International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 07 | July 2022 www.irjet.net p-ISSN: 2395-0072

GAMING BOT USING REINFORCEMENT LEARNING

ROHINI KHALKAR[1], SUMUKH DHAWADE[2], DEVANG PURI[3], APOORV NARANG[4]

[1]Assistant Professor, Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India
[2, 3, 4]Student, Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India

---

Abstract - *Because of its widespread use in a variety of game genres and its ability to simulate human-like behaviour, intelligent agent training has piqued the interest of the gaming industry. In this paper, two machine learning techniques, namely a reinforcement learning approach and an Artificial Neural Network (ANN), are used in a fighting game to allow the agent/fighter to simulate a human player. We propose a customised reward function capable of incorporating specific human-like behaviours into the agent for the reinforcement learning technique. A human player's recorded bouts are used to educate the ANN. The proposed methods are compared to two previously published reinforcement learning methods. Furthermore, we provide a comprehensive overview of the empirical assessments, including the training procedure and the primary characteristics of each approach used. When compared to other current approaches, the testing results revealed that the proposed methods outperform human players and are more enjoyable to play against.*

1. INTRODUCTION

The AI bot's logic is based on Reinforcement Learning Techniques. This method considers the game's previous state and rewards it (such as the pixels seen on the screen in case of the Atari Games or the game score in case of card games like Blackjack). It then devises a plan of action against the environment. The goal is to make the next observation better (in our example, to increase the game score). An agent (Game Bot) selects and performs this activity with the goal of increasing the score. It is then exposed to the environment. The environment records the outcome and reward based on whether the activity was beneficial (did it win the game?).

This project aims to develop a virtual assistant that can learn a variety of games, beginning with simple card games like Blackjack and progressing to more complex video games in the future. The AI bot uses Reinforcement Learning to train and improve its level of experience in the game. When compared to other machine learning technologies, RL is best suited for developing complex bot behaviour within a game. Like a human player, the learning bot interacts with its surroundings and receives feedback based on its actions.

As feedback, a reward for a successful action or a loss of life/points for a bad action can be used. The bot's behaviour is then improved based on the incentives and punishments it receives. This learning method is similar to how humans learn by trial and error, and it works well in complex game situations. Players are drawn to online player versus player games because human players add an element of surprise that traditional bots do not. This problem is effectively handled by RL. The main goal of the project was to find solutions to a variety of simulations that could one day be real-life challenges rather than games.

This game-playing bot's activities will never be limited to standard steps taken by players, but will instead create its own methods and new variations, making it impossible for a player to simply memorise its moves and beat it.

1.1. SOME FREQUENTLY USED TERMINOLOGIES:

(1) Agent: The person who solves problems can take certain actions in a given setting.

(2) Environment: The residence of an agent. An environment provides responses to an agent based on the actions it performs.

(3) Reward: When an agent performs an action in an environment, there is an associated reward; rewards can be positive, negative(punishment) or zero.

(4) State: An agent's action may cause it to enter a state which you can assume to be a snapshot (partial or whole) of the environment. (Like checkmate (state) on a chess board (environment)).

(5) Policy: Defines an agent's behaviour, can answer questions like *what action should be performed in this state?*

(6) Value**:** Tracks the long-term impact of the action. Provides a portion of reward to intermediate states that led to a final positive stat.

2. BLACKJACK

2.1. HISTORY OF BLACKJACK

Blackjack is the American version of a popular global banking game known as Twenty-One. It is a comparing card game between one or more players and a dealer, where each player in turn competes against the dealer.

Players do not compete against each other. It is played with one or more decks of 52 cards, and is the most widely played casino banking game in the world.

3. RULES OF BLACKJACK

[1] Blackjack is a card game in which the objective is to get as close to 21 as possible without going over. They're up against a predetermined dealer.

[2] The value of the face cards (Jack, Queen, and King) is ten points. Aces can be counted as 11 or 1, and the number 11 is referred to as 'usable'. This game is set up with an unlimited deck of cards (or with replacement). The game begins with one face up and one face down card for each player and dealer.

[3] The player can request more cards until he or she decides to stop or reaches the limit of 21. (bust). The dealer displays their facedown card and draws until their amount is 17 or larger after the player sticks. The player wins if the dealer goes bust. If neither the player nor the dealer busts, the winner (or loser) is determined by whose total is closest to 21.

