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 employeed to solve a number of non-trivial problems. The downside to these networks is that they are a black box, increadibly diffcut (if not impossible) to understand how the LLN is operating. We aim to build on previous research showing that Logical Neural Networks applyed to the MINST dataset having comparable peformance to standard NN's and a more understandable network by formally proving a foundation for LLN's, investigating where LLN's are best applyed and whether we can approximate our LLN as a descrete logic circut.

1. Introduction

Neural Networks (NN's) peform exceptionally well over a wide range of different problems. However once trained these networks become a black box, near impossible for a human to understand what features the network is using to solve the problem presented to it. Logical Neural Networks (LLN's), that is NN's with logic based activation functions, have previously [REFERENCE PREV HONORS PROJECT] been shown to provide a more understandable representation while still peforming well (in the context of the MINST dataset). Despite the promise of LLN's not alot investigation into them has occoured.

2. The Problem

The problem with standard NN's is that once trained they are very diffcult (if not impossible) for a human to understand. LLN's have promise as an alternative to standard NN's with the added benifit of having sparse weights and being easier to understand. We suggest this problem still hasnt been completly solved as we have yet to develop a rigourous and formal foundation for LLN's. First we ask, can it be proven that like standard NN's, LLN's are also universal function approximators? Before we can hope to answer this key question we also must ask whether we have the right paramaterisation of noisy gates as these form the foundation of an LLN.

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

Finally we pose a concept which LLN's seem uniquicly quallified to solve given they are based of logic gates. Being able to create hardware NN's would be increadibley powerful (as hardware is alot faster). There dosnt seem to be any intuative way to achieve this with standard NN's, however approximating the continous logic gates in our LLN as descrete ones would make it possible. But is it possible to create an approximation in a feasable way without sacrifacing to much accuracy?

3. Proposed Solution

Research has shown that training an LLN with layers consisting of Noisy-OR and Noisy-And neurons can achieve a reasonable accuracy on the MINST dataset while also providing a more understandable network. To begin this project we take a step back to ensure we have the right approach. A network built out of neurons that resembel 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 fundimental step towards proving LLN'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. Consider the issue of representing some know boolean expression using descrete logic circut, but we only have access to AND and OR gates, AND combined with OR is not functionally complete. We propose to implement a number of differet sets of functionally complete nurons, such as NAND and NOR. These logical neurons would be paramaterised so we can observe the same nice

propertys we see with descrete gates, i.e. NOT(AND) = NAND. If an LLN 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 first 4 weeks of this project would be taken to fully understand the concept of Noisy gates, using the biblyography of the previous research and other reasources. Along with developing our paramaterisation of logical gates. The gates we aim to paramaterise and implement are AND, OR, NOT, NAND, NOR.

Our choice of gates have paramaterisations which matain convenient propertys and are functionally complete, so intuatively we would expect our LLN to be able to learn any known boolean function, the results here will be interesting as we might find that theroetcally our LLN can approximate any boolean function but in practice some can not be learnt so easily. As part of experements with learning boolean functions it would be interesting to experement with the training data given to the LLN, how can we train a network which generalises well. Say we trainied a LLN with one of the boolean inputs fixed at 0, then would the network be able to generalise 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 demontrainting that LLN's can approximate any know boolean function along with strong evidence showing which collections of neurons work well together, give the most understanble trained network and in what configuration yeields the best results.

Next we address a fundimental question, are LLN's universal function approximators? Working towards a proof or strong evidence showing this in ether direction we can start by experementing with an LLN learning some general functions and comparing its peformance to a standard NN. These experements are not equivelnt to a proof but will reveal which direction we should persue. At the conculsion of this part of the project we will answer this with a proof or strong evedience, we allocate 6 weeks for this section.

Now that we have explored the foundation of LLN's, using this knowledge we want to use our LLN to solve standard benchmark problems and compare agains peformance of standard NN's. We would use K-Fold Cross Validation, which is commonly used for comparasons like this. It would then be useful to examine the trained LLN and see if we can understand how the network has learned to represent the problem as this really is the motivation behind LLN's. We allocate 4 weeks for this.

Seeing we have esentially constructed NN's out of neurons which resemble continous logic gates, something to explore is can we feasabley approxamate these continous gates with descrete ones, esentially allowing us to create hardware NN's. Peforming such a task would centianly require more gates than neurons but would this blowup result in something that could feasibly be implemented on a circut board. We allocate any remaining time to investigating this problem and developing/implementing methods to solve it.

An overview of the proposed solution folows

- Produce bibliography and paramaterisation of gates (est: 4 weeks)
- Proving LLN's can/can not approximate any known boolean function (est: 6 weeks)
- Proving LLN's are/are not universal function approximators (est: 6 weeks)
- Evaluate LLN's on benchmark problems (est: 4 weeks)

- Explore approximating LLN as descrete logic circut (est: remaining time)

4. Evaluating your Solution

A solution to this problem should build the foundation to LLN's. Giving a proof that LLN's can approximate any known boolean function and strong evidence/proof showing that LLN's are universal function approximators. This would require an indepth investigation into various paramaterisations of logic gates. Folowing this it should identify where LLN's are most powerful and where a standard NN should be used insted. Finally a study demonstraiting ways to approximate an LLN as a logic circut, commenting on the feasability and accuracy of these methods.

5. Resource Requirements

I do not forsee any requirements I dont allready have access to.