

Connect 4 With Machine Learning

<Date>

Table of Contents

[Analysis 3](#_Toc51675487)

[Documented Design 4](#_Toc51675488)

[Technical Solution 5](#_Toc51675489)

[Testing 6](#_Toc51675490)

[Evaluation 7](#_Toc51675491)

# Analysis

## Introduction

The problem area that I am interested in is Machine Learning which is a large topic in computer science research. I am planning on creating a machine learning algorithm which will train a neural network to act like a human at playing the board game, Connect 4. The main reason I chose this was because I watched a video on machine learning[[1]](#footnote-1) and was instantly mesmerized. I decided that I wanted to learn how to make something similar and chose Connect 4 as the game because it is quite simple and can demonstrate a learning AI quite well.

Machine learning algorithms have been around since the 1950s but has only recently taken off due to the rise of the internet and masses of data being transferred daily. Companies such as Google and Facebook are using machine learning algorithms daily to provide a better user experience, better than any human could do on their own. Many tools have been made to allow machine learning to be used by many people without much background in the topic itself. One of my inspirations is Google’s DeepMind[[2]](#footnote-2) AI which taught itself to beat a world-champion at the Chinese game of Go[[3]](#footnote-3).

Machine learning is based on mathematical ideas to adjust parameters such that a cost function is minimized. In my case of Connect 4, the cost function will be how different the AI’s choice is to the human’s choice so that when training, the AI will minimize this and start imitating the human better. I will derive all mathematics behind the algorithm as it is quite complex. The implementation is simple but, can get quite complex for efficient code due to a few tricks in organising the data in matrices and using matrix math to do calculations.

I am planning on only using a single Machine Learning algorithm for my final product as the number of tweaks to parameters I can make to the algorithm is massive. I should be able to adjust the algorithm so that I can get the most performance out of the single algorithm. If I finish the algorithm and realize that I there aren’t enough parameters to tweak, I can add another algorithm but this will add a lot of complexity and take a lot of time as a large rework will have to be made. This is because I would have to allow a modular kind of program where the number of algorithms available isn’t hardcoded, but more can be added at any time.

# Documented Design

# Technical Solution

# Testing

# Evaluation

1. <https://youtu.be/gn4nRCC9TwQ> [↑](#footnote-ref-1)
2. [https://deepmind.com](.git) [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Go_(game)> [↑](#footnote-ref-3)