International Journal For Technological Research In Engineering
Volume 9, Issue 10, June-2022 *ISSN (Online): 2347 - 4718*

IMPLEMENTATION OF A REINFORCEMENT LEARNING BOT FOR GAME PROBLEM SOLVING

[1]Sagnik Halde, [2]Parth Bhatia, [3]Jaipal Singh , [4] Prof.Gurpreet Kaur
[1,2,3]Students, [4]Assistant Professor
Bhagwan Mahaveer College of Engineering and Management, Sonipat, India

Abstract-This report is about studying the basic components of Reinforcement Learning technology such as: Reward: The reward is a scalar feedback signal which indicates how well the agent is doing at step time t. In reinforcement learning we need define our problem such that it can be applied to satisfy our reward hypothesis. An example would be playing a game of chess where the agent gets a positive reward for winning a game and a negative reward for losing a game. Reward Hypothesis: All goals can be described by the maximisation of expected cumulative reward. Since our process involves sequential decision making tasks, our actions we make early on may have a long-term consequence on our overall goal. Sometimes it may be better to sacrifice immediate reward (reward at time step Rt) to gain more long-term reward. An example applied to chess would be to sacrifice a pawn to capture a rook at a later stage. Since then, this topic is declared to research and create an online game, Our project is modelled after the popular online multiplayer game haxball. HaxBall is a physics-based multiplayer soccer game where teamwork is key. In this example however we are trying to build a model that is capable of learning on how to play and become good enough to beat other strong players in 1v1 matches. This model will make use of python libraires like numpy, pandas, gym, pygame and frameworks like Google's TensorFlow and OpenAI baselines to build a system that can be used to train multiple models in parallel without the need of any manual play

Introduction

Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty. In Reinforcement Learning, the agent learns automatically using feedbacks without any labelled data, unlike supervised learning. Since there is no labelled data, so the agent is bound to learn by its experience only. RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc. The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards. The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that "Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that." How a Robotic dog learns the movement of his arms is an example of Reinforcement learning. It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention. Example: Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback. The agent continues doing these three things (take action, change state/remain in the same state, and get feedback), and by doing these actions, he learns and explores the environment. The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative 1 point. The objectives of this thesis were to illustrate and understand the fundamental concepts and usage Fundamental Concepts of Reinforcement Learning, as well as their compatibilities and advantages as compared to other machine learning algorithms. The thesis achieved that goal by Increasing the level of abstraction, the scenario we're considering can be described as: a generic agent performs actions inside an environment and receives feedback that is somehow proportional to the competence of its actions. According to this feedback, the agent can correct its actions in order to reach a specific goal. This document structure was organized as follow. The first section brought in the goal of the thesis and technologies used. Next, essential concepts and theoretical background of each technology in the stack was introduced along with example, followed by the third section which demonstrated carefully and thoroughly the application development process, from back-end to front-end. In the end, this paper provided discussion of the project with further improvements and gave conclusion about the final product.

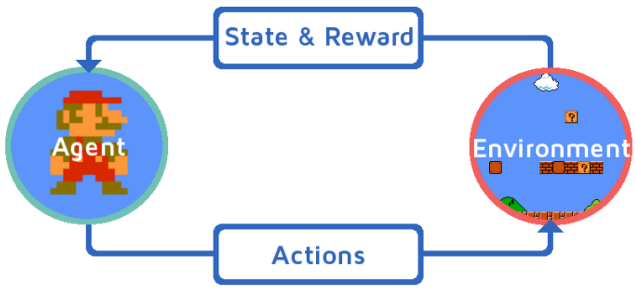
PROBLEM STATEMENT

The goal of this project is to implement a bot using Reinforced Learning (DQN) in python using OPENAI baselines and other python data science and simulation libraries that will surpass an above average player in the popular football/air hockey hybrid game haxball and be a proof of concept about the application of reinforcement learning in other fields. The Project will have to replicate haxball's game system and then devise methods that can be used to better train the model

efficiently taking least resources possible for the operation. The problem also involves finding the optimal algorithm that can be used for the said problem and will make a comparative analysis of the performances to set up benchmarks for future use.

