's long-term goals. In contrast to traditional value functions that predominantly focus on state-action pairs, WVFs consider the global state of the environment, taking into account the agent's objectives and the underlying dynamics of the world. This expansion allows for a more comprehensive understanding of the environment's structure and the implications of different states on the agent's overall performance. By evaluating the desirability of the entire environment rather than isolated state-action pairs.
Deep Q-learning for World Value Functions (DQL-WVF) is a development from the traditional Deep Q-learning framework that focuses on learning world-value functions. By incorporating the concept of WVFs, DQL-WVF aims to capture the global desirability of the entire environment while considering long-term goals and the broader implications of various configurations. This approach utilizes deep neural networks to approximate the WVFs, enabling the representation of complex environmental structures and dynamics.

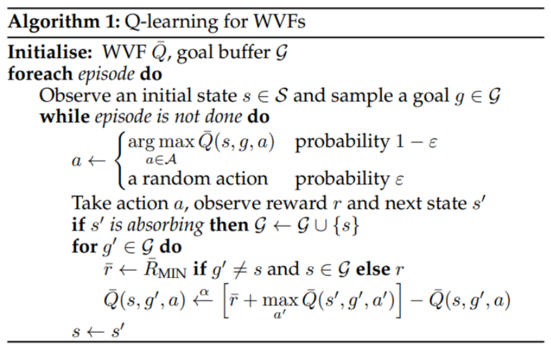
E. *Model-Free Algorithms*

These are part of reinforcement learning. These algorithms refrain from using the transition probability distribution and reward function related with the MDP. Model-free algorithms are a popular approach for learning world value functions because they do not call for past knowledge of the surrounding's dynamics compared to model-based algorithms which does. They estimate the value function related to experience of the agent in the environment. Some examples of model-free algorithms are Q-learning, Soft Actor Critic, Deep Q-Networks (DQN) and State Action Reward State Action (SARSA), which these algorithms can be used for learning world value functions.

1) State-Action-Reward-State-Action (SARSA): This is an on-policy algorithm where it learns and updates the policy while interacting with the environment [Qiang and Zhongli 2011]. On-policy methods learns by following the current policy. The SARSA algorithm learns the Q values for state-action pairs and can converge to the optimal Q-values and policy in certain cases.

There are different variations of the SARSA algorithm, one known as Expected SARSA algorithm [van Seijen *et al.* 2009] of which have improvements to the main algorithm.

2) Q-learning: This algorithm is off-policy; rather than adhering to the policy, it learns by monitoring and modifying the Q values. Off-policy methods learn from a different behaviour policy. We as well used the dyna-q learning algorithm which was invented by [Barto *et al.* 2017] to speed up convergence.



```
Algorithm 1: Q-learning for WVFs
Initialise: WVF $\bar{Q}$, goal buffer $\mathcal{G}$
foreach episode do
    Observe an initial state $s \in \mathcal{S}$ and sample a goal $g \in \mathcal{G}$
    while episode is not done do
        $a \leftarrow \begin{cases} \arg\max_{a \in \mathcal{A}} \bar{Q}(s, g, a) & \text{probability } 1 - \varepsilon \\ \text{a random action} & \text{probability } \varepsilon \end{cases}$
        Take action $a$, observe reward $r$ and next state $s'$
        if $s'$ is absorbing then $\mathcal{G} \leftarrow \mathcal{G} \cup \{s\}$
        for $g' \in \mathcal{G}$ do
            $\bar{r} \leftarrow \bar{R}_{\text{MIN}}$ if $g' \neq s$ and $s \in \mathcal{G}$ else $r$
            $\bar{Q}(s, g', a) \overset{\alpha}{\leftarrow} \left[\bar{r} + \max_{a'} \bar{Q}(s', g', a')\right] - \bar{Q}(s, g', a)$
        $s \leftarrow s'$
```

Fig. 2. Q-learning algorithm for WVF's extracted from Tasse *et al.* [2022]

Q-learning does not need any past knowledge of the dynamics of the environment as it can learn directly from experience. It has been applied in different domains, including statistics and artificial intelligence [Clifton and Laber 2020].

