

Reinforcement Learning for Connect Four



**Team: Khushi Dave, Mohamed Abdrabbou And Sajal Kumar**

**Contents**

[What is reinforcement learning and its type 2](#_Toc77802859)

[Neural Networks 2](#_Toc77802860)

[Structure of the equation for neural network: 2](#_Toc77802861)

[Propagation function 3](#_Toc77802862)

[Bias 3](#_Toc77802863)

[Connect Four Game 3](#_Toc77802864)

[Basic Rules of Connect Four 3](#_Toc77802865)

[Q Learning Algorithm 4](#_Toc77802866)

[Bellman Optimality Equation 4](#_Toc77802867)

[Epsilon Greedy Strategy 5](#_Toc77802868)

[Update Q table data 5](#_Toc77802869)

[Introduction to Minimax 5](#_Toc77802870)

[Minimax algorithm 5](#_Toc77802871)

[-Strategy 5](#_Toc77802872)

[Pseudo-code for MiniMax Algorithm: 6](#_Toc77802873)

[Working of Min-Max Algorithm: 6](#_Toc77802874)

[That was the finished work process of the minimax two player game. 9](#_Toc77802875)

[Properties of Mini-Max algorithm: 9](#_Toc77802876)

[Limitation of the minimax Algorithm: 9](#_Toc77802877)

[**Bibliography** 10](#_Toc77802878)

[Figure 1 Q\* Function to calculate the state action pair given the optimal path 4](#_Toc77802879)

[Figure 2 Bellman Optimality Equation for calculating the optimal path 4](#_Toc77802880)

[Figure 3 Q equation to update values in the Q table 5](#_Toc77802881)

[Figure 4 Minimax algorithm in tree format — the initial step 7](#_Toc77802882)

[Figure 5 Minimax algorithm in tree format — the second step 8](#_Toc77802883)

[Figure 6 Minimax algorithm in tree format — the third step 8](#_Toc77802884)

[Figure 7 Minimax algorithm in tree format — the final step 9](#_Toc77802885)

|  |
| --- |
| What is reinforcement learning and its type Reinforcement learning is the science of making appeasement decisions using experience. This is an important part of machine learning generally using in-game playing. It is about taking appropriate steps to maximize rewards in a particular situation. It is employed by specific software and machines to find the best possible behavior or path in a particular situation. Reinforcement learning varies from that in supervised learning in a way that in supervised learning the training data has the solution with it so the model is informed with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent settles upon what to do to perform the given task. In the absence of a training dataset, it is forced to learn from its experience [1]. |

|  |
| --- |
| Neural Networks |

The Neural Networks are function approximations, which are especially obligate in reinforcement learning when the state space or action space are immoderate to be perfectly known. A Neural Network can be used to constructive a value function, or a policy function. That is, neural nets can become acquainted with map states to values, or state-action pairs to Q values. Rather than conduct a lookup table to store, exponent and update all possible conditions and their values, which is impossible with very large problems, we can a train neural network on samples from the state or action space to learn to portend how valuable those are relative to our objective in reinforcement learning. Like all Neural Networks, they use coefficients to presumptive the function relating inputs to outputs, and this is a search for the right coefficients in their learning or weights, by iteratively adjusting those weights toward gradients that assurance less error [2].

# Structure of the equation for neural network:

A neuron with label receiving an input from predecessor neurons consists of the following components.

* an activation ,the neuron’s state, depending on a discrete time parameter.
* an optional threshold , which stays fixed unless changed by learning.
* an activation function that computes the new activation at a given time from and the net input giving rise to the relation

,

and an output function computing the output from all activation

.

Often the output function is simply the identity function. An input neuron has no predecessor but serves as input interface for the whole network. The output neuron serves as output interface of the whole network [3].

# Propagation function

The propagation function has the form

Through it can use the output outputs to calculate the

# Bias

A bias term can be added, changing the form to the following [3]

, where is a bias.

# Connect Four Game

Connect Four is a famous two-player technique game. Each player has their own colored discs and players alternate in electing a slot on the board to spill the disc in with the target of find four of their colored discs in a row, either horizontal, vertical, or diagonal. The game’s roots start from Tic-Tac-Toe, a pen and paper game with the destination of getting three in a row, as converse to four. The game first became famous under the Milton Bradley company after it was first coined Connect Four in 1974.

Connect Four is a compelling game as although it looks easy and straightforward, there is a momentous opportunity to demeanor strategy to raise and even guarantee one's likelihood of winning. For example, targeting earmarked slots over others is important. The slots in the middle of the board are more expensive than the ones on the edges because there is an upper chance of building four in a row. The board and regulations of the game qualify a more variety of feasible final boards with players able to take numerous various actions to reaching them. We aimed to use reinforcement learning to get the optimal policies for the Connect Four Markov Decision Process.

