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 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, incredibly difficult (if not impossible) to understand how the NN is operating once trained. 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 formally proving a foundation for LLN's, investigating where LNN's are best applied and whether we can approximate our LNN as a discrete logic circuit.

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 (LNN's), that is NN's with logic based activation functions, have previously been shown to provide a more understandable representation while still performing well [1] (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 ability to learn sparse weight representations, thus being easier to interpret once trained. 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 rigorous and formal foundation for LNN's. First we ask, can it be proven 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 noisy 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.

Finally we pose a concept which LNN's seem uniquely qualified to solve. Being able to create hardware NN's would be powerful (as hardware is a lot faster). There doesn't seem to be any intuitive way to achieve this with standard NN's, however approximating the continuous logic gates in our LNN as discrete ones would make it possible. But is it possible to create an approximation in a feasible way without sacrificing to much accuracy? Their will certainly be a trade off between complexity (size) of the logic circuit and the accuracy. We are interested in the most accurate approximation we can achieve given a limited number of logic gates.

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 noisy gates and be a fundamental step towards proving 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 gates. 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. These logical

neurons would be paramaterised so we can observe the same nice property's we see with discrete gates, i.e. NOT(AND) = NAND. 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. 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.

Given how we have designed our continuous gates we can construct sets of functionally complete neurons. Intuitively this would mean that our LNN should be able to learn any known boolean function. As part of testing our LNN's on learning known boolean functions it would be interesting to experiment with the training data presented to the network, how can we train a network which generalizes well? Say we trained a LNN with one of the boolean inputs fixed at 0, then would the network be able to generalize to the case where that input is 1? We expect to spend 6 weeks investigating at the conclusion of which we aim to have a proof demonstrating that LNN's can approximate any know boolean function along with strong evidence showing which collections of neurons work well together, give interpretable representation in trained networks and in what configuration yields the best results. The results here will be interesting as we might find that theoretically our LNN can approximate any boolean function but in practice some can not be learned so easily in practice

Next we address a fundamental question, are LNN's universal function approximators? We will work towards a proof or strong evidence showing this in ether direction. To establish a general idea of which direction we will move in we can train our LNN on some known functions and compare performance against standard NN's. These experiments are not equivalent to a proof but will improve our understanding of what we are trying to prove. At the conclusion of this part of the project we will answer this fundamental question providing a proof or strong evidence to justify our conclusion. We allocate 6 weeks for this section.

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 4 weeks for this.

Seeing we have essentially constructed NN's out of neurons which resemble continuous logic gates, something to explore is can we feasibly approximate these continuous gates with discrete ones? If we where able to achieve this we would be essentially creating hardware NN's. Performing such a task would certainly require more gates than neurons but would this blowup result in something that could feasibly be implemented on a circuit board. We allocate any remaining time to investigating this problem and developing/implementing methods to solve it.

An overview of the proposed solution follows

- Produce bibliography and paramaterisation of gates (est: 4 weeks)

- Proving LNN's can/can not approximate any known boolean function (est: 6 weeks)
- Proving LNN's are/are not universal function approximators (est: 6 weeks)
- Evaluate LNN's on benchmark problems (est: 4 weeks)
- Explore approximating LNN as discrete logic circuit (est: remaining time)

This ends up allocating a large portion of time to the last task, however this accounts for falling behind schedule in some sections.

4. Evaluating your Solution

A solution to this problem should build the foundation to LNN's. Giving a proof that LNN's can approximate any known boolean function and strong evidence/proof showing that LNN's are 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 study demonstrating ways to approximate an LNN as a logic circuit, commenting on the feasibility and accuracy of these methods.

Assuming we are able to show (or give strong evidence) LNN's are universal function approximators then also providing a comparison between NN's and LNN's on a number of benchmark problems .

5. Resource Requirements

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

Bibliography

[1] LAURENSON, J. Learning logical activations in neural networks, 2016.