What is reinforcement Learning

Reinforcement learning is the training of machine learning models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning, an artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward. Although the designer sets the reward policy–that is, the rules of the game–he gives the model no hints or suggestions for how to solve the game. It's up to the model to figure out how to perform the task to maximize the reward, starting from totally random trials and finishing with sophisticated tactics and superhuman 2 skills. By leveraging the power of search and many trials, reinforcement learning is currently the most effective way to hint machine's creativity. In contrast to human beings, artificial intelligence can gather experience from thousands of parallel gameplays if a reinforcement learning algorithm is run on a sufficiently powerful computer infrastructure. Examples of reinforcement learning Applications of reinforcement learning were in the past limited by weak computer infrastructure. However, as Gerard Tesauro's backgamon AI superplayer developed in 1990's shows, progress did happen. That early progress is now rapidly changing with powerful new computational technologies opening the way to completely new inspiring applications. Training the models that control autonomous cars is an excellent example of a potential application of reinforcement learning. In an ideal situation, the computer should get no instructions on driving the car. The programmer would avoid hard-wiring anything connected with the task and allow the machine to learn from its own errors. In a perfect situation, the only hard-wired element would be the reward function. For example, in usual circumstances we would require an autonomous vehicle to put safety first, minimize ride time, reduce pollution, offer passengers comfort and obey the rules of law. With an autonomous race car, on the other hand, we would emphasize speed much more than the driver's comfort. The programmer cannot predict everything that could happen on the road. Instead of building lengthy "if-then" instructions, the programmer prepares the reinforcement learning agent to be capable of learning from the system of rewards and penalties. The agent (another name for reinforcement learning algorithms performing the task) gets rewards for reaching specific goals.



Pros of Reinforcement Learning

- Reinforcement learning can be used to solve very complex problems that cannot be solved by conventional techniques.
- This technique is preferred to achieve long-term results, which are very difficult to achieve.
- This learning model is very similar to the learning of human beings. Hence, it is close to achieving perfection.
- The model can correct the errors that occurred during the training process.
- Once an error is corrected by the model, the chances of occurring the same error are very less.
- It can create the perfect model to solve a particular problem.
- Robots can implement reinforcement learning algorithms to learn how to walk.
- In the absence of a training dataset, it is bound to learn from its experience.
- Reinforcement learning models can outperform humans in many tasks. DeepMind's AlphaGo program, a reinforcement learning model, beat the world champion Lee Sedol at the game of Go in March 2016.
- Reinforcement learning is intended to achieve the ideal behavior of a model within a specific context, to maximize its performance.
- It can be useful when the only way to collect information about the environment is to interact with it.
- Reinforcement learning algorithms maintain a balance between exploration and exploitation. Exploration is the process of trying different things to see if they are better than what has been tried before. Exploitation is the process of trying the things that have worked best in the past. Other learning algorithms do not perform this balance.

Cons of Reinforcement Learning

- Reinforcement learning as a framework is wrong in many different ways, but it is precisely this quality that makes it useful.
- Too much reinforcement learning can lead to an overload of states, which can diminish the results.
- Reinforcement learning is not preferable to use for solving simple problems.
- Reinforcement learning needs a lot of data and a lot of computation. It is data-hungry. That is why it works really well in video games because one can play the game again and again and again, so getting lots of data seems feasible.
- Reinforcement learning assumes the world is Markovian, which it is not. The Markovian model describes a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

- The curse of dimensionality limits reinforcement learning heavily for real physical systems. According to Wikipedia, the curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience.
- Another disadvantage is the curse of real-world samples. For example, consider the case of learning by robots. The robot hardware is usually very expensive, suffers from wear and tear, and requires careful maintenance. Repairing a robot system is costs a lot.
- To solve many problems of reinforcement learning, we can use a combination of reinforcement learning with other techniques rather than leaving it altogether. One popular combination is Reinforcement learning with Deep Learning.

Difference between Reinforcement Learning and Deep Learning

The main difference between reinforcement learning and deep learning is this: Deep learning is the process of learning from a training set and then applying that learning to a new data set. But reinforcement learning is the process of dynamically learning by adjusting actions based on continuous feedback to maximize a reward.

Deep learning makes use of the existing available data and uses that data to predict patterns. Reinforcement learning can learn from its experience through trial and error.

Applications of Reinforcement Learning

A variety of problems can be solved using reinforcement learning. Some of them are game-playing, robotics, and many other fields.

As I mentioned earlier, reinforcement learning is the best technology used for game playing. It can even beat world champions.

Reinforcement learning can be used effectively to determine the best move to make in a game, depending on several different factors. It is very handy in games like Chess, Go, etc.