3) Deep Q-Networks (DQN): This algorithm is an off-policy method for handling ambient spaces that combines Q-learning and deep neural networks. Deep Q-Learning [Mnih *et al.* 2015] is a variation of the classic Q-Learning algorithm with some contributions [Roderick *et al.* 2017]:

- A deep CNN structure for Q values estimation
- Using network parameters from before to estimate the Q values of the state after.
- Using mini-batches of random training data

4) Soft Actor Critic (SAC): This algorithm uses a small version from the Q-learning algorithm which the actor maximises that return and entropy at the same time of this algorithm follows an off-policy approach [Haarnoja *et al.* 2018a]. This algorithm is an excellent algorithm to be used in robotic tasks [Haarnoja *et al.* 2018a] as it can handle challenging domains.

5) Related Work: Recently, model-free reinforcement learning algorithms have been effective, particularly when applied to the context of Atari 2600 games [Bellemare *et al.* 2012]. A model-free algorithm with a non-linear function approximator was used in Mnih *et al.* [2015] to play the games with a representation that only accepted raw pixels as input. This research demonstrated that on a small set of games, the model-free algorithm can outperform a human player. The agent had access to a wealth of experience in the Atari domain.
Universal Value Function Approximator is another type of framework in reinforcement learning that aim to learn a wide range of value functions across different tasks and domains. This framework is designed to generalize across tasks, allowing agents to apply what is learned in one work to another. These function approximators are able to generalize a previously unseen goal [Schaul *et al.* 2015]. They continue to be an active area of research in reinforcement learning.

III. Problem Representation

RL has existed since the 1960s Mahoney [2021] and some main recent difficulties in RL are to determine the most effective algorithm for learning these world value functions introduced above. Model-free algorithms learn world value functions without the prior knowledge of the environment's dynamics. However, choosing the right algorithm for a specific RL task can be challenging. There are many model-free algorithms, but every one of these algorithms have its strengths and its weaknesses, as we examine these algorithms. By examining these algorithms, we can learn a great deal about these algorithms' ability to adapt and function in a variety of settings by looking at how they change to fit different goal-reward configurations. Conducting an analysis of the impact of various goal-reward combinations on the learning process might yield a thorough comprehension of the agents' capacities to accomplish predetermined goals and effectively navigate intricate surroundings. We might speculate that the choice of different goal-reward combinations affects model-free algorithms' learning rates and overall performance in complicated situations by substantially affecting their effectiveness and flexibility. Through a deeper understanding of the complexities of goal-oriented learning, this research aids in the creation of more adaptable and successful learning systems for the agent.
My purpose is to comprehensively explore and understand the implications of different goal-reward functions on the performance of model-free algorithms in learning WVFs, with the aim of enhancing the design and implementation of more effective and adaptable learning strategies in the field of RL.

A. Algorithms Evaluating

- State-Action-Reward-State-Action (SARSA)
- Deep-Q Network (DQN)
- Q-Learning
- Soft Actor Critic (SAC)

B. Questions

- What is the performance of the different model-free algorithms mentioned above for learning world value functions in a simulated environment?

- How is the selection of hyperparameters impact the performance of model-free algorithms in learning world value functions?
- What is the impact of different goal-reward functions on the performance of model-free algorithms in learning world value functions?

IV. Methodology

A. Aims

The aims of this paper are to: (1) analyze the impact of distinct goal-reward configurations on the convergence and performance of model-free algorithms in learning World Value Functions (WVFs); (2) assess the adaptability and robustness of model-free algorithms when confronted with varying goal-reward scenarios in different simulated environments; (3) mainly evaluate the performance of several different model-free algorithms for learning world value function in a simulated environment in determining which is better to use.

B. Domain

The procedure for testing the hypothesis entails setting up the training environment and placing the agents in action. In this paper we have used the simulated environment on OpenAI Gym[1], using the Markov Decision Process as the framework to characterize the method by which the environment and the agent interact. The stimulated environment used represents a grid-world environment (Cliffwalking, Maze) of we control how large the action space, state space and rewards are structured is.

1) Cliffwalking: This grid-world is a 4x12 matrix as depicted in figure 3 , with actions of moving up, down, left and right. Within this grid is a cliff of which if the agent steps on the cliff, the agent falls and dies and restarts. We use this cliff-walking environment to evaluate the algorithms Q-learning and SARSA with the value function approach and with the world-value function approach to compare. Using this environment allows these agents of these algorithms to handle the exploration-exploitation[2] trade-off. The agents balance the risk of falling down the cliff and the pursuit of short-term rewards.

