



[PYTHON \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/\)](https://www.analyticsvidhya.com/blog/category/python-2/)

[REINFORCEMENT LEARNING \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/MACHINE-LEARNING/REINFORCEMENT-LEARNING/\)](https://www.analyticsvidhya.com/blog/category/machine-learning/reinforcement-learning/)

Introduction to Monte Carlo Tree Search: The Game-Changing Algorithm behind DeepMind's AlphaGo

[ANKIT CHOUDHARY \(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/ANKIT2106/\)](https://www.analyticsvidhya.com/blog/author/ankit2106/), JANUARY 24, 2019
 [LOGIN TO BOOKMARK THIS ARTICLE \(HTTPS://ID.ANALYTICSVIDHYA.COM/ACCOUNT-\)](https://www.analyticsvidhya.com/account/)

Introduction

A best of five game series, \$1 million dollars in prize money – A high stakes shootout. Between 9 and 15 March, 2016, the second-highest ranked Go player, Lee Sidol, took on a computer program named AlphaGo.

AlphaGo emphatically outplayed and outclassed Mr. Sidol and won the series 4-1. Designed by Google's DeepMind, the program has spawned many other developments in AI, including AlphaGo Zero. These breakthroughs are widely considered as stepping stones towards Artificial General Intelligence (AGI).



In this article, I will introduce you to the algorithm at the heart of AlphaGo – Monte Carlo Tree Search (MCTS). This algorithm has one main purpose – given the state of a game, choose the most promising move.

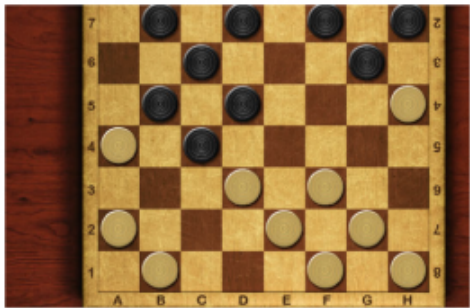
To give you some context behind AlphaGo, we'll first briefly look at the history of game playing AI programs. Then, we'll see the components of AlphaGo, the Game Tree Concept, a few tree search algorithm, and finally dive into how the MCTS algorithm works.

Table of Contents

- 1. Game AI – A Summary
- 2. Components of the AlphaGo Program
- 3. The Game Tree Concept
- 4. Tree Search Algorithms
 - 1. Uninformed Search
 - 2. Best First Search
 - 3. Minimax
- 5. Monte Carlo Tree Search
 - 1. Tree Traversal and Node Expansion
 - 1. UCB1 (Upper Confidence Bound)
 - 2. Rollout
 - 2. Complete Walkthrough with an example

Game AI – A Summary

AI is a vast and complex field. But before AI officially became a recognized body of work, early pioneers in computer science wrote game-playing programs to test whether computers could solve human-intelligence level tasks.



Checkers



Chess



Your Ultimate path to Becoming a DATA Scientist!

To give you a sense of where Game Playing AI started from and it's journey till date, I have put together the below key historical developments:

- 1. A. S. Douglas programmed the **first software that managed to master a game in 1952**. The game? Tic-Tac-Toe! This was part of his doctoral dissertation at Cambridge
- 2. **A few years later, Arthur Samuel was the first to use reinforcement learning that to play Checkers by playing against itself**
- 3. In 1992, Gerald Tesauro designed a now-popular **program called TD-Gammon to play backgammon at a world-class level**
- 4. For decades, Chess was seen as "the ultimate challenge of AI". **IBM's Deep Blue was the first software that exhibited superhuman Chess capability**. The system famously defeated Garry Kasparov, the reigning grandmaster of chess, in 1997
- 5. **One of the most popular board game AI milestones was reached in 2016 in the game of Go**. Lee Sedol, a 9-dan professional Go player, lost a five-game match against Google DeepMind's AlphaGo software which

Name

Email

Contact Number

Download Resources

featured a deep reinforcement learning approach

6. Notable recent milestones in video game AI include **algorithms developed by Google DeepMind to play several games from the classic Atari 2600** video game console at a super-human skill level
7. Last year, OpenAI built the popular **OpenAI Five** (<https://www.analyticsvidhya.com/blog/2018/06/openai-five-a-team-of-5-algorithms-is-beating-human-opponents-in-a-popular-game/>) system that mastered the complex strategy game of DOTA

And this is just skimming the surface! There are plenty of other examples where AI programs exceeded expectations. But this should give you a fair idea of where we stand today.

Components of the AlphaGo

The core parts of the Alpha Go comprise of:

- **Monte Carlo Tree Search:** AI chooses its next move using MCTS
- **Residual CNNs (Convolutional Neural Networks):** AI assesses new positions using these networks
- **Reinforcement learning:** Trains the AI by using the current best agent to play against itself

In this blog, we will **focus on the working of Monte Carlo Tree Search** only. This helps AlphaGo and AlphaGo Zero smartly explore and reach interesting/good states in a finite time period which in turn helps the AI reach human level performance.

It's application extends beyond games. MCTS can theoretically be applied to any domain that can be described in terms of *[state, action]* pairs and simulation used to forecast outcomes. Don't worry if this sounds too complex right now, we'll break down all these concepts in this article.