Using reinforcement learning, we can improve and personalize the gaming experience in real-time. It is the algorithm that can solve different games and sometimes achieve super-human performance.

This technology is used for the learning of robots. Robots are trained using the trial and error method with human supervision. Reinforcement learning teaches robots new tasks while retaining prior knowledge.

E-commerce websites like Amazon can use reinforcement learning to solve their problems to generate the maximum revenue by displaying the most relevant ads to interested buyers.

Self-driving cars also implement some reinforcement learning algorithms. Reinforcement learning can also be applied to optimizing chemical reactions

RESEARCH AND DEVELOPMENT

There were many applications for constructing a reinforcement learning bot, and we used OpenAI baselines and py gym technologies to develop a bot in this study. The libraries being used to handle the training data and process it are standard data science libraries like Numpy, Pandas and Tensorflow. These libraries provide a well documented set of tools that can be used to process n-dimensional at superb speeds.

MATH BEHIND REINFORCEMENT LEARNING

States & Rewards

Let's consider a sequence of states S1, S2, …, Sn each of them has some kind of reward R1, R2,…,Rn. An agent has the job to maximise its total reward. It will choose the states-path that provides the maximum rewards.

Suppose the agent is at a random state, there should be a way for it to know what is the best path that maximises its reward. The challenge here being that the agent does not see beyond its immediate neighbouring states. Thus, in addition to the reward of each state S, we are also going to store V.

V represents rewards of other states to which each state is connected to.

Example, V1 represents the total rewards of all the states connected to S1. The reward R1 is not part of V1. But the reward R2 at S2 is part of V1 at S1.

This way by simply looking at the next state, the agent will have an idea what lays behind.

The value V stored at state S is computed via a function called "Value Function". The Value function computes the future rewards. The final states, also called terminal states, do not have value V since there are no future states or rewards.

The Importance of States & Rewards in Reinforced Learning

$$V(s) = \sum_{t} \gamma^{t} R(S_t)$$

This will create a sequence of increasing V from the origin till the end, which constitutes a hint to the agent on which direction maximises its reward. So, V(s), the Value Function returns the future rewards coming from other states.

Here, St are all the states that are connected directly or indirectly to S. Computationally it becomes more efficient to calculate V(s) (current state) when we know the V(s') next states, instead of summing all the rewards of all future states. The Equation now becomes :

$$V(s) = r + \gamma * V(s')$$

So far we have assumed that all states are connected in sequence, however this is rarely the case, each state can be connected to multiple other states to which the agent can potentially move to. Let S1 be connected to S2, S3, S4, and S5, the value V at S1 should reflect this situation.

$$V(S1) = (R2+R3+R4+R5 + \gamma *(V2 + V3 +V4+V5))/4.$$

V(S1) is the average of all values of states to which it is connected. This suggests that from S1 we can go to S2, S3, S4, and S5 without any preference to any particular state.

This is not accurate, because we know that there is a certain probability to go to each neighbour state and these probabilities might not be the same.

Lets call these probabilities p2, p3, p4, p5 respectively. So V(S1) becomes

$$V(S1) = p2*R2+p3*R3+ p4*R4+p5*R5 + \gamma *(p2*V2 + p3*V3 +p4*V4+p5*V5)$$

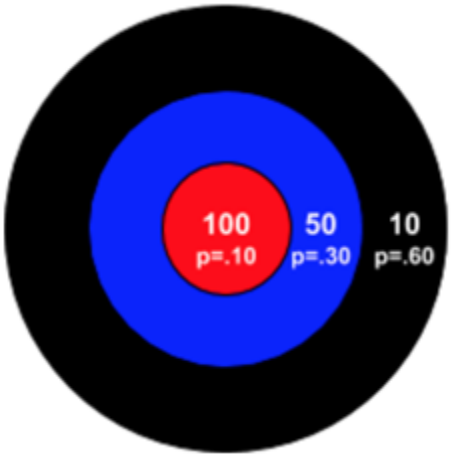
These probabilities are called transition probabilities and they express the likelihood of transitioning from one state to another. To express them as a matrix P where Pij is the probability of transition from a state i to a state j. When no transition is possible the Pij will be zero. We now have transformed the equation to :

