Evolution of Neural Network Controllers for Gameplay Behaviours

Ryan O'Flaherty
40168766

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of
BSc (Hons) Games Development

School of Computing

October 24, 2017

Authorship Declaration

I, Ryan O'Flaherty, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed:

Date:

Matriculation no:

Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

Contents

List of Tables

List of Figures

Acknowledgements

Insert acknowledgements here

I would like to thank my cat, dog and family.

1 Introduction

You can fill out sections as you please.

Most of the formatting is taken care for you but you can add this yourself as you please.

1.1 Project Aims and Structure

Or have sections that are relevant to your main body of work above but warrant there own section. Both - with numbering would be entered into the Table of contents.

1.1.1 Overview Of Project Content and Milestones

This is a sub sub section with a list of bullet points.

- A working X, that will be used for this investigation.
- Investigation of current tools and their potential use during an investigation of X .
- Programming of X with related frameworks Y and Z.
- That is all.

2 Background

2.1 Introduction

The following section of this dissertation will go on to discuss the history of artificial intelligence within the context of video games, before going on to explain neural networks, evolutionary algorithms, and the NeuroEvolution of Augmenting Topologies (NEAT) library.

2.2 History of AI in Games

Video games have been a popular area of interest for artificial intelligence developers and researchers for many decades.

Over several tens of years, a large amount of research and development has been done in an attempt to perfect chess-playing artificial intelligence agents[**?**], and work has more recently been put in to do the same with the game of Go.

In March of 2016, the goal of getting such an agent to compete and win at the highest level was reached, when AlphaGo, a program engineered by Google, managed to overcome the Go world champion human player, Lee Sedol[**?**]. This was then reported as a major breakthrough for the artificial intelligence field.

2.3 What is a Neural Network?

Artificial neural networks are a rough representation of the human brain. They are built using a series of layers of network nodes. These nodes are used to represent neurons in the brain. The first is an input layer, followed by two or three hidden layers before a final output layer of nodes[**?**]. The connection between these nodes is representative of axons in the brain. The input and output layers of the network take data in and output results respectively, with the processing of said data being done within the hidden layers - but how does it work?

2.3.1 Perceptrons

One type of artificial neuron (or node) is known as a 'perceptron.' Each perceptron receives several inputs and uses them to produce one binary output[**?**].

Frank Rosenblatt, the scientist who developed the perceptron in the 1950s and 1960s, devised a rule for computing the output from these neurons. Using what he called 'weights,' the importance of each input is assessed and expressed. Each input has a weight assigned to it, and the resultant output from these inputs - either a 1 or 0 - is dependent on whether or not some

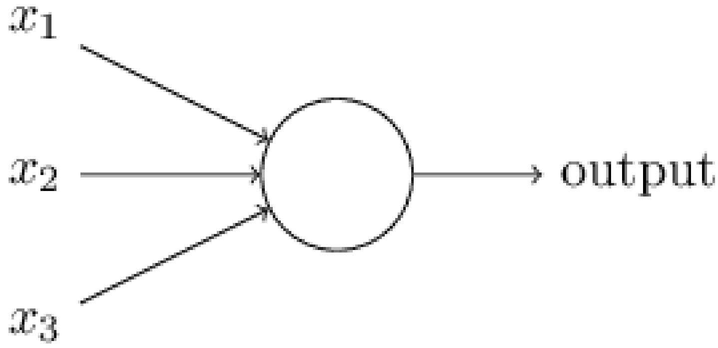

Figure 1: A Single Perceptron

threshold value is less than or greater than the sum of the weights from all of the inputs to that particular perceptron. Therefore, if the weighted sum is less than or equal to the threshold value, the output is a 0. Otherwise, a 1[?]. Both the threshold value and the input weights are real numbers. These can be tweaked to alter the decisions made by a neural network.

2.3.2 Sigmoid Neurons

Sigmoid neurons are akin to perceptrons, however, they are modified in such a way that marginal alterations in their weights and bias cause only a small change to their output[?]. This crucial difference is what affords a network consisting of sigmoid neurons the ability to learn.

