1. Introduction

The Ultimate Tic-Tac-Toe (UTTT) is a variant of the Tic-Tac-Toe game that is more difficult and enlarged. UTTT has nine small Tic-Tac-Toe grids that are placed in a bigger 3x3 format in place of one 3x3 board, which creates a playable environment of 9x9. The most important twist here is that the movement of the player would dictate the small board on which the opponent is required to make the next move. This rule gives a lot of strategic depth to the game because the players not only have to pay attention to the local strategies on a small board but also on the global positions on the large board.

The state space of regular Tic-Tac-Toe is small (approximately $\mathbf{3^9 \approx}$ 19,683 states) and can thus be solved either by brute force or by simple rules and strategies. Conversely, the state space of UTTT is significantly larger than 1012: the local board states are complicated, and the global board interactions are also complicated. Also, the forced-move mechanic establishes interdependencies between moves making it difficult to employ simple look-ahead strategies.

Due to this complication, Reinforcement Learning (RL) is very appropriate in UTTT. There are no strategies that RL presupposes or human demonstrations. Rather, an agent is exposed to the game world and learns through trial and error and slowly becomes better through experience of the consequences of its deeds. Rarely can the optimal strategy be apparent, and long-term strategy is also required in the RL methods, which is the reason why they are a good fit with the dynamic decision-making process in UTTT.

2. Methodology

2.1 State Representation

The state of the game is represented in the form of a 9x9 grid with each square possibly having no objects, an X, or an O. In order to grasp the rule that only allows the next move to stay on a particular small board, there is the inclusion of an active board indicator. This indicator is one of the 9 small boards (0 to 8). In case the small board needed is taken or already occupied, the agent has the option to use any blank cell that is free and indeed this is reflected by the indicator.

State representation includes:

- A 9×9 matrix representing all cell states
- A single integer representing the active small board

This is to make both the local and global game contexts reflected and processed by the learning model.

2.2 Action Space

The number of potential moves in the total action space is 81, and each of them is associated with a cell on the board. Nevertheless, there are only subsets of legal actions at any given turn. Illegal moves include:

- Moves in the wrong small board
- Moves in already-occupied cells

- Moves in completed or full boards when a playable board is still required

Thus, legal move screening is carried out. The RL agent is provided with a set of valid moves before the agent makes a decision, so it learns efficiently and does not have to perform irrelevant updates as a result of no-go actions.

2.3 Reward Structure

The reward system guides the agent toward optimal gameplay. It includes:

- **+1** for winning the game
- **–1** for losing the game
- **0** for a draw
- **Small negative penalty** (e.g., –0.05) for illegal move attempts
- Optional small rewards for winning a small board

This structure encourages strategic thinking while discouraging illegal or careless moves.

2.4 Algorithm: Q-learning or SARSA

It is based on the learning process that takes either Q-learning or SARSA (function approximation model which can be a simple neural network). Understanding Q-values of all possible states is impractical since the state space is enormous, and so, a function approximator is used to estimate the value of each state-action pair.

- Q-learning relies on a maximum future reward to update the Q-values and hence it is more aggressive.
- SARSA uses a more realistic version of the next action to be followed which generates more consistent yet occasionally slower learning.

Exploration is managed by an **ε-greedy policy** which picks random moves with probability ε and exploits learned values in other cases where it progressively reduces ε with time to transition between learning and exploiting.

Exploration is handled using an **ε-greedy policy**, which selects random moves with probability ε and exploits learned values otherwise. ε decreases over time to shift from learning to exploiting.

2.5 Training Approach: Self-Play

The agent is trained using self-play in that it competes against itself repeatedly. This method does not need human data on playing the game and assists the agent to experience a great amount of strategies. With the evolution of training, the agent automatically has tougher competition, its own advanced versions, which motivates continuous enhancement.

Modern reinforcement learning Self-play is popular in complex games due to the fact that it provides:

- Equivalent exposure to various strategic choices.
- Gradual increase of the difficulty.
- No necessity of professional performances.

3. Findings

3.1 Learning Curve and Convergence

There are obvious steps in development of the agent which are observed in the learning curve:

1. **Early stage**: The agent is played almost randomly. The illegal actions are common, and the play is soon terminated because of simple errors.
2. **Middle stage**: The agent slowly decreases the number of illegal moves and starts to see patterns in small boards.
3. **Later stage**: The agent is aware of the impact of local actions in global strategy. Q-values are stabilized and it means that there is some convergence.

