Reinforcement Learning (CAP 6629)

Report

On

Project - 3

Submitted To

Prof. Xingnan Zhong
Dept. EECS

Submitted By

Group - 1

Course Project 3 (Group Project)
Atari Game: Enduro Racing

CAP 6629: Reinforcement Learning

James Kos | Valentin Nechita | Javier Nunez | Neelima Rajawat | Mani Teja Rayana
(Group 1)

Florida Atlantic University

Actor Critic Implementation in Enduro v4 Arcade Learning Environment

Neelima Rajawat | James Kos | Valentin Nechita | Javier Nunez | Mani Teja Rayana

1.Introduction

Reinforcement Learning (RL) is an exciting and growing field of artificial intelligence research that is concerned with how a machine can learn from experience and interaction, with modern RL as we know it today coming together in the 1980s (Sutton & Barto 2018). It takes inspiration from many scenarios across the whole spectrum of academic fields, from psychology to mathematics. Further, RL is a field of study that is simultaneously a problem and a class of solution methods that work well on the problem. It is different from the more well-known supervised and unsupervised machine learning methods in the way that learning is not based on training on a set of labeled examples, nor on attempting to find structures hidden in collections of unlabeled data. Rather, RL is type of machine learning technique that enables an agent to learn in an interactive environment by trial and error, using feedback from its own actions (Sutton & Barto 2018).

To better understand RL problems, it is useful to define some key components: an agent is a component that makes the decision of what action to take, an environment is the space in which the agent operates, a state refers to the current situation of the agent, rewards are feedback from the environment, a policy is a method to map agent's state to actions, and values are future rewards that an agent would receive by taking an action in a particular state (Sutton & Barto 2018).

Within RL, Actor-Critic methods are an important class of algorithms that combine elements of value-based methods and policy-based methods. In simple terms, the actor is responsible for selecting actions, while the critic evaluates the chosen actions. Then, value-based methods are algorithms that involve estimating the value of different actions or states in order to make decisions (Sutton & Barto 2018). The goal is to associate a value with each possible action or state. This is representing the expected cumulative reward that an agent can obtain by taking that action or being in that state and following a certain policy. A common type of value-based method is a Deep Neural Network which will be used in the Enduro Atari game described in this report. In contrast, policy-based methods focus on learning a policy directly. In other words, the value-based methods focus on finding the optimal policy itself.

In our class project we follow the Actor-Critic reinforcement learning approach, where our model is updated through a combination of policy gradient (actor) and value function (critic) updates.
to play the Atari game Enduro, a car racing game in which the player (agent) races other cars. Then, we evaluate the player's performance by observing the reward history. The overall goal is to pass as many cars as possible in order to achieve as high of a reward (score).

2.Problem Formulation

Enduro races are typically to long-distance endurance tracks, where drivers must overtake a certain number of cars each day to stay on the race. In this project, we will develop an agent that can learn.

Rewards
The reward structure is as follows: The player will begin the game with an initial reward of 0. For every car the agent passes, it receives a +1 reward. In contrast, each car that passes the agent causes in a reward reduction of -1.
State
Additionally, there is a time component to the game. The player will begin at daytime and must achieve the goal—passing 200 cars on the first day and 300 cars every day after that—by nigh time, which marks the end of the first day. In that sense the agent's initial state is the beginning of the day, and terminal state is nightfall and the agent's goal of passing 200 or 300 cars which has or has not been met. Though total kilometers traveled is recorded, the agent does not seek to optimize the number of kilometers traveled; it is more useful at the start of the game with 0 kilometers signaling the start of the game.

Actions
The player has nine actions to choose from, as detailed in the table below:

Value	Meaning	Value	Meaning	Value	Meaning
0	NOOP	1	FIRE	2	RIGHT
3	LEFT	4	DOWN	5	DOWNRIGHT
6	DOWNLEFT	7	RIGHTFIRE	8	LEFTFIRE

4.Method Design

To achieve our goal, an Actor-Critic design will be implemented, as detailed below:

Objective Function

Minimize total loss:
Total Loss = Actor Loss + Critic Loss

Actor Loss = $-\log(\pi(a|s)) \times$ Advantage
Critic Loss = $0.5 \times (\text{Advantage})^2$
Advantage = $R + \gamma V(s') - V(s)$

Where,

- $\pi(a|s)$ is the probability of taking action a in state s, predicted by the actor.
- R is the immediate reward.
- $V(s)$ is the estimated value of the current state s predicted by the critic.