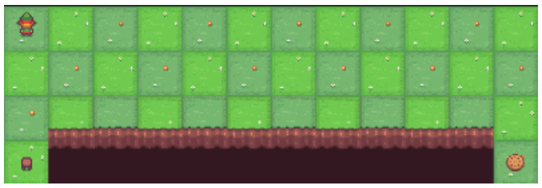


Fig. 3. Cliff-walking environment from Bowyer [2023]

[1]Toolkit that provides the simulated environment for testing the agents for reinforcement learning models

[2]The probability the agent exploits actions is 1-ϵ and the probability the agent explores actions is ϵ

2) Maze: This environment is a simple maze, as the agent needs to find its way through the maze to the goal. The actions are moving up, down, left and right but if the way is blocked with a wall, it will remain in the same position and need to decide on another action. We use this environment to evaluate the SARSA and Q-learning algorithms.

3) Pendulum: A pendulum with one end fixedly attached to a point and the other end free is the subject of this problem. The objective is to exert torque on the free end of the pendulum, which is initially positioned randomly, in order to swing it upright and place its center of gravity directly above the fixed point. We use this environment to evaluate the Soft-Actor Critic (SAC) algorithm. Using this environment for this algorithm is better because this environment has a continuous action spaces and continuous state spaces which corresponds the algorithm as the SAC can handle these types of action spaces and state spaces.

4) CartPole: In this environment a pole is attached to a cart of which the cart is moving along a friction-less track. On the cart, the pendulum is upright, and the objective of the agent is to apply force on the left and right of the cart to balance the pole. For this environment, we use the Deep-Q Network method.

C. Metrics for Evaluation

For evaluation part of the process, the cumulative rewards gained over the number of episodes were used as the main metric of evaluation for the algorithms. Using this metric let me analyze the algorithms agents ability to cumulate reward over a certain number of episodes and compare the agent that the cumulated the most rewards over the number of episodes and achieved the certain goal of the game. The second evaluation metric used was the mean episode length as this allowed me to compare the agent that reached the goal state the fastest and the algorithm that converged the fastest. Using these metrics gave a general idea and understanding of the algorithm's performance in learning world-value functions.

V. Experiments and Results

A. Baseline

We replicate the work of Vita [2020] and Tasse *et al.* [2022] to verify that our implementation is correct. See Appendix for more information.

B. Q-learning and SARSA using Value Function

1) Purpose: Using the Cliffwalking domain, we implement a Q-learning and SARSA using the normal value function as we wanted to compare the difference between the implementation of a value function and the recently new idea of the world-value function, of which the e-greedy policy is used for the exploration-exploitation trade-off.

2) Results: As we can see from the plot in figure 4 that the SARSA algorithm performs better of the two, as the average rewards over the episodes is higher for the SARSA algorithm than it is for the Q-learning and as well the Q-learning method fluctuates a lot from high to low over episodes and is not stable. Tuning the hyperparameter, epsilon, by decreasing it to 0.01 from 0.1, Q-learning changes to perform quite well, better than SARSA.

