VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Logical Neural Networks: Opening the black box

Daniel Thomas Braithwaite

Supervisor: Marcus Frean

Submitted in partial fulfilment of the requirements for
Bachelor of Science with Honours in Computer Science.

Abstract

Neural Networks (NN's) have been shown to be universal function approximators and have been employed to solve a number of non-trivial problems. The downside to these networks is that they are a black box, diffcult to intepret the trained NN's weights. Previous research has shown that Logical Neural Networks applied to the MINST data set obtains comparable performance to standard NN's and a more understandable learned representation. We aim to build on this by prodiving evedence that builds a foundation for LLN's.

1. Introduction

Neural Networks (NN's) perform exceptionally well over a wide range of different problems. However once trained these networks become a black box, near impossible for a human to interpret the learned representation and understand patterns in the features identified by the NN.

Logical Neural Networks (LLN's) are NN's with activation functions which resemble logic gates. The idea being that if we are able to think about the output of neuron as a logical combination of inputs and weights then prehaps we can understand the learned structure of the network.

LNN's, have previously been shown to provide a simpler representation while still performing well [**?**] (in the context of the MINST data set). Despite the promise of LNN's they have not been deeply explored.

2. The Problem

LNN's are an exciting concept because of their interpretability, this combined with their reasonable performance (when compared to standard NN's) makes them a promising solution to the usual "black boxeyness" of NN's. We suggest this problem still hasn't been solved as we have yet to develop a foundation for LNN's. First we ask, can it be demonstrated through experementation that like standard NN's, LNN's are also universal function approximators? Before we can hope to answer this key question we also must ask whether we have the right parametrization of logical gates as these form the foundation of an LNN.

Regardless of our answer to the first question a next logical step for our investigation is to ask, what types of problems is an LNN best suited to? Which ones can it not only perform well on but also provide a interpretable learned representation. This will help build the foundation of LNN's, giving us a better understanding of how to configure them for different problems, i.e. network layout and which logical neurons to use.

3. Proposed Solution

To begin this project we take a step back to ensure we have the right approach. A network built out of neurons that resemble logic gates we would expect should be able to learn any given boolean formula. This would show we have good paramaterisations for our logical gates and be a fundamental step towards showing LNN's are universal function approximators. First we must address a key issue with our existing setup, namely our choice of "gates" is not a functionally complete set. So far LNN's have been shown to perform with a reasonable accuracy on the MINST data set using layers consisting of only Noisy-OR and Noisy-AND neurons. Consider the problem of representing some known boolean expression using discrete logic circuit, but we only have access to AND and OR gates. These two gates combined does not make a functionally complete set. We propose to implement a number of different sets of functionally complete neurons, such as NAND and NOR. If an LNN consisted of multiple layers of NAND's or NOR's we would expect to find groups of gates learning to act like some of our other gates. One issue we might encounter here is that there is a set of weights which allows a given network to represent a given boolean expression but the network is unable to learn these weights. The gates we aim to paramaterise and

implement are AND, OR, NOT, NAND, NOR.

The first 4 weeks of this project would be taken to fully understand the concept of Noisy gates, using the bibliography of the previous research and other resources, along with developing our paramaterisation of logical gates. By the end of this we will have produced a bibliography and our gate paramaterisations.

We would like to demonstrait that we can use our logical neurons to learn any given boolean expression (if we have an arbarary number of neurons). We will design experements by generating some boolean expressions and then simply feeding the expression into a LNN. It would be interesting to also feed these expressions into a standard NN for a comparason in peformance. We might find that inpractice that some boolean expressions cant be learnt, in this case we will investigate and provide justification for this outcome. During these experements we can also identify what training data will yeild a LNN which generalises effectively i.e. can we fix an input variable to be only 0 while training and obtain a network which generalises to the case where that variable is 1? We expect to spend 7 weeks investigating and running experements at the conclusion of which we will have evedience allowing us to provide an answer to our question.

Next we address a fundamental question, are LNN's universal function approximators? We design experements in a similar fassion as before except this time rather than being boolean expressions we will generate integer and real valued functions, similarly comparing peformance to standard NN. We allocate 7 weeks for this section, at the conclusion of which we will have evidence to provide an answer to the question posed above.

Now that we will have explored the foundation of LNN's, using what we have discovered we can use our LNN to solve standard benchmark problems and compare against performance of standard NN's. We would use K-Fold Cross Validation, which is commonly used for comparisons like this. It would then be useful to examine the trained LNN's and see if we can interpret how the network has learned to solve the problem. This is a logical place to return to as this really is the motivation behind LNN's. We allocate 5 weeks for this.

An overview of the proposed solution follows

- Produce bibliography and paramaterisation of gates (est: 5 weeks)
- Demonstraiting LNN's are/are not able to learn any known boolean function (est: 7 weeks)
- Demonstraiting LNN's are/are not universal function approximators (est: 7 weeks)
- Evaluate LNN's on benchmark problems (est: 5 weeks)

This only allocates 24 weeks, the left over time is to account for slippages.

4. Evaluating your Solution

A solution to this problem should build the foundation to LNN's. Giving strong evedience through experements that LNN's can or cant learn any known boolean function and strong evidence showing that LNN's are or are not universal function approximators. This would require an in depth investigation into various paramaterisations of logic gates. Following this it should identify where LNN's are most powerful and where a standard NN should

be used instead. Finally a comparason of accuracy and interpretability of LLN's against standard NN's over various benchmark problems.

5. Resource Requirements

I do not for see any requirements I don't already have access to.