Artificial Civ: Deep Learning Strategies for Multiplayer Turn-Based Video Games

Cohl Dorsey, Jackson Nelson

Advised by Erik Steinmetz

Augsburg University

Abstract

This project analyzes the performance of an artificial intelligence (AI) in a non-perfect information, stochastic video game: one where players do not know all available information, and has variables with elements of randomness. Whereas other projects of this nature have focused simply on the reaction of the AI to the pixels shown on the computer screen, we attempt to utilize methods of machine learning to teach the AI successful strategies in the video game Freeciv using game-state information. Our method of teaching the AI involves the use of a neural network and reinforced learning such that the AI player learns successful strategy. Our intended goal is to modify the AI such that it would use predictive behavior rather than reactive behavior when it takes an action.

Background

Development of artificial intelligence has historically focused on perfect information, non-stochastic environments, where there is no hidden knowledge or elements of randomness (Buchanan 2005). This type of research often involves the use of objectively correct information or expert opinions that allow AI to learn what is right or wrong in the given context. Yet this work has left much to be desired in cases where there is no objectively correct data and expert opinion would be too expensive or time consuming to produce. Work in the former area is effectively finished, with the growing success of tasks such as image and speech recognition (LeCun et al. 2015). Similar success is being found in perfect-information yet stochastic environments, exemplified by AI programs that have completely outclassed the performance of humans in games such as Chess, and further demonstrated by the AlphaGo Zero experiment from DeepMind, a division of Google.

In the AlphaGo Zero experiment, DeepMind created an AI that was able to achieve superhuman performance in the ancient Chinese game Go by training only against itself and with minimal prior knowledge of the game (Silver, et al. 2017). Zero is only the latest iteration of DeepMind's long sought after goal: AlphaGo was created in 2014 and has subsequently gone through several versions, with more than three years of work preceding Zero. The previous versions of AlphaGo differed from Zero in that they were trained using thousands of sample matches from expert human Go players. Zero, on the other hand, learned through matches against itself. Initially having only knowledge of basic rules, Zero eventually came to surpass all human competitors and previous versions of AlphaGo alike. It took forty days of self training for Zero to become the world's premier player of Go.

Despite AlphaGo Zero's success, little has been done thus far with AI in non-perfect information, stochastic environments, most of which seems to be representative of reactive behavior (Kunanusont, Kamolwan, et al. 2017). The AI currently used in these situations frequently is only reacting to the elements on screen, without any form of long-term strategy. The focus of this research is to explore the idea of manipulating an AI such that it does incorporate strategy and predictive action in a stochastic, non-perfect information environment. We have sought to emulate the style of AlphaGo Zero in our own experiments; in our study, we would also refrain from teaching the AI and instead allow it to learn on its own through repeated trials. Given that the scope of stochastic environments is significantly broader than non-stochastic environments, the goal of our research is quite relevant to the effectiveness of AI in many different fields in the future. Some possible implications include financial trading, security applications, and many more. Furthermore, numerous video games utilize AI, and our research could contribute to the improvement of AI in any of these games.

The necessity of a non-perfect information environment with stochastic elements precluded the use of classic games such Chess and Go, which otherwise are frequent subjects for similar tasks (Buchanan 2005). Instead, we sought a game that encouraged the use of strategy, had a complexity on the scale of Chess or Go, and was open-source. This led us to selecting Freeciv, an open-source strategy video-game heavily inspired by Firaxis' *Civilization* series of video-games, as the subject of our research (freeciv.org). Freeciv is a turn-based strategy game where players take the role of a tribal leader of one of hundreds of historical civilizations. Players must guide and build a successful nation through the development of cities and expansion of their military. Players discover new technologies as they progress through the game, which allow them to build new buildings, establish relations with other players, change governments, and develop new military units. At the end of each turn, a score is assigned to players that is based on the size of their civilization, their scientific progress, and other factors estimating the success of each civilization.

Freeciv was a relevant choice as its depth allows for numerous different strategies, the game can be manipulated to allow for perfect and non-perfect information implementations, and the game is open-source, meaning the source code of the game is publically available and can be freely modified. This allows us to inspect and modify the underlying code of the game, which otherwise would not be possible, and permits the inclusion of a neural network within the game's code.