$$V(s) = \sum_k P(s_k|s)\,(R_k + \gamma\, V(s_k))$$

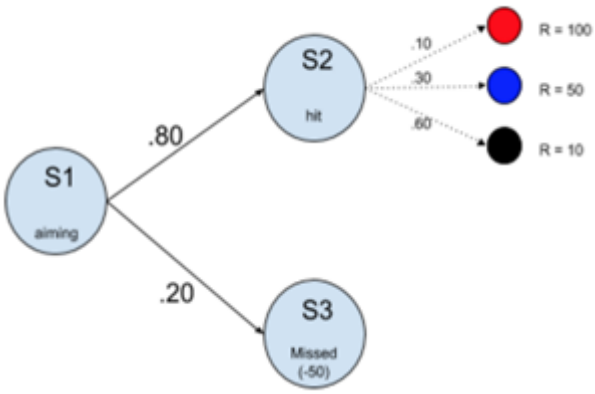
Here P(Sk|S) is the probability of reaching state Sk knowing our current state.

Stochastic Rewards

The reward itself is not deterministic, which means you can't assume that R is precisely known at every state. In fact it is probabilistic and can take different values. Consider the aim on a target, we will suppose there are only three states, S1 (aiming), S2 (hit), S3 (miss).



Thus the probability in such a case won't be uniform.



So the value at state S1 will be :

$$V(S1) = .8*(.1*100 + .3*50 + .6*10 + \gamma *V(S2)) + .2*(1*-50 +\gamma * V(S3))$$

Since S2 and S3 are terminal states, then V(S1) and V(S2) are zero, but they are mentioned above to keep reminding of the general formula.

The rewards in every state are multiplied, then summed together and each state is multiplied by the transition probability that leads to it.

$$V(s) = \sum_i p(s_i|s) \sum_j p(r_j|s)(r_j + \gamma V(s_i))$$

$$V(s) = \sum_i \sum_j p(s_j|s) p(r_j|s)(r_j + \gamma V(s_i))$$

$$V(s) = \sum_i \sum_j p(s_i, r_j|s)(r_j + \gamma V(s_i))$$

From the above we can deduce the general formula:
Here, p(si, rj|s) is read as the probability of transiting from state s to si with a reward rj. We can thereby generalize and state

$$V(s) = \sum_{s' \in S} \sum_{r \in R} p(s', r \mid s)\,(r + \gamma V(s'))$$

Actions and Policy

The probability of going from state s after performing action a, to the state s' and getting reward r is not 100%. That's why we write p(s',r|s, a) which is the probability of transiting to state s' with reward r given a state s and an action a.

The strategy that dictates which action to use at a certain state is called policy. As usual we quantify this in the V(s) by averaging all these possibilities.

$$V_\pi(s) = \sum_a \pi(a|s) * f(a, s, r)$$

(a|s) is the probability of using action a following the policy π given that we are at state s.
V$\pi\pi$(s) is the value at state s when applying policy $\pi\pi$.
f(a, s, r) is used here as a shorthand for the Value function V(s).
The use of f(a, s, r) is simply meant to reduce the complexity and emphasise the role of $\pi\pi$(a|s).

Upon assembling together, we achieve :

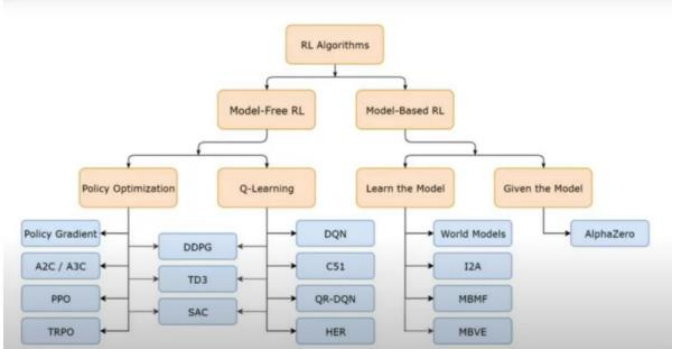
$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s' \in S} \sum_{r \in R} p(s', r \mid s, a)\,(r + \gamma V_\pi(s'))$$