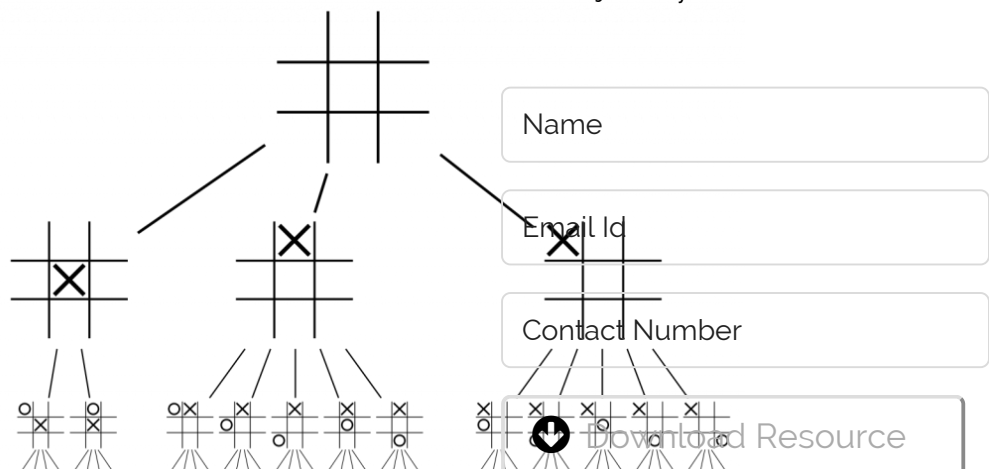
The Game Tree Concept

Game Trees are the most well known data structures that can represent a game. This concept is actually pretty straightforward.

Each node of a game tree represents a particular state in a game. On performing a move, one makes a transition from a node to its children. **The nomenclature is very similar to decision trees wherein the terminal nodes are called leaf nodes.**

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.



For example, in the above tree, each move is equivalent to putting a cross at different positions. This branches into various other states where a zero is put at each position to generate new states. This process goes on until the leaf node is reached where the win-loss result becomes clear.

Tree Search Algorithms

Our primary objective behind designing these algorithms is to find best the path to follow in order to win the game. In other words, look/search for a way of traversing the tree that finds the best nodes to achieve victory.

The majority of AI problems can be cast as search problems, which can be solved by finding the best plan, path, model or function.

Tree search algorithms can be seen as building a search tree:

- The root is the node representing the state where the search starts
- Edges represent actions that the agent takes to go from one state to another
- Nodes represent states

The tree branches out because there are typically several different actions that can be taken in a given state. Tree search algorithms differ depending on which branches are explored and in what order.

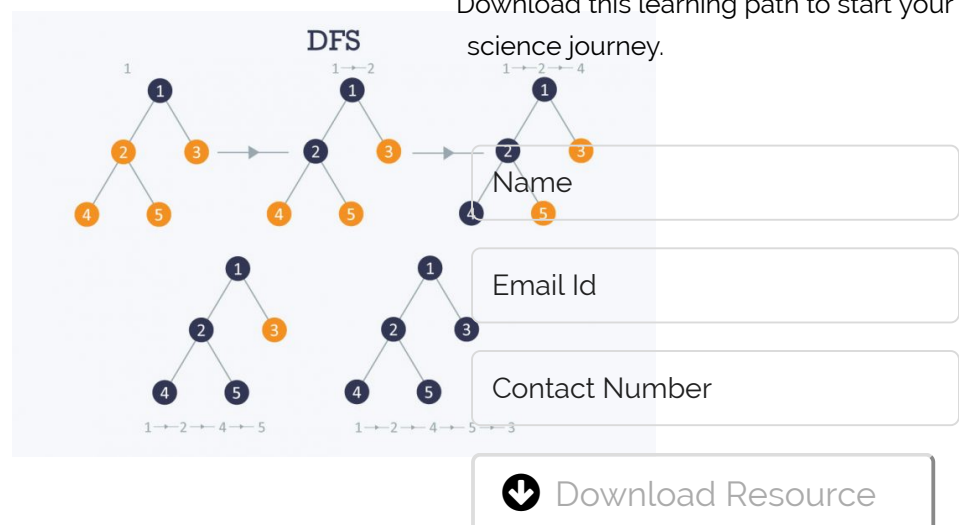
Let's discuss a few tree search algorithms.

Uninformed Search

Uninformed Search algorithms, as the name suggests, search a state space without any further information about the goal. These are considered basic computer science algorithms rather than as a part of AI. Two basic algorithms that fall under this type of search are **depth first search (DFS)** and **breadth first search (BFS)**. You can read more about them in this [blog post \(https://www.analyticsvidhya.com/blog/2021/04/your-ultimate-path-for-becoming-a-network-analysis-python-codes/\)](https://www.analyticsvidhya.com/blog/2021/04/your-ultimate-path-for-becoming-a-network-analysis-python-codes/).

DATA Scientist!

Download this learning path to start your data science journey.



Best First Search

The Best First Search (BFS) method explores a graph by expanding the most promising node chosen according to a specific rule. The defining characteristic of this search is that, unlike DFS or BFS (which blindly examine/expand a cell without knowing anything about it), BFS uses an evaluation function (sometimes called a "heuristic") to determine which node is the most promising, and then examines this node.

For example, A* algorithm keeps a list of "open" nodes which are next to an explored node. Note that these open nodes have not been explored. For each open node, an estimate of its distance from the goal is made. New nodes are chosen to explore based on the lowest cost basis, where the cost is the distance from the origin node plus the estimate of the distance to the goal.