[4] Winning gives you +1, drawing gives you 0, and losing gives you -1.

3. PROBLEMS WITH TRADITIONAL ML GAMING ALGORITHMS

Following that, game developers attempted to simulate how humans play a game by developing a gaming bot that mimicked human intelligence. As a result, machine learning (ML) trained bots were created, which were trained using a variety of ML techniques.

This project will attempt to teach a computer (reinforcement learning agent) how to play blackjack and outperform the average casino player, with the goal of eventually moving on to other games and features.

Most traditional ML-based gaming models have the problem of being programmed to respond to hardcoded rules or patterns that are usually followed by players, but because humans are unpredictable, these ML Models fail to keep up and constantly improve themselves.

4. OUR APPROACH TO THE PROBLEM

To address this issue, Q-learning, a reinforcement learning technique, will be used. For each state-action pair, a Q-table is created, and the related entry in the Q-table is changed based on the prize won after each round of the game. After the agent has thoroughly studied the environment, the learning process is complete. At this point, we'd have the optimised Q-table, which is the blackjack strategy the agent has learned. It is a reasonable choice as a performance measure because a prize is awarded at the end of each round of the game. To assess the efficacy of various approaches, a large number of rounds must be played in order to get close to their true rewards. As a result, the average reward after 1000 rounds of the game will be used to compare the simulated performance of the typical casino player to that of the trained agent.

We mainly use Q-learning, Monte Carlo Prediction and Actor-Critic Learning to overcome the problems faced by traditional ML Models. Actor-Critic Learning constantly evolves the machine learning model by using reward-punishment mechanism with a learning agent.

4.1 Q-LEARNING

Quality is represented by the letter 'q' in q-learning. In this situation, quality refers to how valuable a specific activity is in obtaining a future reward. Given a current state, Q-Learning is a Reinforcement learning policy that will determine the next best action. It selects this action at random with the goal of maximising the reward.

Given the present state of the agent, Q-learning is a model-free, off-policy reinforcement learning that will determine the best course of action. The agent will pick what action to take next based on where it is in the environment. The model's goal is to determine the optimum course of action given the current situation. To accomplish this, it may devise its own set of rules or act outside of the policy that has been established for it to obey. This means there isn't a real need for a policy, which is why it's called off-policy.

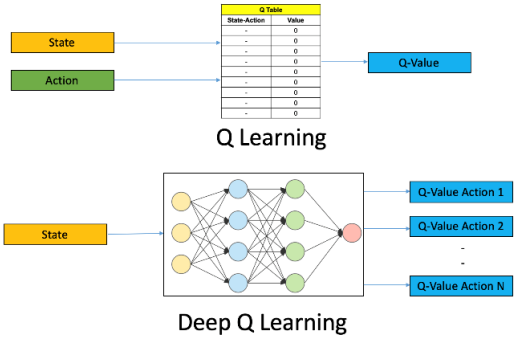


Fig -1: Q-Learning Model

Model-free indicates that the agent moves forward based on estimates of the environment's expected response. It does not learn through a reward system, but rather by trial and error.

4.2 MONTE CARLO PREDICTION

Monte Carlo simulation is a computer-assisted mathematical technique for incorporating risk into quantitative analysis and decision-making. Professionals in a variety of sectors, including finance, project management, energy, manufacturing, engineering, research and development, insurance, oil and gas, transportation, and the environment, employ the technique. For each given course of action, Monte Carlo simulation provides the decision-maker with a range of probable outcomes as well as the probabilities that they will occur.

It depicts the extreme outcomes— the implications of going for broke and the most conservative decision—as well as all possible outcomes for middle-of-the-road selections. Scientists working on the atomic bomb were the first to employ the technology, which was named after Monte Carlo, the Monaco vacation town known for its casinos. Monte Carlo simulation has been used to model a range of physical and conceptual systems since its beginnings during World War II.

