Project Proposal

Project Information

Project:	Deep Reinforcement Learning via Neural Network Agents
Project Time frame:	10/29/2024 to 11/17/2024
Summary:	The goal of this project is to extend traditional reinforcement learning agents to utilize deep reinforcement learning by enhancing the agent with a neural network model using the Gymnasium library in the Python programming language. Traditional preference algorithms such as Monte Carlo, Temporal Difference, SARSA or Q-Learning learn the optimal preference strategy without requiring a neural network. Deep reinforcement learning strategies involving a neural network agent exist that can learn to predict the optimal state-value and action-value functions used within the Q preference learning algorithm. For this purpose we will use the reinforcement learning library Gymnasium, the neural network library Pytorch, and the deep reinforcement learning library agileRL in Python within the Google colaboratory computational environment. Thus, we aim to train a deep neural network to learn the optimal reinforcement learning strategy within Gymnasium's Blackjack or complex Atari game environments.
Feedback or Comments? Reach out to:	Alexis Bryan Ambriz | bryan.ambriz@sjsu.edu , Suresh Ravuri | suresh.ravuri@sjsu.edu, Sri Vinay Appari | srivinay.appari@sjsu.edu

Process impact: This proposal, along with drafts of related documents, will be used by the professor to determine whether or not to approve work on this project. A clear and precise project plan helps set expectations that will be used later to evaluate the success of the project.

Background and Motivation

1. What is the setting and history behind this project?

Reinforcement learning's foundational principles via classical conditioning emerged from behavioral psychology research in the early 20th century, particularly through the work of researchers like Thorndike (1911) and Skinner (1938). Classical conditioning was first systematically studied by Ivan Pavlov in his experiments with dogs (1927), where he discovered that dogs would salivate not only at the sight of food but also at the sound of a bell that had been repeatedly paired with food presentation. While classical conditioning, demonstrated by Pavlov's experiments, showed how organisms learn associations between stimuli, operant conditioning—more closely related to modern reinforcement learning—demonstrated how behaviors are modified through consequences. In reinforcement learning, an agent learns optimal behavior through interactions with an environment, receiving rewards or penalties that shape its decision-making process.

The mathematical framework often used to formalize reinforcement learning is the Markov Decision Process (MDP), a term first introduced by Richard Bellman in 1957 when he built upon Andrey Markov's earlier work on state transition probabilities to develop a framework for sequential decision-making under uncertainty. Traditional reinforcement learning algorithms such as the Monte Carlo, Temporal Difference, SARSA, and Q-Learning algorithms make use of the mathematical definition of an MDP. These algorithms allow an agent in an environment to learn a decision following strategy over time by applying linear state-value and action-value or Q-Learning functions to estimate the goodness of a state or action.

2. What is the problem to be addressed?

Traditional reinforcement algorithms and agents are inefficient and lack the ability to pick up complex patterns from an MDP environment, such as in Gymnasium’s Blackjack and Atari environments. These learning algorithms (i.e SARSA or the Q-Learning strategies) use a strategy that utilizes state and action value expectation functions to influence the Q-learning algorithm. A limitation of these approaches is that these traditional algorithms use simple linear functions to modify the Q strategy matrix.

3. What are some current approaches to this problem?

In deep reinforcement learning, neural networks emulate the strategy for the agent within the MDP framework to approximate the value functions via non-linear regression, allowing the agent to learn and make decisions in complex state-action mappings or high-dimensional state spaces. The Gymnasium and the agileRL libraries in Python are popularly used in both reinforcement learning and deep reinforcement learning. Gymnasium provides a variety of simple and complex action and state spaces or environments that can be used to contextualize an MDP, while agileRL provides a variety of neural network architectures and training strategies for training the deep reinforcement learning agent. In Gymnasium's CartPole environment using the Q-Learning strategy, for example, the actions are discrete while the states are continuous and require discretization and binning to be able to map the states to actions in the Q-Learning matrix.

4. Why is this problem worth solving or worth solving better?

Deep reinforcement learning provides more flexible, accurate, and efficient learning within Gymnasium's more complex environments, or within custom environments. Traditional reinforcement learning algorithms that require converting the continuous states or actions into discrete values will inherently lose valuable information in the process. Deep learning via neural networks solves this problem as it is able to understand the more complex, continuous input. Our project aims to solve the learning problems posed in Gymnasium's Blackjack and potentially more complex Atari environments to leverage the benefit of deep reinforcement learning.

Goal

Scope

We want to focus on creating a solution oriented deep reinforcement learning agent, created out of Reinforcement and Deep Learning Techniques. The agent should be able to solve problems faced within a gaming environment.

In Scope	Out of Scope
Developing a deep reinforcement learning agent to operate within gaming environments.	Building a new python library that supports chat bot functionality
Working the most popular IDEs and environments like Anaconda for development phase	Working with uncommon or complicated IDEs and environments for development

Data trained on open source data set and self analyzed common questions	Paid dataset derived from APIs and other services

Deliverables

- **Deep Reinforcement Learning Agent**: A trained RL agent capable of navigating and making optimal decisions in the Blackjack environment.
- **Neural Network Model for Decision Making**: A neural network that integrates with the RL agent to predict optimal moves based on previous training.
- **Documentation**: Detailed technical documentation for setting up, training, and deploying the chatbot.

Risks and Rewards

What are the main risks of this project?

- **Model Complexity**: Deep RL agents may require extensive computational resources, and complex models can become inefficient without adequate optimization.
- **Training Time**: The training process, especially in a Blackjack environment, can be time-consuming and resource-intensive, potentially leading to delays.
- **Reward Function Tuning**: Incorrectly tuned reward functions might lead to suboptimal decision-making, affecting the chatbot's performance.

What are the main rewards if this project succeeds?

- **Efficient Training**: If optimized, Deep RL could reduce the need for manual feature engineering and potentially enhance the chatbot's ability to learn from dynamic environments.
- **Scalability of Solution**: Once a functional RL chatbot is built, it can serve as a foundation for more advanced agents, expanding its use cases in other games or customer service scenarios.