Minimax

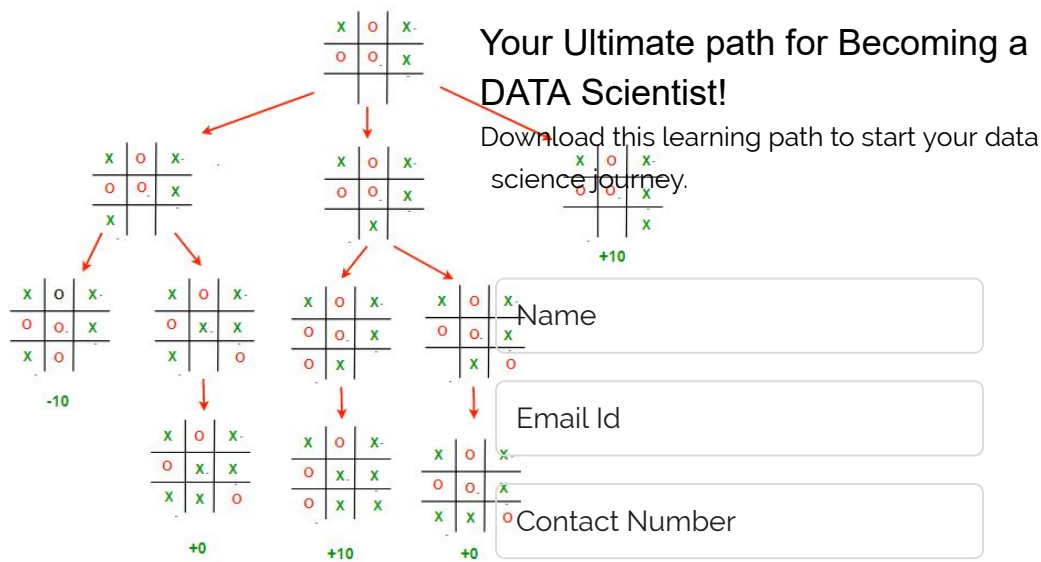
For single-player games, simple uninformed or informed search algorithms can be used to find a path to the optimal game state. What should we do for two-player adversarial games where there is another player to account for? The actions of both players depend on each other.

For these games, we rely on adversarial search. This includes the actions of two (or more) adversarial players. **The basic adversarial search algorithm is called Minimax.**

This algorithm has been used very successfully for playing classic perfect-information two-player board games such as Checkers and Chess. In fact, it was (re)invented specifically for the purpose of building a chess-playing program.

The core loop of the Minimax algorithm alternates between player 1 and player 2, quite like the white and black players in chess. These are called the min player and the max player. All possible moves are explored for each player.

For each resulting state, all possible moves by the other player are also explored. This goes on until all possible move combinations have been tried out to the point where the game ends (with a win, loss or draw). The entire game tree is generated through this process, from the root node down to the leaves:



Each node is explored to find the moves that give us the maximum value or score.

Monte Carlo Tree Search

Games like tic-tac-toe, checkers and chess can arguably be solved using the minimax algorithm. However, things can get a little tricky when there are a large number of potential actions to be taken at each state. This is because minimax explores all the nodes available. It can become frighteningly difficult to solve a complex game like Go in a finite amount of time.

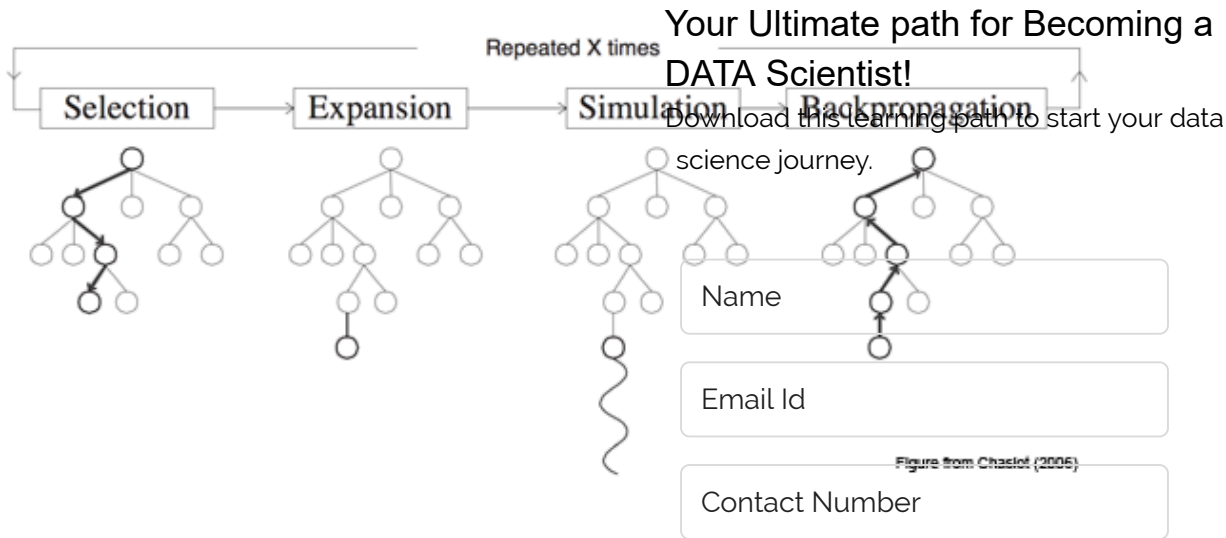
Go has a branching factor of approximately 300 i.e. from each state there are around 300 actions possible, whereas chess typically has around 30 actions to choose from. Further, the positional nature of Go, which is all about surrounding the adversary, makes it very hard to correctly estimate the value of a given board state. For more information on rules for Go, please refer this [link \(https://en.wikipedia.org/wiki/Rules_of_Go\)](https://en.wikipedia.org/wiki/Rules_of_Go).