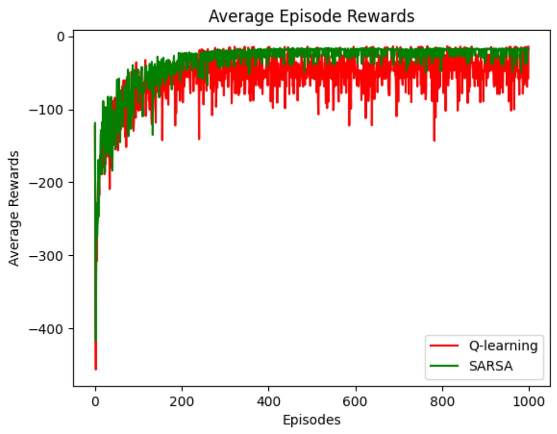


Fig. 4. SARSA & Q-learning Value Function Comparison epsilon=0.1

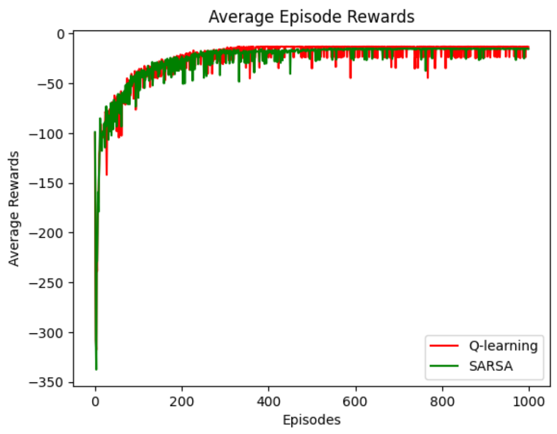


Fig. 5. SARSA & Q-learning Value Function Comparison epsilon=0.01

C. *Q-learning and SARSA using World Value Function*

1) Purpose: Even though SARSA is a on-policy method and Q-learning is a off-policy, we use them on the same domain as they both handle discrete[3] state spaces and continuous[4] action spaces. This is a reason why these two algorithms were chosen to perform on the same domain, Cliffwalking. We implemented the algorithms on this domain using the world-value function approach. The agents q-values are updated using the WVF approach. We use a 'goal value' with the reward and the discounted future rewards. This is done to prioritize the goal buffer and the use of the epsilon-greedy policy approach to ensure exploitation-exploration trade off.

[3]Finite set of possible actions/states, they are countable

[4]Infinite number of possible actions/states, they are noncountable

2) Results: The results of the cumulative rewards over the number of episodes have shown us that SARSA performs more effectively than Q-learning algorithm as it does the same when using the normal value function. As we can see under the appendix figure 7, SARSA starts of by having lower rewards than Q-learning but as the episodes goes along and the agent learns more and more, we can see that instead of Q-learning agent improving, we can it not improving where as with SARSA we can see the agent improving, and converging to an optimal policy at a faster rate in another evaluation method. SARSA generally performs better than Q-learning because of it being an on-policy method. In the cliff-walking environment, the SARSA agent will be cautious in its actions as it wont want to get near the cliff and fall off. Whereas the q-learning agent is an off-policy method and this will want to take more risks and explore more and taking riskier actions. We have well seen, as shown in figure 5 ,that dynamic q-learning using the same domain, Cliffwalking, as performed worse than SARSA and around about the same as q-learning.

D. *Soft-Actor Critic (SAC)*

1) Purpose: We use algorithms that handle discrete state space and continuous and disrete action space but now we want to see cases where the action and state spaces are both continuous. That is where the SAC algorithm comes forward and the world-value function approach in the pendulum domain, as this domain as a continuous action space and continuous state space and the SAC is an algorithm that handles the continuous state space and continuous action space. The critic network of the SAC algorithm is where the WVF idea is implemented. The critic networks in general handles the evaluation of the actions taken and the feedback. In this approach, critic part estimates the state-action value function which is the expected cumulative rewards from state s and taking action a.

2) Results: SAC is known to perform well in complex continuous action spaces. The pendulum environment being a complex continuous action space, we have seen that the SAC agent has performed not the best but well considering that it is a complex environment. In the experiment, the agent showed improvement in learning the WVF's. The agent can balance the exploitation-exploration trade-off of which this can show that this agent can be a choice that is used in different environments.

E. *Deep Q-Network (DQN)*

1) Purpose: In the above algorithms we evaluated ones which had a discrete state space and continuous and discrete action space and ones where both the action and state space were continuous. Now we decided to not just only stick to those ones but to go further and evaluate one where the state space is continuous and action space is discrete.

2) Results: Results shown in Table 1 under appendix. The DQN agent is trained using a neural network using TensorFlow. The agent take actions, stores values, and learns from past experiences using the DQN algorithm. The agent wants to learn a policy that balances the pole for a longer amount of time by maximizing the cumulative rewards over the episodes. As we can see from the results that the score, which is the number of time steps the agent managed to keep the pole balanced, fluctuated up and down, but from episode 6 it increased onwards only. Seeing that the exploring rate is decreasing over the episodes, is showing that the agent is exploring less and less over time and exploiting its learnt policy which is a good thing and it shows the optimal policy is being learnt. We can as well see that when tuning the hyperparameter, learning rate, to be larger from 0.001 to 0.1, this affects the score negatively as the score is lower in every episode, even though there is some episodes that do well.