CODE:

```
[ ]  def generate_episode_from_limit_stochastic(bj_env):
         episode = []
         state = bj_env.reset() #here, bj_env is the env instance
         while True:
             probs = [0.8, 0.2] if state[0] > 18 else [0.2, 0.8]
             action = np.random.choice(np.arange(2), p=probs)
             next_state, reward, done, info = bj_env.step(action)
             episode.append((state, action, reward))
             state = next_state
             if done:
                 break
         return episode
```

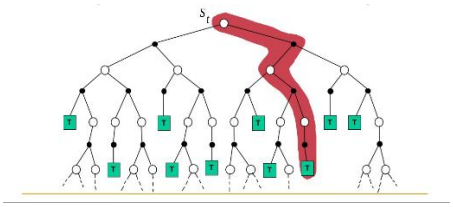


Fig -2: Monte Carlo Prediction

4.3 ACTOR-CRITIC LEARNING

Work with a parameterized family of policies while using actor-only techniques. The actor parameters are directly evaluated by simulation to determine the performance gradient, and the parameters are then updated in an upward manner.

Such approaches may have a disadvantage in that the gradient estimators may have a high variance. Additionally, a new gradient is estimated independently of earlier estimates as the policy changes. As a result, there is no "learning" in the sense of gathering and consolidating prior knowledge.

The goal of critic-only approaches, which solely use value function approximation, is to discover a rough solution to the Bellman equation, which will hopefully subsequently suggest a close to optimal course of action.
Such approaches are indirect because they don't directly attempt to maximise over a policy space. This kind of approach might be successful in creating a "decent" approximation of the value function, but it lacks trustworthy assurances for the near-optimality of the resulting policy.

The advantages of actor-only and critic-only methodologies are intended to be combined in actor-critic methodologies. The critic learns a value function through simulation and an approximation architecture, then uses that knowledge to modify the actor's policy parameters.

Actor-Critical Algorithms 1009 are directed toward enhancing performance. As long as they are gradient-based, such approaches may have desirable convergence features as opposed to critic-only methods, where convergence is only possible in very specific circumstances.

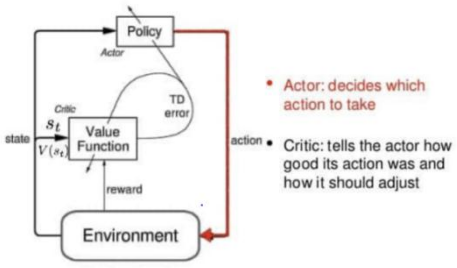


Fig -3: Actor-Critic Learning Model

When compared to actor-only approaches, they have the potential to achieve faster convergence (owing to variance reduction). On the other hand, lookup table representations of policies have been the only focus of theoretical understanding of actor-critic approaches.

The actor uses the state as input and produces the best action. It effectively regulates the agent's behaviour by teaching it the best course of action (policy-based). On the other side, the critic computes the value function to assess the action (value based).

These two models play a game where they both improve over time in their respective roles. As a result, the whole

architecture will develop better game skills than the two individual ways.

OPTIMIZATION:

One way we can improve is by using reward discounting to more intelligently allocate rewards for actions. It is best to support the most recent action or frame in the case of a positive reward and to discourage it in the case of a negative reward because it is the most pertinent. An exponential factor discounting factor, (gamma), with an initial value of 0.99, assigns less credit to any activities or time frames that occur before the reward was earned.

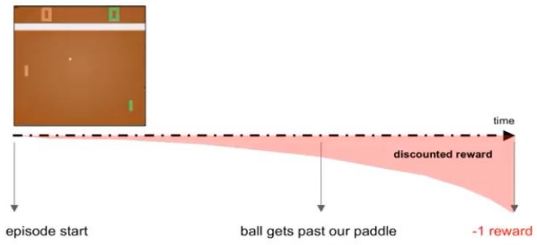


Fig -4: Representation of Optimization

The effect of discounting rewards — the -1 reward is received by the agent because it lost the game is applied to actions later in time to a greater extent

Discounting helps us come closer to our goals by more precisely assigning the reward to the behavior that is probably a significant contribution to the benefit.

In the case of Pong and other Atari games, the policy function, or policy network, used to determine what to do next based on the input, is a fully-connected Neural Network with hidden layers.