Image: Lee Jin-man/ AP

There are several other games with complex rules that minimax is ill-equipped to solve. These include Battleship, Poker with imperfect information and non-deterministic games such as Backgammon and Monopoly. Monte Carlo Tree Search, invented in 2007, provides a possible solution.

The basic MCTS algorithm is simple: a search tree is built, node-by-node, according to the outcomes of simulated playouts. The process can be broken down into the following steps:



1. Selection

Selecting good child nodes, starting from the root node R, that represent states leading to better overall outcome (win).

2. Expansion

If L is not a terminal node (i.e. it does not end the game), then create one or more child nodes and select one (C).

3. Simulation (rollout)

Run a simulated playout from C until a result is achieved.

4. Backpropagation

Update the current move sequence with the simulation result.

Tree Traversal & Node Expansion

Before we delve deeper and understand tree traversal and node expansion, let's get familiar with a few terms.

UCB Value

UCB1, or upper confidence bound for a node, is given by the following formula:

$$UCB1 = V_i + 2 \sqrt{\frac{\ln N}{n_i}}$$

where,

- V_i is the average reward/value of all nodes beneath this node
- N is the number of times the parent node has been visited, and
- n_i is the number of times the child node i has been visited

Rollout

What do we mean by a rollout? Until we reach the leaf node, we randomly choose an action at each step and simulate this action to receive an average reward when the game is over.

**Your Ultimate path for Becoming a
DATA Scientist!**

Loop Forever:

if S_i is a terminal state:

 return Value(S_i)

A_i = random(available_actions(S_i))

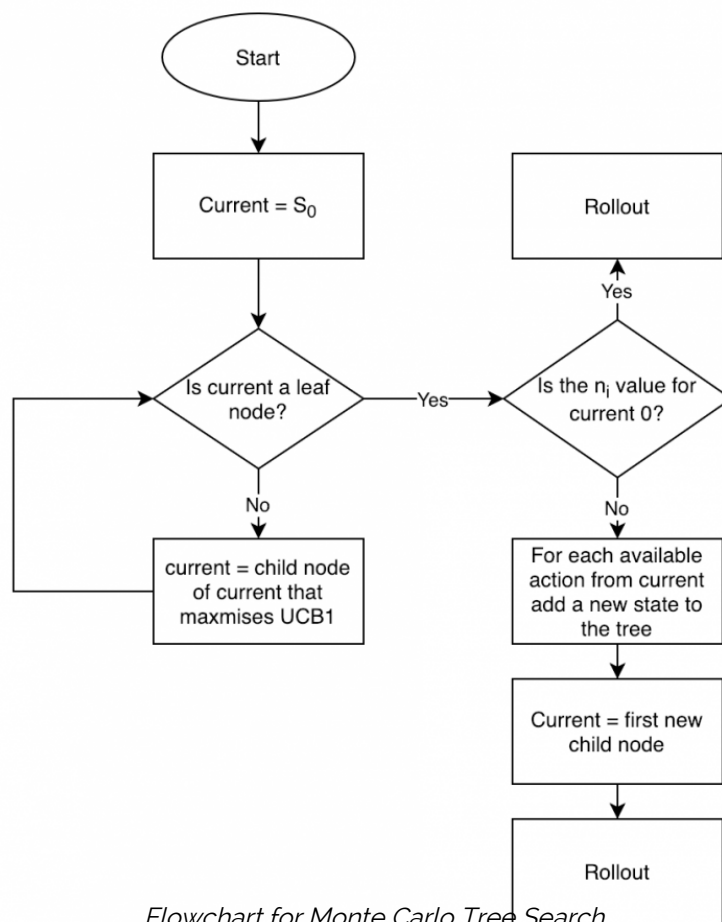
S_i = Simulate(S_i , A_i)

This loop will run forever until you reach a terminal state.

Download this learning path to start your data science journey.



Download Resource



Tree Traversal & Node Expansion

You start with S_0 , which is the initial state. If the current node is not a leaf node, we calculate the values for UCB1 and choose the node that maximises the UCB value. We keep doing this until we reach the leaf node.

Next, we ask how many times this leaf node was sampled. If it's never been sampled before, we simply do a rollout (instead of expanding). However, if it has been sampled before, then we add a new node (state) to the tree for each available action (which we are calling expansion here).

Your current node is now this newly created node. We then do a rollout from this step.

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.

Complete Walkthrough with an example

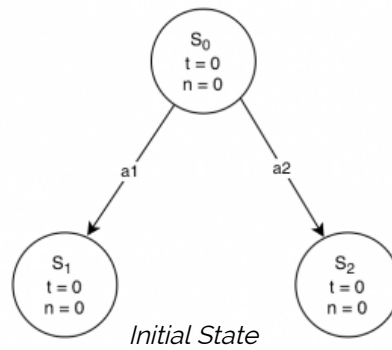
Let's do a complete walkthrough of the algorithm to truly ingrain this concept and understand it in a lucid manner.