VI. Strengths & Weaknesses

A. State-Action-Reward-State-Action (SARSA)

1) Strengths:

- Simple to implement and understand.
- On-policy learning method, allowing for the optimization of the policy being followed during learning.
-

2) Weaknesses:

- Slower learning rate and convergence rate because of it being an on-policy method
- Can sometimes have difficulty with complex environments because the algorithm is fairly simple
- Bootstrapping

B. Q-learning

1) Strengths:

- Can converge to an optimal policy
- Flexibility
- Offline training

2) Weaknesses:

- Curse of dimensionality
- Can overestimate
- When using complex environments, requires more computer resource space

C. Soft-Actor Critic (SAC)

1) Strengths:

- It can handle continuous state/action spaces, so can handle complex environments
- Robust learning

2) Weaknesses:

- When using complex environments, requires more computer resource space
- Can be complex to understand

D. Deep Q-Network

1) Strengths:

- Can be used in multiple different complex environments
- Can generalize to new states and actions
- Can be used to solve wide range of problems

2) Weaknesses:

- Computationally expensive
- When environment is stochastic, algorithm can be unstable

VII. Discussion and Future Work

We have done only four algorithms in this paper but there are many more model-free algorithms out there as well there are many more domains that come with it. Future work could be done on the other different type of model-free algorithms, as well testing the algorithms used in this paper on the many other different domains. Reinforcement learning is a topic where there will always be room for improvement and growth.

VIII. Conclusion

In this paper we have analyzed and compared four different model-free algorithms for learning the world-value functions. We have seen that SARSA has outperformed Q-learning but even though SARSA may be better than Q-learning in one environment, further research can be done to determine if this is true for all environments. By incorporating the WVF into the SARSA and Q-learning algorithms, we showed its impact on the agents' decision-making and policy optimization. World-Value Functions contributed to enhancing the agents' learning capabilities and the agent to make informed decisions in complex environments. The DQN's performance showed fluctuating results and this can further be evaluated by exploitation-exploration balance and parameter optimization analysis. There is not a definite answer as to which algorithm is better as different algorithms both have their own advantages and disadvantages and reinforcement learning is a topic that will continue to grow and evolve.

IX. Appendix

A. Baselines

We replicated two prior works to establish baselines.

1) Tasse et al. [2022]: In this work, the authors introduced the idea of the world-value function and the properties of it. They demonstrate this WVF using the Q-learning algorithm and on the Four Rooms domain.

2) Vita [2020]: Here, the author of the source has used the maze domain of which he evaluated the State-Action-Reward-State-Action (SARSA) and Q-learning algorithms with the value function. We have used this baseline as a comparison of the value function approach and idea of the world-value function approach.

B. Code link

https://github.com/abby2412/Evaluating-Model-free-Algorithms-for-learning-World-Value-Function.git

Episode	Score	Exploration Rate
1	40	0.96
2	15	0.89
3	17	0.82
4	8	0.79
5	33	0.67
6	13	0.62
7	21	0.56
8	25	0.50
9	34	0.42
10	52	0.32

TABLE I
DQN RESULTS THAT SHOW THE EPISODE NUMBER, SCORE: WHICH INDICATES THE TIME STEPS THE AGENT MANAGED TO KEEP THE POLE BALANCED AND THE EXPLORATION RATE WHICH DECREASES AS THE AGENT LEARNS FROM THE ENVIRONMENT, USING LEARNING RATE=0.001

Episode	Score	Exploration Rate
1	14	1.0
2	35	0.91
3	10	0.87
4	11	0.83
5	27	0.72
6	9	0.69
7	97	0.42
8	41	0.34
9	32	0.29
10	46	0.23

TABLE II
DQN RESULTS THAT SHOW THE EPISODE NUMBER, SCORE: WHICH INDICATES THE TIME STEPS THE AGENT MANAGED TO KEEP THE POLE BALANCED AND THE EXPLORATION RATE WHICH DECREASES AS THE AGENT LEARNS FROM THE ENVIRONMENT, USING LEARNING RATE=0.1