The NN receives a game frame image as input, where each pixel represents a separate input neuron, and returns a decimal probability between 0 and 1, which translates to the likelihood that the action our agent previously took for a related image was a UP move. It should be noted that the probability of the agent making a DOWN motion increases if the probability is below 0.5 and gets closer to 0. This policy is stochastic since it produces a probability. The number of units in the hidden layer is a hyperparameter of the system which is decided by Optuna.

Since the chance that a certain action should be executed over time is more likely to lie towards the extremes of 0 or 1, the randomness factor becomes less important as the NN becomes more trained and the policy average reward increases.

Another advantage of the NN approach is that it doesn't matter if the previously described map hasn't actually seen the input image because a large portion of the pixels may be similar to an image that the NN has already seen and trained on. As a result, the NN is likely to output a prediction that is similar to the previously seen image. Again, we could perform this similarity comparison manually, but that would require much more work since the NN already performs it for us.

OTHER 'TRICKS 'TO IMPROVE THE PERFORMANCE OF OUR AGENT

There are a few things we can do to speed up performance in general, reduce computation time, and run through more episodes in a given period of "wall-clock time."

Preprocessing the image

Image cropping — as you can see from the Pong screenshots there are a lot of redundant areas which provide no value for our algorithm, for example the score section, and the white bar underneath the score. This has the added benefit of allowing us to ignore parts of the frame where the game result has already been decided, i.e., after the ball has past the paddle already.

Diagram illustrating the case for ignoring / cropping the image section after the ball passes the paddle.

Gray-scaling — the color of the pixels is irrelevant so we can afford to remove them, for both the background and paddle colors. The paddles and ball can be the same color, but should be a different color from the background.

Downsampling / reducing resolution — lets us care about less of the image, and reduces the memory and computation requirements.

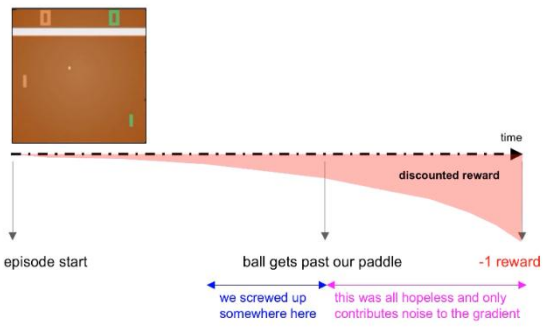


Fig -5: Representation of Optimized Performance

5. RESULTS AND DISCUSSION

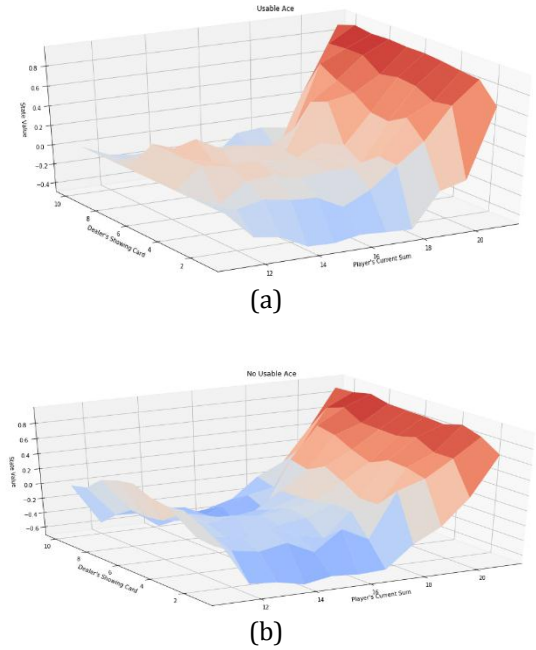
It was noted that our bot found it simple to pick up on the many games that were given to it. For each game, the output generally stayed relatively low until a certain

point, at which point the bot began to genuinely learn and develop its own strategy.

Additionally, it was noted that our bot adopted several tactics throughout the training sessions. One of the most crucial aspects of reinforcement learning is that it prevents players from picking up our bot's techniques.

As a result, they can play with the same bot repeatedly without feeling as though it is getting easier for them. Graphs have been used to depict the outcomes of the bot playing the different games.