Iteration 1:

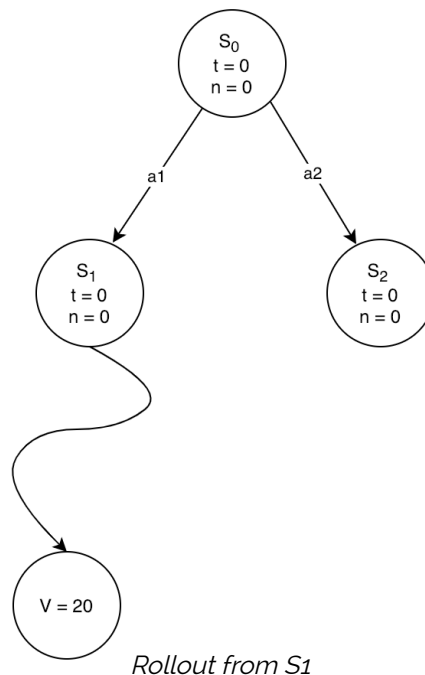
- We start with an initial state S_0 . Here, we have actions a_1 and a_2 which lead to states s_1 and s_2 with total score t and number of visits n . But how do we choose between the 2 child nodes?



Download Resource



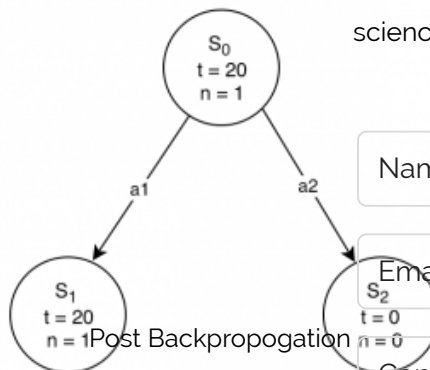
- This is where we calculate the UCB values for both the child nodes and take whichever node maximises that value. Since none of the nodes have been visited yet, the second term is infinite for both. Hence, we are just going to take the first node
- We are now at a leaf node where we need to check whether we have visited it. As it turns out, we haven't. In this case, on the basis of the algorithm, we do a rollout all the way down to the terminal state. Let's say the value of this rollout is 20



- Now comes the 4th phase, or the backpropagation phase. The value of the leaf node (20) is backpropagated all the way to the root node. So now, $t = 20$ and $n = 1$ for nodes S_1 and S_0 .

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.



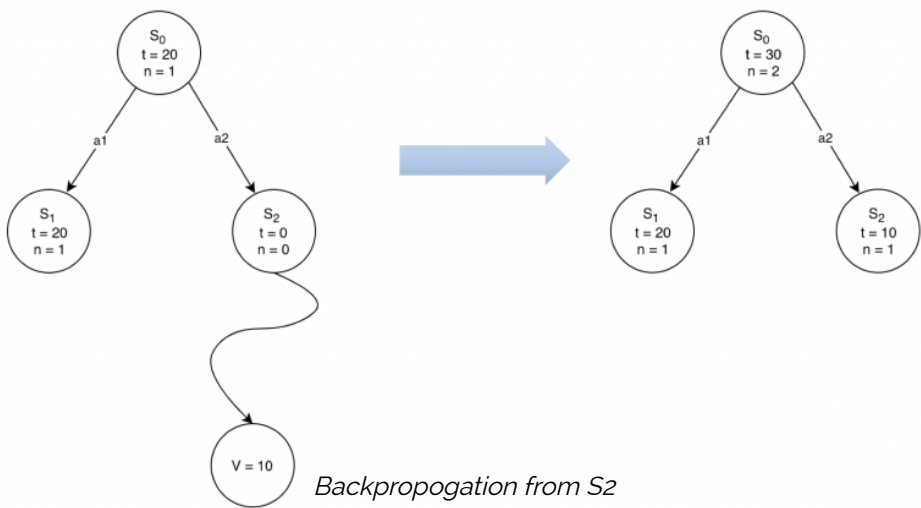
- That's the end of the first iteration

 [Download Resource](#)

The way MCTS works is that we run it for a defined number of iterations or until we are out of time. This will tell us what is the best action at each step that one should take to get the maximum return.