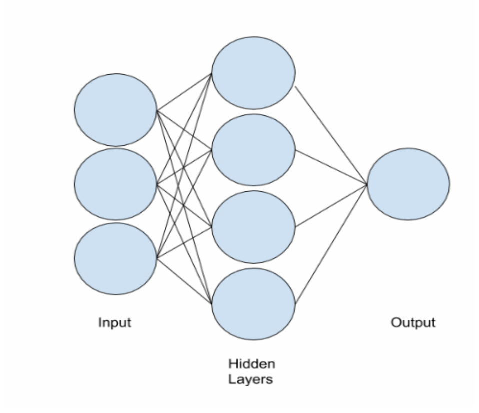

Figure 1: A sample neural network model

A neural network is a general class of algorithms designed to teach an AI to accomplish a specific task (Hardesty, Larry 2017). The structure of a neural network is intended to simulate the function of neurons within the human brain, which, in theory, allows an AI to ‘think.’ (Wang, Haohan. Raj, Bhiksha 2017) All neural networks are composed of an input layer, an output layer, and a variable number of hidden layers. Each of these layers is composed of individual, densely connected nodes that utilize a function which provides a value that will be fed to the next layer. The input layer contains as many nodes as there are predefined inputs to the neural network: the data that is being manipulated. The output layer contains a small number of nodes that represent the output of the network: the finalized, manipulated data. The hidden layers differ for each network: there may be any number of hidden layers, and each layer may contain any number of nodes. The purpose of the hidden layers is to provide small adjustments to the data as it is passed through each node. The data is manipulated through the use of nonlinear functions, multiplying input vectors by an unknown weight, taking the sum of those values, and evaluating them: if the sum exceeds a certain threshold, the node passes the data to the next node, mimicking how a neuron within the brain fires. If the data does not exceed that threshold, the sum is not passed and the data is halted. A neural network is said to be deep when it contains more than one hidden layer, meaning that more revisions are applied to the data before it is sent to the output layer (Foote, Keith 2017). The novelty of a neural network stems from their functionality remaining a mystery to scientists: weights and thresholds are initially randomized, then as the network evolves the weights and thresholds are continuously adjusted in unknown ways, and therefore it is unknown how input data is transformed into functional outputs.

Method

The scope of this project is defined by four overarching ideas: the creation of a feed-forward deep neural network, which is the mathematical model that would provide the structure for learning; the use of TensorFlow, a library of prebuilt functions with significant applications to machine learning tasks; an unsupervised learning algorithm, where the AI that we are using does not receive human input during the course of its training; and reinforcement learning, a style of machine learning where certain actions reward the AI while others penalize it, thereby incentivizing in-game strategies that lead to success.

Concurrent use of both reinforcement learning and unsupervised learning for machine learning is intentional: use of unsupervised learning frees an AI from the limits of human knowledge, allowing the system to develop strategy that a human may not conceive of, while sparing the grueling process of collecting accurate training data. Meanwhile, reinforcement learning allows for the system to dynamically adapt its strategy by comparing its success to itself and that of its opponents mid-game, rather than simply checking a win-loss state postmortem, thereby likely accelerating the rate of progress.

We began our research with a foundational study of the tools necessary for the task at hand: TensorFlow, the programming languages Python and C, as well as an investigation into the inner-workings of Freeciv. Upon delving into the game's source code, we discovered that the in-game AI was governed by three essential numeric traits that dictate the player's actions: Trader, which influences the AI's desire to establish trade with other players and develop its own economy; Expansionist, which influences the AI's desire to settle new cities and expand their empire; and Aggression, which influences the AI's desire to wage war against other players. These traits, varying on a scale of 1-100, each work in tandem to guide the actions of the AI, with higher values of Aggression and Expansionist favoring military development, while higher values of Trader favor economic development. Rather than developing our own AI to play Freeciv, we instead chose to modify the preexisting AI by constructing a neural network that would take game-state information as inputs and return the three trait values as outputs. These modified trait values would then be fed back into the game, and the relative difference in score between turns would signify a successful or unsuccessful turn. The neural network would then use this difference to develop a policy for which values of the traits led to a more successful outcome, and thereby learn effective strategy within the game.