5.1 RESULT OF BLACKJACK PLAYED USING MONTE CARLO CONTROL:



(a)

(b)

Fig -6: Graph State using Monte Carlo

Based on the above graphs, out bot devises a strategy for one training round that can be represented by the below graph:

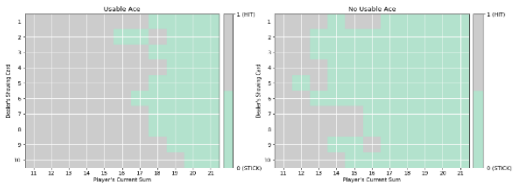


Fig -7: Blackjack Predictions

The above figure shows the strategy devised by our bot. Our policy-based model performs better than the conventional 80-20. as well as a random policy.

This is significant improvement from baselines.

5.2 RESULTS OF ATARI GAMES PLAYED USING ACTOR CRITIC ALGORITHM

This is the performance graph obtained on one particular training of bot while playing PONG.

The results vary with every training that is carried out.

Fig -8: Performance using Actor-Critic Learning

The reward (average of 100 episodes) is plotted along with the number of episodes and as we can see that the performance of our bot has increased significantly as the training progressed.

6. CONCLUSION

Reinforcement learning is used by the AI bot to develop its skills and advance in the game. RL is most suited for creating complicated bot behavior within a game when compared to other machine learning techniques. The learning bot is engaging with its surroundings and getting feedback depending on its activities, just like a human player would.

For our first card game, blackjack, which has a smaller state space, we applied Q learning and Monte Carlo Control.

But as we shifted to playing Atari games, the state space and complexity of our games grew, thus we had to train our bot with more complex algorithms like the Actor Critic Algorithm.

In fact, our bot does not require any training data at all and does not rely on any initial training data. As it advances in skill with each repetition, it learns by playing the game itself and develops its own tactics.

The project's outcome exceeded my expectations because my bot was able to pick up all the games quickly and, after a set number of training iterations, was ready to compete

against players of average skill. If properly taught over an extended period of time, this bot may even be able to defeat the greatest players.

I intend to integrate Amazon Alexa into my work in the future and add the possibility for learners to play games as a feature. If this project is successful, Alexa's popularity will rise considerably. It will be directed at those residing in nursing homes and assist in addressing the long-ignored issue of depression prevalent among older age groups.

ACKNOWLEDGEMENT

We would like to thank Dr. S.B. Vanjale of Bharati Vidyapeeth Deemed to be University for his ongoing advice, support, and understanding. More than anything, he taught me to be patient in my work. My relationship with him extends beyond academia, and I am really fortunate to have the chance to collaborate with a thinker and ML and AI expert.

We would like to convey our gratitude to all of the instructors for creating a productive environment and for serving as an inspiration throughout the duration of the course.

We convey imaginatively our sincere gratitude to our academic staff and the people who serve as the foundation of our university for their non-self-centered excitement and timely encouragements, which inspired me to gain the necessary information to successfully complete my course study. We want to express our gratitude to my parents for their help.

It is a joy to express our gratitude to our friends for their encouragement and persuasion to take on and complete this assignment. Finally, we would like to offer our appreciation and gratitude to everyone who has contributed, directly or indirectly, to the successful completion of this project.

REFERENCES

[1] Robin, Baumgarten & Simon, Colton & Mark, Morris. (2018). Combining AI Methods for Learning Bots in a Real-Time Strategy Game. International Journal of Computer Games Technology.

[2] Azizul Haque Ananto .(2017). Self-Learning Game Bot using Deep Reinforcement Learning

[3] Volodymyr Mnih Koray Kavukcuoglu David Silver (2018). Playing Atari with Deep Reinforcement Learning

[4] Vincent-Pierre Berges(2018). Reinforcement Learning for Atari Breakout, Stanford University

[5] Leemon Baird. Residual algorithms: Reinforcement learning with function approximationrgan Kaufmann, 1995.

[6] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research

[7] Marc G Bellemare, Joel Veness, and Michael Bowling. Investigating contingency awareness using atari 2600 games, 2012.

[8] Marc G. Bellemare, Joel Veness, and Michael Bowling. Bayesian learning of recursively factored environments, 2018.

[9] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. January 2012.

[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks, 2015.

[11] Matthew Hausknecht, Risto Miikkulainen, and Peter Stone. A neuro-evolution approach to general atari game playing. 2018.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. 2012.