- *V(s')* is the estimated value of the next state s′ predicted by the critic.
- γ is the discount factor.

Input
- Preprocessed image of the game screen.

Actor-Critic Network
Convolutional Layers:
- Convolutional Layer 1: 32 filters, kernel/filter size (5, 5), and ReLU activation.
- Convolutional Layer 2: 64 filters, kernel/filter size (5, 5), and ReLU activation.

- Flattening Layer: The output from the convolutional layers is flattened into a 1D tensor.

Dense Layers:
- Dense Layer 1: 128 neurons and ReLU activation.
- Dense Layer 2: 128 neurons with Softmax activation, producing policy logits.
- Dense Layer 3: 128 neurons and ReLU activation
- Dense Layer 4: 1 neurons as output

Critic Network
- Input Layer: Same convolutional layers as the actor network.
- Hidden Layer: Dense layer with 128 neurons and ReLU activation.
- Output Layer: Dense layer with 1 neuron (value function output).

Learning Process
Initialization:
- Both the actor and critic networks are initialized with random weights.

Training Loop
- The training loop runs for a 60 episodes
- The agent interacts with the environment, and the state, action, reward, and next state are collected.
- The state and next state are processed through the neural networks to get policy logits, values V(s), and next values V(s').

Actor Update
- The actor loss is calculated using the policy logits, the selected action, and the advantage.
- The advantage is calculated as the difference between the discounted next value and the current value.
- The actor loss is the negative log probability of the selected action multiplied by the advantage.

Critic Update
- The critic loss is calculated as the mean squared value of the advantage.

Optimization

- Gradients of the total loss with respect to the model's trainable variables are computed using the gradient descent method.
- The optimizer Adam is used to apply the gradients and update the model's weights.

Output

- Reward values.

5.Experiment

Here is where the agent will be trained to play the game of Enduro. In this stage the agent will be trained through interacting with the game. In the usual manner, the agent will take an action, observe a new state, and collect the reward for moving into that new state; then this process will repeat. Through each step, the Actor-Critic model will update. The design of the experiment put in place to train the agent to play Enduro is as follows: the training run will consist of 60 episodes of the agent playing the game with its initial state being the start of the game, the first day of the game, and the new states being whichever pixels the agent's action places it in. While the agent is not in the terminal state the agent will determine the next action to take, and the resulting state and reward will be stored and added to a total sum of rewards (representing return) for each episode, respectively.

Pseudocode

For 1000 episodes
Parameter used aside from neural network weights: $\alpha = 0.99$
Repeat (for each episode) until maximum number of trials:
    Initialize total reward for episode.
    Get initial state $X(t)$
    While $X(t)$ is not terminal state (that is, for each step t of the episode):
        Choose action: $u(t) \leftarrow nn_a(t)$
        Calculate estimated reward-to-go: $J(t) \leftarrow nn_c(X(t), u(t))$
        Take action and observe reward $r(t)$ and new state $X(t+1)$
        Repeat nn_c, nn_a weight update until max number of iterations reached:
            Calculate $\Delta w_c, \Delta w_a$

$$w_c(t+1) \leftarrow w_c(t) + \Delta w_c(t)$$

$$w_a(t+1) \leftarrow w_a(t) + \Delta w_a(t)$$

        Update state and reward-to-go: $X(t), J(t-1) \leftarrow X(t+1), J(t)$
    Episode finishes when terminal state is reached.
Training ends when maximum number of episodes is reached.