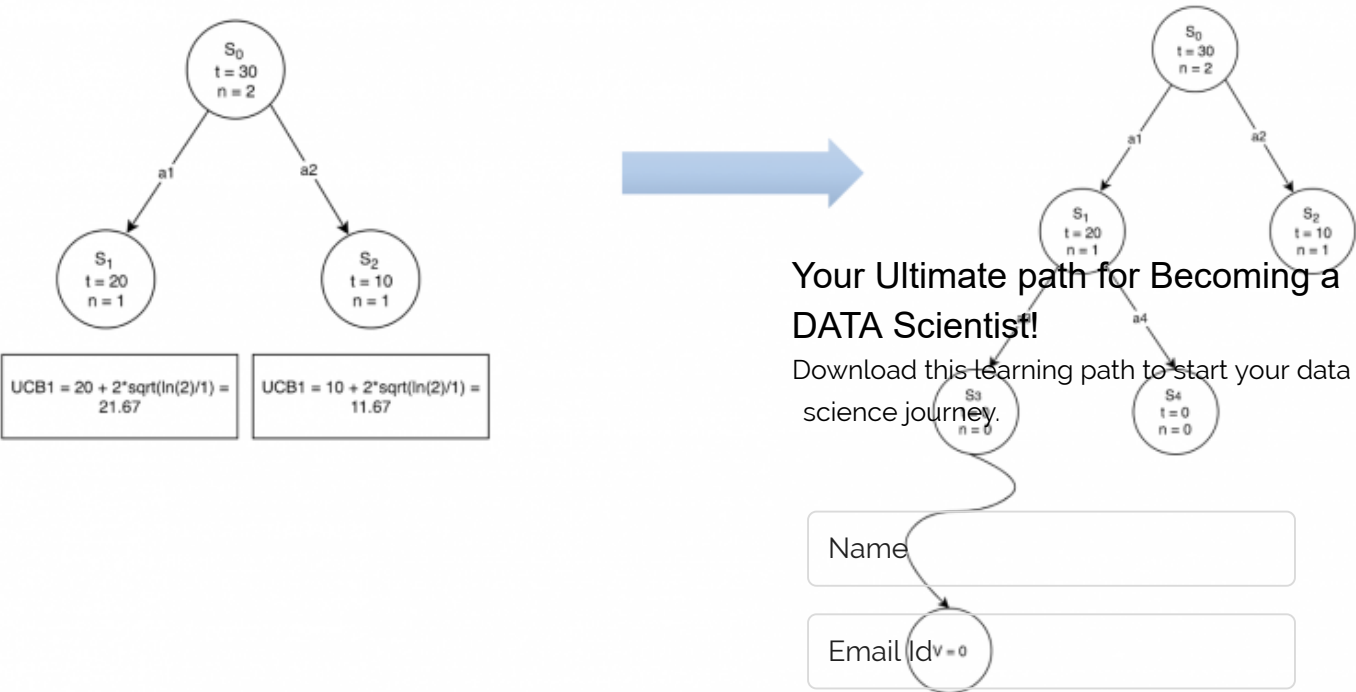
Iteration 2:

- We go back to the initial state and ask which child node to visit next. Once again, we calculate the UCB values, which will be $20 + 2 * \sqrt{\ln(1)/1} = 20$ for S1 and infinity for S2. Since S2 has the higher value, we will choose that node
- Rollout will be done at S2 to get to the value 10 which will be backpropogated to the root node. The value at root node now is 30



Iteration 3:

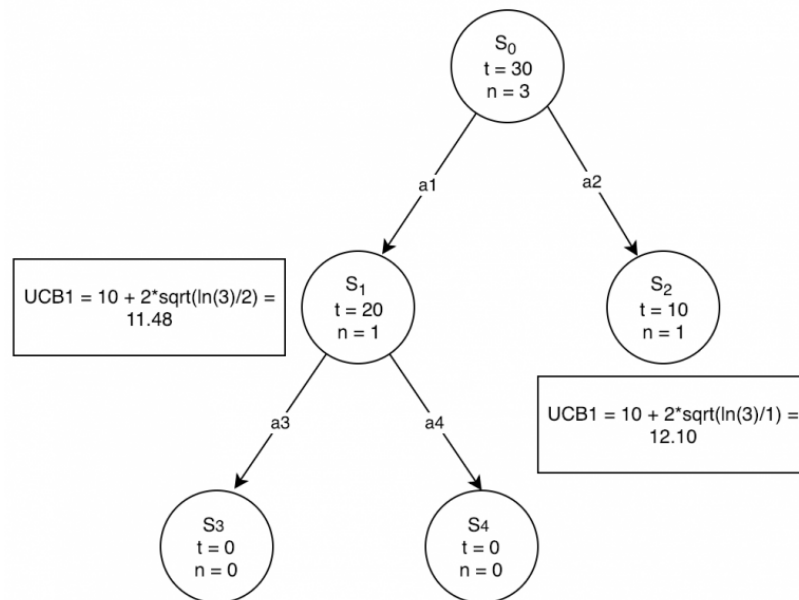
- In the below diagram, S1 has a higher UCB1 value and hence the expansion should be done here:



- Now at S1, we are in exactly the same position as the initial state with the UCB1 values for both nodes as infinite. We do a rollout from S3 and end up getting a value of 0 at the leaf node

Iteration 4:

- We again have to choose between S1 and S2. The UCB value for S1 comes out to be 11.48 and 12.10 for S2:



- We'll do the expansion step at S2 since that's our new current node. On expansion, 2 new nodes are created – S5 and S6. Since these are 2 new states, a rollout is done till the leaf node to get the value and backpropagate

That is the gist of this algorithm. We can perform more iterations as long as required (or is computationally possible). The underlying idea is that the estimate of values at each node becomes more accurate as the number of iterations keep increasing.

End Notes

Deepmind's AlphaGo and AlphaGo Zero programs are far more complex with various other facets that are outside the scope of this article. However, the Monte Carlo Tree Search algorithm remains at the heart of it. MCTS plays the primary role in making complex games like Go easier to crack in a finite amount of time. Some open source implementations of MCTS are linked below:

[Implementation in Python \(https://github.com/int8/monte-carlo-tree-search\)](https://github.com/int8/monte-carlo-tree-search)

[Implementation in C++ \(https://github.com/PetterS/monte-carlo-tree-search\)](https://github.com/PetterS/monte-carlo-tree-search)

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.

I expect reinforcement learning to make a lot of headway in 2019. It won't be surprising to see a lot more complex games being cracked by machines soon. This is a great time to learn reinforcement learning!