Currently, we are working to solve two problems: we are continuously working to create a hook wherein we can retrieve data as the game is running, and subsequently feed that information into our neural network. We have thus far established the structure of both our neural network and hook, and continue to work on the implementation to make them functional. Our neural network will take in seven variables as inputs: the player's current gold, government type, number of cities, number of technologies researched, number of allied players, number of players they are at war with, and a condition if there are enemy units nearby. These values are then fed through three hidden layers, each consisting of five nodes. The hidden layers take their input as a vector (a 1 x *n* array of values, with *n* being the number of inputs), perform matrix multiplication with the computed weight, and apply the non-linear function ReLU (Rectified Linear Unit) to normalize the values to zero if they are negative, or a positive decimal value otherwise, resulting in a new vector to be fed to the subsequent nodes.

The hook to retrieve and supply information with the game will be embedded within the main loop of the game: the section of code that is iterated over with every passing turn to update and process new information, while prompting the next player of their turn. Prior to implementing real data pulled during the course of a game, we will utilize artificial sample data that is representative of data that could have been pulled from a simulation round and feed it to our neural network in order to demonstrate its ability.

Conclusions

Significant complications that arose during the course of this project include discovering the exact ways in which the AI is governed and takes actions, how information within the game is processed, and how data is passed from players to the server where it is processed. Freeciv, in particular, suffers from the fact that since it is open source, many disparate parties have been involved in separate aspects of its development, resulting in poor documentation and confusion regarding how the entire system works together. This lack of documentation required us to devote more time to investigating the source code of the game, which in turn delayed work on other aspects of the project.

Future Work

Currently, much remains to be done concerning the actual results of our research: we have yet to finish a functional neural network, utilize real data, or implement any kind of training on the preexisting AI. The scope of such a project is not feasible to complete within two months of work: similar projects such as AlphaGo are the result of many years worth of work.

We will continue working to develop our neural network and implement it with Freeciv over the coming months. We will first continue working to retrieve the necessary game-state information from Freeciv and either write it to a separate file or pull it directly into the neural network. Once this is accomplished, we will further improve the functionality of the neural network; initially such that it is able to process data from individual turns, then expanding to a full game, and finally to processing multiple games. We will observe the performance of the AI during each of these situations and document its success or failure. We hope that during each of these situations we are able to observe progress in improving the policies that the AI is using. We will test the performance of the AI using our neural network over many thousands, if not millions, of simulation games and record the results. Our ultimate measure of success will be when we are able to observe that the AI is not only learning throughout a single game, but when it is able to begin each round with previous experience in mind, and therefore not beginning *tabula rasa* each game. Should we be successful in our endeavors moving forward, a second paper will be drafted detailing our results and process throughout the duration of this project, and our findings will be presented in future public forums. Furthermore, if we are successful, we may contact the development team behind Freeciv and offer our code to be implemented in future releases of the game.

References

Buchanan, Bruce. "A (Very) Brief History of Artificial Intelligence." *AI Magazine*, 2005, pp. 53–60.

Foote, Keith. "A Brief History Of Deep Learning" Dataversity, 7 February, 2017. http://www.dataversity.net/brief-history-deep-learning/

"Freeciv.org - Open Source Empire-Building Strategy Game.", 5 Jan. 1996, freeciv.org/.

Hardesty, Larry. "Explained: Neural Networks" MIT News Office, 14 April 2017. http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Kunanusont, Kamolwan, et al. "General Video Game AI: Learning from Screen Capture." *2017 IEEE Congress on*

Evolutionary Computation (CEC), 2017, doi:10.1109/cec.2017.7969556.

LeCun, Yann, et al. “Deep Learning.” *Nature*, 28 May 2015, pp. 436–444.

Silver, David, et al. “Mastering the game of go without human knowledge.” Nature, 550:354– 359, 2017.

Stone, Peter, et al. "Artificial Intelligence and Life in 2030." One Hundred Year Study on Artificial Intelligence: Report of the 2015-2016 Study Panel, Stanford University, Stanford, CA, September 2016.

Wang, Haohan. Raj, Bhiksha. “On the Origin of Deep Learning” arXiv:1702.07800v4, 3 Mar 2017, https://arxiv.org/abs/1702.07800