6.Results

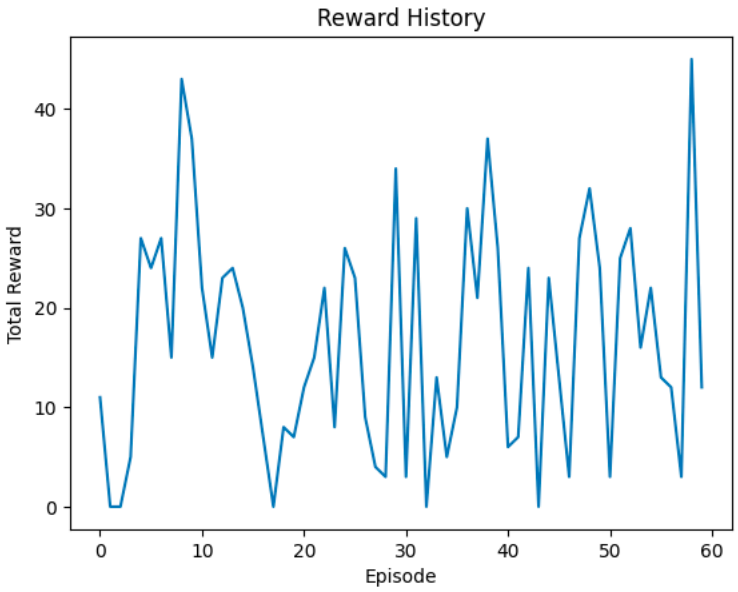
Arcade learning environment:

The implemented Actor-Critic algorithm was trained on the Enduro game environment. The training loop ran for a total of 60 episodes, where each episode involved the agent interacting with the environment and updating the model's parameters based on the observed rewards.

The following table summarizes the total reward achieved by the agent for selected episodes during training:

Episode	Total Reward
0	11.0
10	22.0
20	12.0
30	3.0
40	6.0
50	3.0

The plot below summarizes the agent rewards graphically:

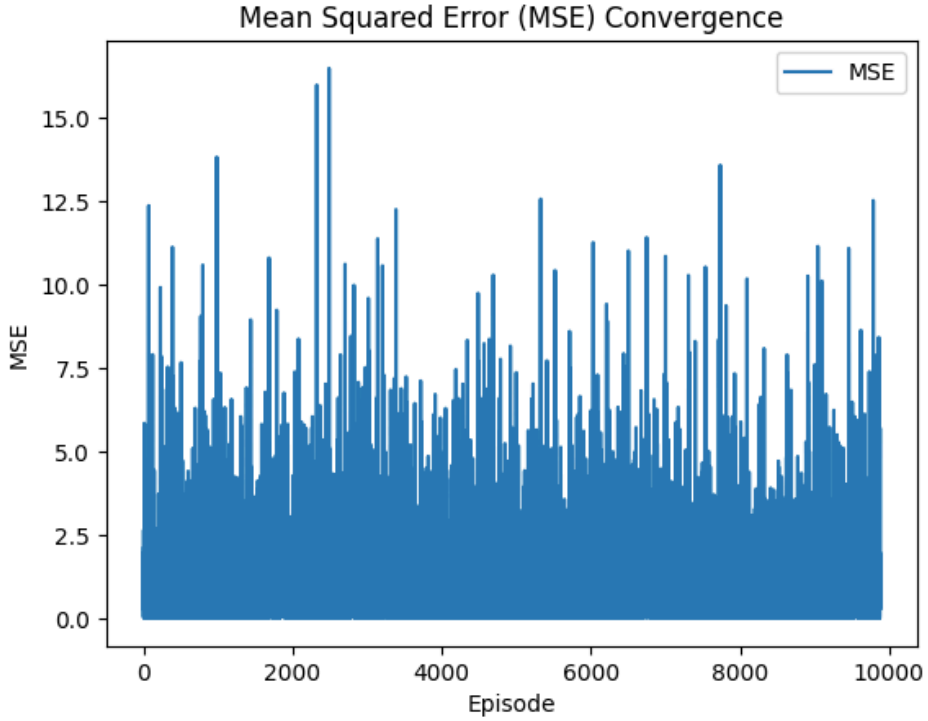


From the table and graph here, we observe that the initial agent action returned promising results. On episode 0 (zero), the reward was 11, which means that the player (agent) passed more cars than they passed it. Further, by episode 10, we see an increase in reward. With a value of 22, this uptick means that the agent has started to grasp more effective strategies for interacting with the environment. The learning process appears to be evolving positively. However, from episodes 18 to 20 we observe learning challenges. Having the overall reward value decline to 12 at episode 20, it is possible that the agent is exploring different strategies or is facing challenges that affect its learning performance. Unfortunately, from here on we observe mixed progress. The total reward increased to 6 in Episode 40 but dropped again to 3 by Episode 50. The reward does eventually increase again, but continues to fluctuate randomly, which indicate that the agent is actively exploring different strategies, and the learning process involves periods of both improvement and regression. While the agent is following its policy of trial an error, we expected to see a positive increase in reward. In other words, in terms of winning, we did not observe that the agent passed 200 cars at any given episode.

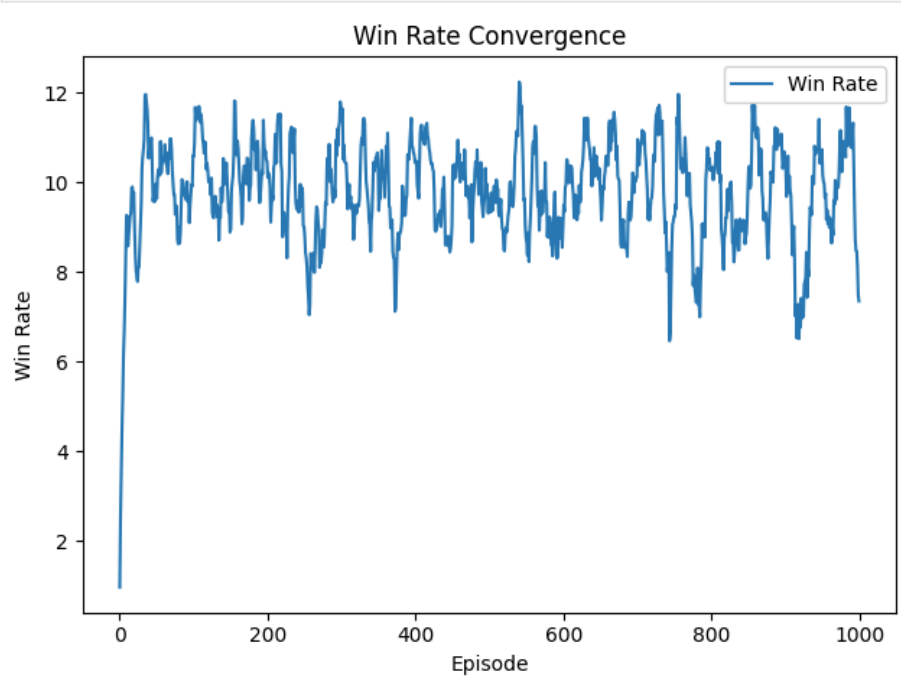
As it was difficult to come up with the 1000 episode result so for shortening the run time, we developed a hypothetical model which can run faster, and results can be analyzed. Below plots are the result visualization.