I would love to hear your thoughts and suggestions regarding this article and this algorithm in the comments section below. Have you used this algorithm before? If not, which game would you want to try it out on?

Name

Email Id


You can also read this article on Analytics Vidhya's Android APP




[//play.google.com/store/apps/details?](https://play.google.com/store/apps/details?)

Download Resource


Share this:

 (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/?share=linkedin&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/?share=facebook&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/?share=twitter&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/?share=pocket&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/?share=reddit&nb=1>)

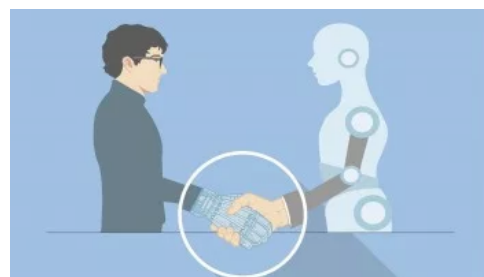
Like this:

Loading...

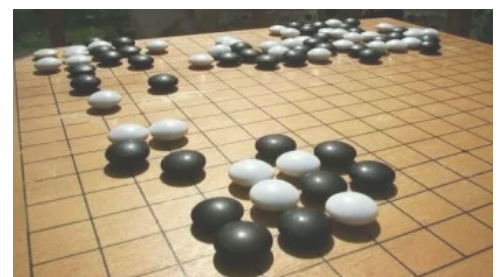
Related Articles



<https://www.analyticsvidhya.com/blog/2018/01/minigo-an-open-source-python-implementation-inspired-by-deepminds-alphago/>
Minigo: An Open-Source Python Implementation Inspired By DeepMind's AlphaGo
(<https://www.analyticsvidhya.com/blog/2018/01/minigo-an-open-source-python-implementation-inspired-by-deepminds-alphago/>)
January 31, 2018
In "AVbytes"



<https://www.analyticsvidhya.com/blog/2016/12/artificial-intelligence-demystified/>
Artificial Intelligence Demystified
(<https://www.analyticsvidhya.com/blog/2016/12/artificial-intelligence-demystified/>)
December 23, 2016
In "Machine Learning"



<https://www.analyticsvidhya.com/blog/2018/05/facebook-open-sources-elf-opengo/>
Inspired by DeepMind, Facebook Open Sources It's Own Go Beating Algorithm
(<https://www.analyticsvidhya.com/blog/2018/05/facebook-open-sources-elf-opengo/>)
May 5, 2018
In "AVbytes"

Your Ultimate path for Becoming a
DATA Scientist!

Download this learning path to start your data science journey.

 Download Resource

TAGS : [AI \(https://www.analyticsvidhya.com/blog/tag/ai/\)](https://www.analyticsvidhya.com/blog/tag/ai/), [ARTIFICIAL INTELLIGENCE \(https://www.analyticsvidhya.com/blog/tag/artificial-intelligence/\)](https://www.analyticsvidhya.com/blog/tag/artificial-intelligence/), [PYTHON \(https://www.analyticsvidhya.com/blog/tag/python/\)](https://www.analyticsvidhya.com/blog/tag/python/), [REINFORCEMENT LEARNING \(https://www.analyticsvidhya.com/blog/tag/reinforcement-learning/\)](https://www.analyticsvidhya.com/blog/tag/reinforcement-learning/), [RL \(https://www.analyticsvidhya.com/blog/tag/rl/\)](https://www.analyticsvidhya.com/blog/tag/rl/)

NEXT ARTICLE

DataHack Radio #16: Kaggle Grandmaster SRK's Journey and Advice for Data Science Competitions

(<https://www.analyticsvidhya.com/blog/2019/01/datahack-radio-tips-crack-data-science-competitions-kaggle-grandmaster/>)

...

PREVIOUS ARTICLE

Top 10 Presentations from rstudio::conf 2019 – The Best R Conference of the Year!

(<https://www.analyticsvidhya.com/blog/2019/01/top-highlights-rstudioconf-2019-best-r-conference/>)



(<https://www.analyticsvidhya.com/blog/author/ankit2106/>)

Ankit Choudhary (<https://www.analyticsvidhya.com/blog/author/ankit2106/>)

IIT Bombay Graduate with a Masters and Bachelors in Electrical Engineering. I have previously worked as a lead decision scientist for Indian National Congress deploying statistical models (Segmentation, K-Nearest Neighbours) to help party leadership/Team make data-driven decisions. My interest lies in putting data in heart of business for data-driven decision making.

[in](https://www.linkedin.com/in/ankit-choudhary-b9360826/) (<https://www.linkedin.com/in/ankit-choudhary-b9360826/>)

4 COMMENTS



VAMSI

January 27, 2019 at 11:01 pm (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/#comment-156816>)

Your Ultimate path for Becoming a
DATA Scientist!

[Reply](#)

Download this learning path to start your data science journey.

Is it Best First Search Tree or Breadth First Search Tree?



ANKIT CHOUDHARY

January 27, 2019 at 11:08 pm (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/#comment-156817>)

Name

[Reply](#)

Email Id

For more information go to this link:

Contact Number

Hi Vamsi, this is best first search as this is not uninformed tree search. https://en.wikipedia.org/wiki/Best-first_search (https://en.wikipedia.org/wiki/Best-first_search)



Download Resource



KARTHIK

[Reply](#)

February 13, 2019 at 1:14 pm (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/#comment-157056>)

Thank you . It was easy to understand .