The inputs to a sigmoid neuron also differ from those of perceptrons. Rather than being binary (1 or 0), these inputs are any number *between* 1 and 0. Much like with perceptrons, these sigmoid neuron inputs are weighted, with an overall bias included. These can be denoted b, w_1, w_2, ... w_n where b represents the bias, and each w is an input weight. This time however, the output is non-binary. To calculate it, we use

$$\sigma(w \cdot x + b) \tag{1}$$

where σ is known as the sigmoid function, which is defined as:

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}. \tag{2}$$

In its full extended form, with x being used to symbolise the inputs, the output of a sigmoid neuron is calculated as

$$\frac{1}{1 + \exp(-\sum_j w_j x_j - b)}. \tag{3}$$

//
To understand the similarity to the perceptron model, suppose z $\equiv w \cdot x + b$ is a large positive number. Then $e^{-z} \approx 0$ and so $\sigma(z) \approx 1$. In other words, when $z = w \cdot x + b$ is large and positive, the output from the sigmoid neuron is approximately 1, just as it would have been for a perceptron. Suppose on the other hand that $z = w \cdot x + b$ is very negative. Then $e^{-z} \to \infty$, and $\sigma(z) \approx 0$. So when $z = w \cdot x + b$ is very negative, the behaviour of a sigmoid neuron also closely approximates a perceptron. It's only when $w \cdot x + b$ is of modest size that there's much deviation from the perceptron model[**?**].
//

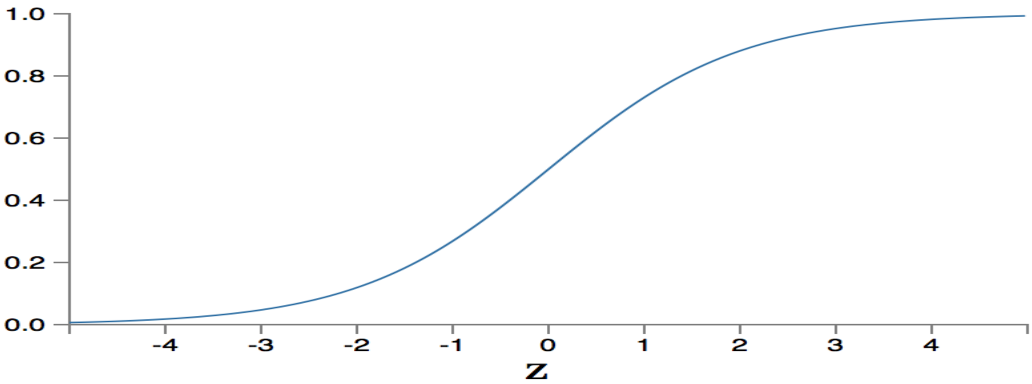

Figure 2: Sigmoid Function

The shape of a plotted σ function can be seen in Figure 2.
//
The smooth nature of σ means that minor changes in the weights and bias - which are depicted as Δw_j and Δb_j respectively - will consequently create small changes to the output from the neuron. That change - Δoutput - can be approximated with calculus:

$$\Delta \text{output} \approx \sum_j \frac{\partial \, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \, \text{output}}{\partial b} \Delta b \tag{4}$$

The sum is over all of the weights, w_j, and $\partial \, \text{output}/\partial w_j$ and $\partial \, \text{output}/\partial b$ denote partial derivatives of the output with respect to w_j and b, respectively.

As Δoutput is a linear function of the changes Δw_j and Δb_j in the weights and bias. This linearity makes it easy to choose small changes in the weights and biases to achieve any desired small change in the output. So while sigmoid neurons have much of the same qualitative behaviour as perceptrons, they make it much easier to figure out how changing the weights and biases will change the output.

//

2.4 Evolutionary Algorithms

As the name might suggest, an evolutionary algorithm is one that evolves. It does so to encourage finding the most optimal solution to a problem. A vast amount of varying evolutionary algorithms exist, but at the core of them all is the same principal idea: "given a population of individuals the environmental pressure causes natural selection (survival of the fittest) and this causes a rise in the fitness of the population"[?].