# Basic Rules of Connect Four

Connect Four is played on a perpendicular board with six rows and seven columns which totals 42 playable slots on the board. At the top vertical edge of the board, for each column, there is incise where the discs are slotted into. Once the disc has been slotted into a column it goes down the column to the lowest row or it stands the row above the disc that was last played in that column. It is a two-player game and each player must have twenty-one disc-like pieces in various colors to the other player.

The game begins by randomly ascertaining whether player one or player two or player one accepts the first turn. After the first player takes their turn, turns are then taken alternatively by the two players. A turn form of a player spilling their colored piece into a column on the board. Each turn a player has to spill their colored disc into a column, they must not ‘miss their turn.’ If a column is full (six pieces in the column), then that column cannot play into the piece. Additionally, once the disc has been played, it cannot be undone or took from the board. The aim of Connect Four game pieces in a row (either vertically, horizontally, or diagonally) without any gaps on the board between the four discs. And this is the end of this game and the player who completed the four pieces’ wins. Otherwise, all the discs are played, the outcome is the board is full, then the game is called a tie [4].

# Q Learning Algorithm

Q-learning is a reinforcement learning algorithm that doesn’t depend on the modeling of the environment. Q- learning model-free nature enables it to operate under stochastic conditions without adaption relying mainly on rewards. It finds an optimal policy in which the expected return through overall successive time steps is the maximum achievable. Once calculated, the optimal policy/path (Q function) enables the algorithm to determine certain actions that maximize Q\* for each state-action pair[5].

Q\*-Function: returns the expected output of an action given the current state with the maximum reward achievable.



Figure Q\* Function to calculate the state action pair given the optimal path

s = a particular state

a = an action adding a coin)

π = optimal policy

Also, the Q\* function has to satisfy the bellman Optimality equation.

# Bellman Optimality Equation

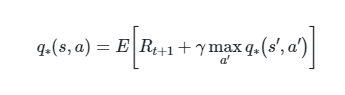
****

Figure Bellman Optimality Equation for calculating the optimal path

The bellman optimality equation states that for any state action pair (s, a) at any time t, the expected output from these arguments under the optimal policy equals the expected reward (R+1) added to the maximum possible discounted output from any following state action pair (s’, a’)[5]

The Q table represents all the possible state-action pairs. Columns will show actions and rows will show states. It starts with zero but after playing several times the values in the q table will be updated.

# Epsilon Greedy Strategy

We introduce two concepts: exploring the environment to get more information and then exploit this information to maximize the return. The agent has to utilize both methods efficiently. In the beginning, the agent will explore. As it builds knowledge about the game, it will exploit it.

To control this tradeoff, we introduce the epsilon greedy strategy. First, we define the exploration rate, epsilon, set to be 100% at the beginning. Epsilon is a probability explaining the likelihood of exploring the environment instead of exploiting it. During the first episode, it will explore the environment randomly. Then, the agent acquires more information about the environment every round. We set a factor, for epsilon to decay with, to reflect this learning potential and give less weight to exploration. Then we introduce a randomly generated number between 0 and 1 and compare it to epsilon, if the number is larger, the agent disregards exploration and chooses the highest q-value action in its current state from the Q table through exploitation[6].

# Update Q table data

First we introduce the learning rate (between 0 and 1) which represents the likelihood of the agent to disregard the old Q value and acquire a new Q value. The new value will be calculated as a weighted sum between the old and learned value as follows[6]:

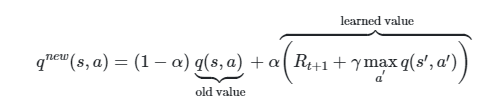
****

Figure Q equation to update values in the Q table

# Introduction to Minimax

Minimax is a choice guideline basically utilized in Artificial Intelligence, choice hypothesis, game hypothesis, measurements, and theory for limiting the conceivable misfortune for a direst outcome imaginable. When managing gains, it is alluded to as 'maximin'- to boost the base addition. Initially figured for n-player lose-lose situation hypothesis, covering both the situations where players take substitute moves and where synchronous moves, it has likewise been reached out to more mind boggling games and to general dynamic within the sight of vulnerability [7].

# Minimax algorithm

## -Strategy

* Minimax algorithm uses backtracking to seek answers in the game hierarchy.
* Minimax calculation is, for the most part utilized for game playing in AI. Such as Chess, Checkers, Tic-tac-toe, go and various two-player games. This calculation registers the minimax choice for the present status.
* In minimax algorithm when two players play the game, one is called MAX and other is called MIN.
* Both the players battle it as the adversary player gets the base advantage while they get the greatest advantage.
* The minimax calculation plays out a profundity first quest calculation for the investigation of the total game tree.
* The minimax calculation continues right down to the terminal hub of the tree, then, at that point backtrack the tree as the recursion.