DAWID

[Reply](#)

May 29, 2019 at 2:47 am (<https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/#comment-158359>)

Great article, brilliant explanation!

LEAVE A REPLY

Your email address will not be published.

Comment

Name (required)

Email (required)

Website

SUBMIT COMMENT

☐ Notify me of new posts by email.

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.

JOIN THE NEXTGEN DATA SCIENCE ECOSYSTEM

Get access to free courses on Analytics Vidhya

Get free downloadable resource from Analytics Vidhya

Save your articles

Participate in hackathons and win prizes

Name

Email Id

Contact Number



Download Resource

(https://id.analyticsvidhya.com/accounts/login/?
next=https://www.analyticsvidhya.com/blog/?
utm_source=blog-subscribe&utm_medium=web)

Join Now



(https://software.seek.intel.com/DataCenter_to_Edge_REG?

registration_source=AnalyticsVidhya-APJ)

POPULAR POSTS

- 24 Ultimate Data Science Projects To Boost Your Knowledge and Skills (& can be accessed freely)
(https://www.analyticsvidhya.com/blog/2018/05/24-ultimate-data-science-projects-to-boost-your-knowledge-and-skills/)
- Essentials of Machine Learning Algorithms (with Python and R Codes)
(https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/)
- 7 Types of Regression Techniques you should know!
(https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/)
- A Complete Tutorial to Learn Data Science with Python from Scratch
(https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/)
- Understanding Support Vector Machine algorithm from examples (along with code)
(https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/)
- Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes)
(https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/)
- Introduction to k-Nearest Neighbors: Simplified (with implementation in Python)
(https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/)
- A Simple Introduction to ANOVA (with applications in Excel)
(https://www.analyticsvidhya.com/blog/2018/01/anova-analysis-of-variance/)

Your Ultimate path for Becoming a
DATA Scientist!

Download this learning path to start your data science journey.

Name

Email Id

Contact Number

Download Resource

RECENT POSTS

Top 7 Machine Learning Github Repositories for Data Scientists (<https://www.analyticsvidhya.com/blog/2019/06/top-7-machine-learning-github-repositories-data-scientists/>)

JUNE 6, 2019

The AI Comic: Z.A.I.N – Issue #1: Automating Attendance using Computer Vision (<https://www.analyticsvidhya.com/blog/2019/06/ai-comic-zain-issue-1-automating-computer-vision/>)

JUNE 3, 2019

DataHack Radio #23: Ines Montani and Matthew Honnibal – The Brains behind spaCy (<https://www.analyticsvidhya.com/blog/2019/06/datahack-radio-ines-montani-matthew-honnibal-brains-behind-spacy/>)

JUNE 3, 2019

Exclusive Interview with Sonny Laskar – Kaggle Master and Analytics Vidhya Hackathon Expert (<https://www.analyticsvidhya.com/blog/2019/05/exclusive-interview-sonny-laskar-kaggle-master-analytics-vidhya-hackathon-expert/>)

MAY 30, 2019

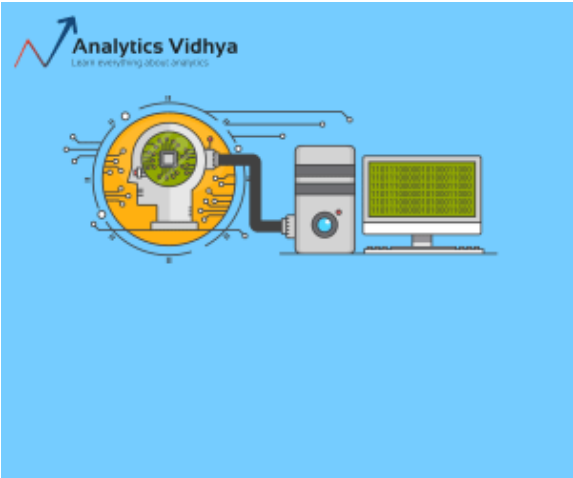


([https://datahack.analyticsvidhya.com/contest/intel-ai-enterprise-](https://datahack.analyticsvidhya.com/contest/intel-ai-enterprise-meetup-mumbai-invite-only/?utm_source=Sticky_banner1&utm_medium=display&utm_campaign=IntelMum)

[meetup-mumbai-invite-only/?utm_source=Sticky_banner1&utm_medium=display&utm_campaign=IntelMum](https://datahack.analyticsvidhya.com/contest/intel-ai-enterprise-meetup-mumbai-invite-only/?utm_source=Sticky_banner1&utm_medium=display&utm_campaign=IntelMum))

Your Ultimate path for Becoming a DATA Scientist!

Download this learning path to start your data science journey.




([https://courses.analyticsvidhya.com/courses/applied-machine-](https://courses.analyticsvidhya.com/courses/applied-machine-learning-beginner-to-professional?utm_source=Sticky_banner2&utm_medium=display&utm_campaign=Applied_ML)

[learning-beginner-to-professional?utm_source=Sticky_banner2&utm_medium=display&utm_campaign=Applied_ML](https://courses.analyticsvidhya.com/courses/applied-machine-learning-beginner-to-professional?utm_source=Sticky_banner2&utm_medium=display&utm_campaign=Applied_ML))

Name

Email Id

Contact Number

Download Resource