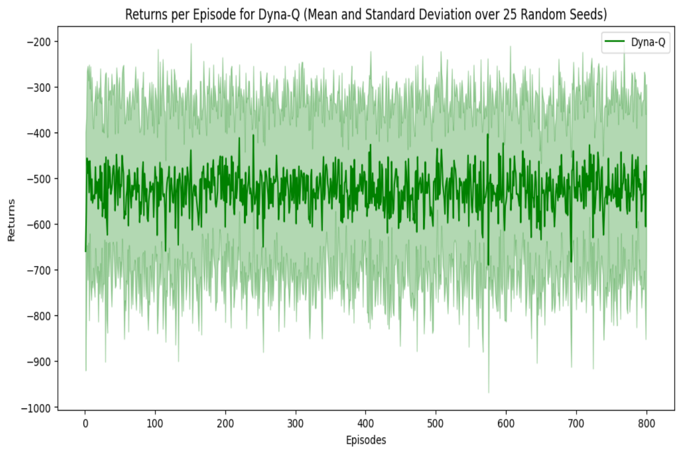


Fig. 6. Dynamic Q-learning Results

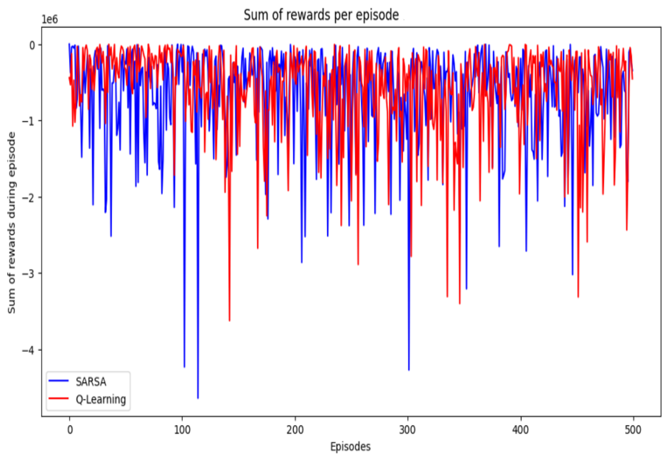


Fig. 7. Q-learning & SARSA Results

References

[Barto *et al.* 2017] Andrew G Barto, Philip S Thomas, and Richard S Sutton. Some recent applications of reinforcement learning. In *Proceedings of the Eighteenth Yale Workshop on Adaptive and Learning Systems*, 2017.

[Bellemare *et al.* 2012] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents (extended abstract). In *International Joint Conference on Artificial Intelligence*, 2012.

[Bowyer 2023] Caleb M. Bowyer. Setting up the cliff walking environment for reinforcement learning (rl). *Medium Daily*, 2023.

[Clifton and Laber 2020] J. Clifton and E. Laber. Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, 7(1):279–301, 2020.

[Coronato *et al.* 2020] Antonio Coronato, Muddasar Naeem, Giuseppe De Pietro, and Giovanni Paragliola. Reinforcement learning for intelligent healthcare applications: A survey. *Artificial Intelligence in*

Medicine, 109:101964, 2020.

[Garaffa *et al.* 2021] Luíza Caetano Garaffa, Maik Basso, Andréa Aparecida Konzen, and Edison Pignaton de Freitas. Reinforcement learning for mobile robotics exploration: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021.

[Haarnoja *et al.* 2018a] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft Actor-Critic Algorithms and Applications. Dec. 2018.

[Haarnoja *et al.* 2018b] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[Mahoney 2021] Chris Mahoney. Reinforcement learning: A review of the historic, modern, and future applications of this special form of machine learning. *Towards Data Science*, 2021.

[Mnih *et al.* 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[Mousavi *et al.* 2018] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*, pages 426–440. Springer, 2018.

[Nguyen *et al.* 2018] T. Nguyen, N. Nguyen, and S. Nahavandi. Deep Reinforcement Learning for Multi-agent Systems: A Review of Challenges, Solutions and Applications. abs/1812.11794, 2018.

[Qiang and Zhongli 2011] W. Qiang and Z. Zhongli. Reinforcement learning model, algorithms and its application. In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pages 1143–1146, 2011.

[Raschka 2018] S. Raschka. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. abs/1811.12808, 2018.

[Roderick *et al.* 2017] Melrose Roderick, James MacGlashan, and Stefanie Tellex. Implementing the deep q-network. *ArXiv*, abs/1711.07478, 2017.

[Schaul *et al.* 2015] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, Jul 2015. PMLR.

[Sharma *et al.* 2021] A. Sharma, R. Gupta, K. Lakshmanan, and A. Gupta. Transition Based Discount Factor for Model Free Algorithms in Reinforcement Learning. page 1197, July. 2021.

[Sutton and Barto 2018] R. Sutton and A. Barto. Reinforcement learning: An introduction. In *2018 MIT Press*, volume 2, Nov. 2018.

[Tasse *et al.* 2022] G. Tasse, S. James, and B. Rosman. World Value Functions: Knowledge Representation for Multitask Reinforcement Learning. May. 2022.

[van Seijen *et al.* 2009] Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184, 2009.

[Vita 2020] Luca Di Vita. Reinforcement learning maze solver. *GitHub*, 2020.