The large state space can be extremely expensive to train, particularly requiring thousands of training episodes to completely converge. Learning curve generally starts with enormous variations but gets gently with a rise in the experience of the agent.

3.2 Win Rate Against Random and Rule-Based Opponents

The trained RL agent will eventually win much more frequently, 70 to 90 percent, on average, when playing their randomly selected opponent. It proves that the agent already has mastered the ability to find the beneficial positions and prevent elementary errors.

The RL agent is less likely to win, typically with a win rate of 40 to 60 percent, against a rule-based player that follows the simplest heuristics (such as holding positions in the center or preventing early threats). This implies that as the agent acquires successful strategies, more advanced agents take advantage of the vulnerabilities of the agent to its partial long-term planning capacity without further training.

3.3 Strategic Insights Learned by the Agent

The agent eventually becomes aware of a number of strategies that are meaningful:

1. **Center Control**: This agent is often choosing center cells and boards, which are providing a greater influence on future movements.
2. **Forcing the Opponent**: The agent trains to make moves which place the opponent in a hard or disadvantageous small boards.
3. **Macro-Level Planning**: The agent does not just look at how to win small boards, but instead intends to achieve the macro victory by winning three boards aligned.
4. **Blocking Threats**: This is a type of agent that becomes more efficient to block the opponent when they are about to play two-in-a-row.

These plans are automatically obtained during RL, without any programming.

3.4 Limitations and Challenges

Despite positive results, several limitations emerged:

1. **Large State Space**
   UTTT's huge number of possible states makes learning slow and resource-intensive.
2. **Sparse Rewards**
   Rewards are typically given only at the end of the game, which makes it hard for the agent to trace which specific moves caused success or failure.
3. **Balancing Exploration and Exploitation**
   Choosing the right ε-greedy decay schedule is difficult. Too much randomness slows progress; too little limits learning.
4. **High Computational Demand**
   Effective training requires many self-play episodes, which increases processing time.
5. **Performance vs. Skilled Opponents**
   Although the agent can beat random opponents, it will still not be as effective against players who play based on rules, unless its training lasts a very long time.

4. Discussion

This project shows that reinforcement learning can effectively be implemented to a complicated and strategy intensive game such as Ultimate Tic-Tac-Toe. The agent is able to progressively change a random behavior to an intelligent strategic play through self-play and Q-learning or SARSA. The progression of the agent resembles that of a human learning curve: the agent begins with simple knowledge of boards, then becomes more tactical in their work and finally learns some basics of the long-term plan.

One of the most intriguing facts is that the agent acquires local (winning minor boards) and global strategies (managing the board layout, in general). This two-layer decision-making is crucial in UTTT and indicates that RL is able to conform to complex hierarchical issues.

In other ones, the RL agent finds non-standard actions that human beings seldom take but they can be beneficial. That implies that the RL approach does not just mimic human reasoning but it tries to search through the entire strategy space it can.

All in all, the project shows strengths and difficulties of using reinforcement learning in large and complex board games. Although the agent can learn meaningful strategies and can learn to perform better over time, more complicated algorithms, e.g., deep reinforcement learning or Monte Carlo tree search, are needed to master it.

Conclusion

Overall, this project illustrates that Reinforcement Learning can be useful in learning to play Ultimate Tic-Tac-Toe even though the state space of the game is large and the strategy is complicated. The agent enhanced, decreased errors, and created meaningful strategies such as center control and forced-board play through self-play, value-based techniques such as Q-learning or SARSA. Although the agent is competent with random opponents, higher-level techniques would be required to achieve the expertise level of play. In general, the paper reflects the possibilities of RL in the context of complex board games and creates a solid base upon which new developments can be built.

References

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). *Human-level control through deep reinforcement learning*. **Nature, 518**(7540), 529–533.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Shannon, C. E. (1950). *Programming a computer for playing chess*. **Philosophical Magazine, 41**(314), 256–275.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., … Hassabis, D. (2016). *Mastering the game of Go with deep neural networks and tree search*. **Nature, 529**(7587), 484–489.

Wikipedia contributors. (2023). *Ultimate tic-tac-toe*. In **Wikipedia, The Free Encyclopedia**. Retrieved from https://en.wikipedia.org/wiki/Ultimate_tic-tac-toe