APPLICATION DEVELOPMENT

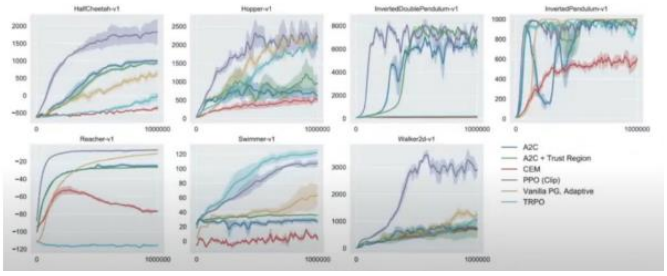
Application Development This section is dedicated to demonstrate the functionalities development process of learning using the reinforcement based algorithms applied from the Open-AI baselines packages and python's gym, the first piece of code is just an attempt to create a base on which further improvementst can be made later on. Back-end Development Basic Setup The very first thing to do is to set up an Python application by creating 16 an open.py file, which is the entrance file of the project

CHOOSING THE RIGHT ALGORITHM
The first run of the application concluded that the parameters which can be configured that is the reward function has to be changed to make sure the model performs properly. A choice was first made to choose between a positive reward function and a negative reward function.

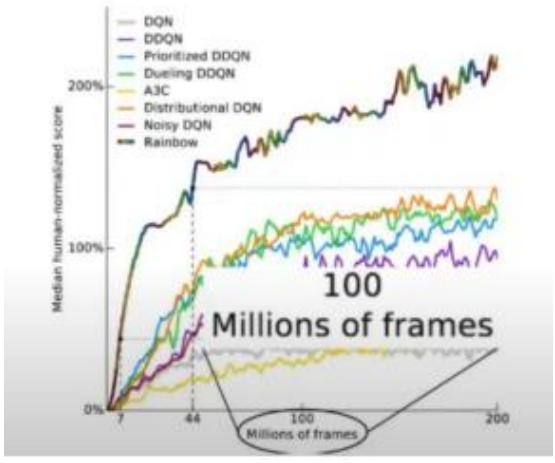
Choosing between positive and negative reward function



Reinforcement Learning Algorithms



Reinforcement Learning Algorithm comparison in one diagram



Reinforcement Learning comparison

CHOOSING THE RIGHT TRAINING METHODOLOGY

Self play aka the model in a match versus itself, However it comes with a lot of caveats of its own. Below are some of the caveats and shortcomings of using self play

i) Possible Overfitting
ii) No effective play quality measure
iii) No effective way to compare different sets of parameters

A counter plan to improve upon this and to get rid of almost all downsides was however devised and was quickly taken into consideration, aka to have a tournaments and let the best models compete against each other, this approach however comes with another downside of taking more resources and hence a self play evolution Tournament was thought of in which
Agents play against many strategies, this approach reduces the chances of overfitting but at the same time possibly introduce rating into the equation, but overall it is the way go when comparing 2 types of models.

FINAL CONFIGURATION AND RESULTS

The final convolutional neural network turned out as the following

Input/Output
-14 state dimensions (coordinates, velocities…)
-10 actions (directions, space bar, "no action")

Implementation Details
-Deep neural network: 4 layers X 128 neurons, (Tensorflow)
-PPO2 algorithm (OpenAI)
-Pre-training on self-play (400M frames on 10FPS)
-Round-Robin tournament(~ 20 different models at a time)

CONCLUSION

The goal of this thesis was to study different characteristic of each technology available in the OpenAI-baselines repo and using that build a reinforcement learning system that can peform on par to an above average player and can outperform most of the the human players. The application was successfully developed at the end. A fully functional end-to-end game replica was built and a methodologies were developed to successfully train the model. This application was meant to solve the problem that is mentioned in the first section of this thesis: to help devise a reinforcement learning system capable of outperforming an above average player. Overall, the thesis can be used as a tutorial or documentation of the openAI baselines, python gym and other libraries involved and has opened pathways for further research and development of applications in this field. Although the application still has some drawbacks and needs more further improvements, both in styling issue and new features, it is a combination of one of the most widely used artificial intelligence technology with one of the most emerging business ideas nowadays – Machine Learning

References

1 "Proximal Policy Optimization from" OpenAI-baselines , OpenAI 2022. San Francisco, California, Pioneer Building, San Francisco, California, US https://beta.openai.com
2 TensorFlow 2.0 2022 "An end-to-end open source machine learning platform" https://www.tensorflow.org/
3 Cython optimising static compiler for both the Python programming language and the extended Cython programming language https://cython.readthedocs.io/en
4 Python Python is a high-level, general-purpose programming language https://www.python.org/
5 Keras open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library https://keras.io/
6 Numpy library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays https://numpy.org/
7 Pandas, software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series https://pandas.pydata.org/