1. Mean Squared Error (MSE) Convergence
2. Win Rate Convergence
3. Weight Trajectories

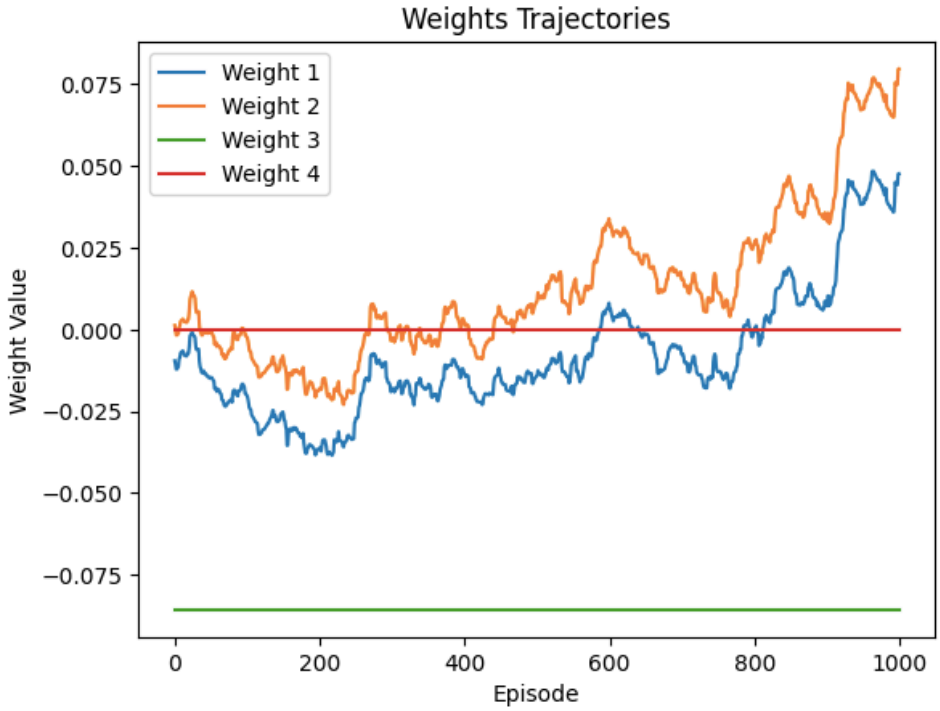
1. **Mean Squared Error (MSE) Convergence** – MSE vs No. of Episode



2. **Win Rate Convergence –** Win Rate vs No. of Episode



3. **Weight Trajectories –** weight value per Episode



7.Roles

James Kos	**Valentin Nechita**	**Javier Nunez**	**Neelima Rajawat**	**Mani Teja Rayana**
Game coding and debugging, Report Writing (Results, Conclusion)	Report writing (Intro, Method Design, Results, Conclusion)	Report writing (Intro, Problem Formulation, Experiment, Results)	Game coding, compilation, debugging, architecture, Report Writing (Results, Conclusion)	Report writing (Problem Formulation, Results, Conclusion)

8.Conclusion

The Atari Enduro Racing experiment aimed to train an agent to play the game of Enduro racing using an Actor-Critic algorithm. Over a training time of 60 episodes, the agent's actions were dictated by its observations of new states and previous and collected rewards. This iterative updating of the Actor-Critic model contributed to agent's learning; however we did not obtain the results we expected. While the initial episodes showed promising results, with the agent swiftly adapting its strategies to outperform opponent cars, the subsequent episodes demonstrated mixed progress. The fluctuations in progress and regress open the door to both challenges and future opportunities. The challenges observed in the learning process, particularly during Episodes 18 to 20, underline the complexity of the Enduro game environment. As far as future opportunities, it is evident that the temporary increases in reward display ongoing exploration and adaptation within the trial-and-error learning approach, meaning that there is room for improvement. Future considerations should focus exactly on that. Classic approaches such as further fine-tuning the training parameters, adjusting the learning rate, or extending the number of training episodes may provide additional insights into enhancing the agent's performance.

Overall, this project provided valuable experience in reinforcement learning and the challenges of properly training an agent. Future work on this project will facilitate a deeper understanding of effective strategies within game environments, and the Actor-Critic model.

9.References

1. Atari Enduro Environment. endtoend.ai. (n.d.). https://www.endtoend.ai/envs/gym/atari/enduro/
2. Farama-Foundation. (n.d.). Farama-Foundation/Arcade-Learning-Environment: The arcade learning environment (ALE) -- A platform for AI research. GitHub. https://github.com/Farama-Foundation/Arcade-Learning-Environment
3. Gymnasium documentation. Complete List - Atari - Gymnasium Documentation. (n.d.). https://gymnasium.farama.org/environments/atari/complete_list/
4. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (n.d.). Playing Atari with Deep Reinforcement Learning.
5. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (Second edition). The MIT Press.