2.5 NeuroEvolution for Augmenting Topologies (NEAT)

"NEAT is a method for evolving speciated neural networks of arbitrary structures and sizes. NEAT leverages the evolution of structure to make neuroevolution more efficient"[?].

2.6 Summary

.

3 Literature Review

And so on for each of the chapters. The template automatically starts new chapters on a new page. The associated guidelines tell you what the available styles do and also how to structure a report. There is a section break on this page that you should be careful NOT to delete otherwise the references and appendices will be numbered continuously with the rest of the document.

4 Additional Information / Knowledge Required

Experience with Linux and managing Virtual machines, networking. So on and so forth...

References

[1] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994.

Appendices

A Project Overview

Initial Project Overview

SOC10101 Honours Project (40 Credits)

Title of Project:

Evolution of Neural Network Controllers for Gameplay Behaviours

Overview of Project Content and Milestones

The idea is to implement a card game with four players. One of the players is the human, another is an AI agent that has no idea how to play the game, and the other two are hard-coded to know the rules and how to play. The intention is for said card game to be Switch, however this is subject to change if the rules are found to be too difficult for the scale of the project – in which case a simpler game will be substituted in.

The agent then learns how to play by trying to make moves based on neural networks. Initially this will be totally random but after the first generation of the algorithm cycle, it will be based on the chromosomes with the highest fitness, which should then begin to provide better results. These moves can be blocked if they are not legal. There'll be a scoring system for the agent that will be negatively affected by illegal moves and it will then use this to learn how to do better the next time it plays. The scoring system will also see the agent penalised for losing or not winning. This will be what our fitness is based on.

It is worth noting that how successful you are in a game of Switch depends entirely on the hand you're dealt, and how your opponents play the hands they are dealt. A lot of the game is about luck, and so negatively affecting the agent's score should take this into account and deploy some leniency.

The project will make use of the NeuroEvolution of Augmenting Topologies (NEAT) library and will most likely be coded in C++. It will use neural network controllers, co-evolving weights and topologies.

The Main Deliverable(s):

- A playable card game that incorporates an Artificial Intelligence agent that must learn how to play the game from scratch based on a score system that penalises the agent for illegal or costly decisions.
- Experimental research into improving the performance (in terms of score) or speeding up the learning process of the agent.
- A report into what positively or negatively affects the agent, and what causes the effects that it has including experiment results using charts and figures. Changes will be made by varying parameter settings of the evolutionary algorithm in a systematic way.

The Target Audience for the Deliverable(s):

Whilst the final product will be a playable game, it will really be aimed more at being experimental research into Artificial Intelligence techniques and, more specifically, evolving neural network controllers for playing games. Thus, the audience most likely to be interested in the project are those who also want to look into artificial intelligence agents.

The Work to be Undertaken:

- Design and build a game of Switch without the AI agent
- Thoroughly test the barebones game to ensure it works perfectly without bugs
- Research neural networks and evolutionary algorithms
- Implement the AI agent
- Experiment with a few different techniques and test how they perform in terms of improving or decreasing the agent's intelligence/performance in game.

Additional Information / Knowledge Required:
Neural networks and evolutionary algorithms

Information Sources that Provide a Context for the Project:

- Lubberts, & Miikkulainen (2001). Co-Evolving a Go-Playing Neural Network.
- Stanley, Bryant, & Miikkulainen (2005). Evolving Neural Network Agents in the NERO Video Game. IEEE Press.
- Thrun (1995). Learning to Play the Game of Chess. MIT Press.

The Importance of the Project:
Exploring possibilities and limits of AI in games, particularly evolved controllers which do not have to be hard-coded.

The Key Challenge(s) to be Overcome:

- Complete lack of knowledge and experience with Artificial Intelligence techniques

A.A Example sub appendices

...

B Second Formal Review Output

Insert a copy of the project review form you were given at the end of the review by the second marker

C Diary Sheets (or other project management evidence)

Insert diary sheets here together with any project management plan you have

D Appendix 4 and following

insert content here and for each of the other appendices, the title may be just on a page by itself, the pages of the appendices are not numbered, unless an included document such as a user manual or design document is itself pager numbered.