## Pseudo-code for MiniMax Algorithm:

function minimax(position, depth, alpha, beta, maximizingPlayer) is

if depth ==0 or game over in position then

return static evaluation of position

if Maximizing Player then, at that point/for Maximizer Player

maxeval= -infinity

 for each child of position

 eval= minimax (child, depth-1, false)

maxeval= max (maxeval, eval)/gives Maximum of the qualities

return maxeval

else                         // for Minimizer player

 mineval= +infinity

 for each child of position do

 eval= minimax (child, depth-1, true)

mineval= min (mineval, eval)/gives least of the qualities

**return** mineval

## Working of Min-Max Algorithm:

* The working of the minimax calculation can be handily portrayed utilizing a model. Underneath we have taken an illustration of game-tree which is addressing the two-player game.
* In this model, there are two players one is called Maximizer and other is called Minimizer.
* Maximizer will attempt to get the Maximum conceivable score, and Minimizer will attempt to get the base conceivable score.

Set the game tree:

**Step-1:** In the underlying advance, the estimation delivers the entire game-tree and apply the utility ability to get the utility characteristics for the terminal states. In the underneath tree chart, how about we take an is the underlying condition of the tree. Assume maximizer takes first turn which has most pessimistic scenario beginning worth =-vastness, and minimizer will take next turn which has most pessimistic scenario introductory worth = +infinity.



Figure 4 Minimax algorithm in tree format — the initial step

**Step 2:** Presently, first we discover the utilities an incentive for the Maximizer, its underlying worth is - ∞, so we will contrast each worth in terminal state and beginning worth of Maximizer and decides the higher hubs esteems. It will track down the greatest among the all.

* Node D: max (-1, - -∞) => max (-1,4) equals 4
* Node E: max (2, -∞) => max (2, 6) equals 6
* Node F: max (-3, -∞) => max (-3, -5) equals -3
* Node G: max (zero, -∞) = max (zero, 7) equals 7



Figure 5 Minimax algorithm in tree format — the second step

**Step 3:**  Minimizer will contrast all hubs esteem and +∞, and will track down the third layer hub esteems.

* Node B: min (4,6) equals 4
* Node C: min (-3, 7) equals -3



Figure 6 Minimax algorithm in tree format — the third step

**Step 4:** Presently it's a turn for Maximizer, and it will again pick the limit of all hubs worth and track down the most extreme incentive for the root hub. In this game tree, there are just 4 layers, consequently we reach quickly to the root hub, yet in genuine games, there will be multiple layers.

* Node A: max (4, -3) equals 4



Figure 7 Minimax algorithm in tree format — the final step

## That was the finished work process of the minimax two player game.

## Properties of Mini-Max algorithm:

* **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if it exists), in the finite search tree.
* **Optimal-** Min-Max algorithm can be optimal if both opponents are playing optimally.
* **Space Complexity-** Space intricacy of Mini-max calculation is likewise like DFS which is O(bm).

## Limitation of the minimax Algorithm:

* The fundamental disadvantage of the minimax calculation is that it gets truly delayed for complex games like Chess, go, and so on.
* This kind of games has an enormous expanding factor, and the player has heaps of decisions to choose.
* This constraint of the minimax calculation is enhanced through alpha-beta pruning [7].

# **Bibliography**

[1] “Reinforcement learning - GeeksforGeeks.” [Online]. Available: https://www.geeksforgeeks.org/what-is-reinforcement-learning/. [Accessed: 14-Jul-2021].

[2] “A Beginner’s Guide to Deep Reinforcement Learning | Pathmind.” [Online]. Available: https://wiki.pathmind.com/deep-reinforcement-learning. [Accessed: 19-Jul-2021].

[3] S. W. ELLACOTT, J. C. MASON, and lain J. ANDERSON, “Mathematics of Neural Networks,” *Springer Sci. Media New York*, vol. 8, no. 0704, 1997.

[4] E. Alderton, E. Wopat, and J. Koffman, “Reinforcement Learning for Connect Four,” 1974.

[5] “Q-Learning Explained - A Reinforcement Learning Technique - deeplizard.” [Online]. Available: https://deeplizard.com/learn/video/qhRNvCVVJaA. [Accessed: 13-Jul-2021].

[6] “Exploration vs. Exploitation - Learning the Optimal Reinforcement Learning Policy - deeplizard.” [Online]. Available: https://deeplizard.com/learn/video/mo96Nqlo1L8. [Accessed: 13-Jul-2021].

[7] “Artificial Intelligence | Mini-Max Algorithm - Javatpoint.” [Online]. Available: https://www.javatpoint.com/mini-max-algorithm-in-ai. [Accessed: 20-Jul-2021].