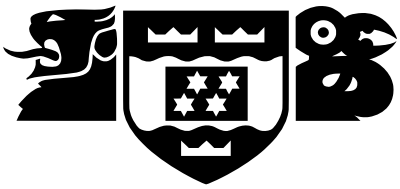
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui



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
Master of Computer Science.

Abstract

A short description of the project goes here.

Contents

Chapter 1

Introduction

Neural Networks (NN's) are commenly used to model supervised learning problems. A well trained NN can generalize well but it is very diffcult to intepret how the network is operating. This is called the black-box problem, many algorythms exsist for extracting rules from neural networks.

There are a number of motivations for wanting to extract rules from NN's. If a NN is able to provide an explination for its output by inspecting the reasoning a deeper understanding of the problem can be developed, the rules learnt by an NN could represent some knolwedge or patten in the data which has not yet been identifyed. Another possibility is that the neural network is being implemented to operate a critical systems which involve the saftey of humans, in this case being able to extract rules and inspect the NN is a necessary part of ensuring the system is safe.

Rule extraction algorythms are generally split into three categories. The **Decompositional Approach** extracts rules by analysing the activations and weights in the hidden layers. The **Pedagogical Approach** works by creating a mapping of the relationship beween inputs and outputs. Finally The **Eclectic Approach** combines the previous two approaches

This report develops a decompositional approach to rule extraction, learning the rules to be extracted happens automatically during the training phase. Once a network has been trained logical rules can be extracted from the hidden and output weights. By restricting the operations peformed by the neurons to be logical or's or and's it becomes possible to intepret the neurons as logical operations on their inputs.

Chapter 2

Background Survey

2.1 Logical Neural Networks

Logical Neural Networks (LNN's) are Neural Networks (NN's) which have activation functions that resemble boolean functions [2]. The two Logical Neurons developed in "Learning Logical Activations" are Noisy-AND and Noisy-OR Neurons.

Definition 2.1.1. A **Noisy-OR** Neuron has weights $\epsilon_1, ..., \epsilon_n$ which represent the irielevence of corosponding inputs $x_1, ..., x_n$. The activation of a Noisy-OR Neurons is.

$$a = 1 - \prod_{i=1}^{p} (\epsilon_i^{x_i}) \cdot \epsilon_b \tag{2.1}$$

Definition 2.1.2. A **Noisy-AND** Neuron has weights $\epsilon_1, ..., \epsilon_n$ which represent the irielevence of corosponding inputs $x_1, ..., x_n$. The activation of a Noisy-AND Neurons is.

$$a = \prod_{i=1}^{p} (\epsilon_i^{1-x_i}) \cdot \epsilon_b \tag{2.2}$$

2.2 Logical Normal Form Networks

"Backpropagation for Neural DNF- and CNF-Networks" presents a class of networks to overcome the black-box problem asociated with nerual networks. Each neuron represents either a disjunction (OR) or an conjunction (AND) of the inputs given to it. After training a network of this form if a low error is achieved then a boolean formula can be extracted from the network [1].

Definition 2.2.1. A **Conjunctive Normal Form (CNF) Network** consists of three layers. The hidden layer contains only Noisy-OR neurons and the output layer contains a single Noisy-AND.

Definition 2.2.2. A **Disjunctive Normal Form (DNF) Network** consists of three layers. The hidden layer contains only Noisy-AND neurons and the output layer contains a single Noisy-OR.

Definition 2.2.3. A **LNF Network** is either a DNF or CNF Network

The work presented in [1] lacks justification and explination for a number of key choices, consequently this report re derives the concept of LNF Networks insted using our Noisy-AND and Noisy-OR neurons.

2.3 Decompositional Methods for Rule Extraction

2.3.1 The KT Method

Chapter 3

Work Done

3.1 NAND Network

In descrete logic any boolean expression can be built using NAND gates. The concept of a continous NAND Neuron is developed with the idea that they could be used as units in a feedfoward network. Restricting the possible operations each neuron can peform will make the network more interpretable, it will allow inspection of the weights to see which inputs are being considered in the NAND operation.

3.1.1 Development of Paramaterisation

Definition 3.1.1. Let $x_1, ..., x_n$ be the inputs to a neuron n. Then n is a **NAND neuron** if the corosponding weights are $\epsilon_1, ..., \epsilon_n$. $0 \leq \epsilon_i \leq 1$ is the beleif that x_i does not effect the output. The activation of a NAND neuron is

$$a(X) = NOT(AND(OR(x_1, \epsilon_1), ..., OR(x_n, \epsilon_n))) \tag{3.1}$$

The folowing "Boolean Like" functions are the continous representations of NOT, AND and OR that will be used.

$$NOT(a) = 1 - x \tag{3.2}$$

$$AND(x_1, ..., x_n) = \prod_{i=1}^{n} x_i \tag{3.3}$$

$$OR(x_1, x_2) = x_1 + x_2 - x_1 x_2 \tag{3.4}$$

The activation 3.1 can now be re written interms of equations 3.2, 3.3, 3.4 as equation 3.5.

$$a(x) = 1 - \prod_{i=1}^{n} (x_i + \epsilon_i - x_i \epsilon_i) \tag{3.5}$$

3.1.2 Dicussion

Using this NAND Neuron paramaterisation networks of these gates are able to learn to be NOT, NAND, AND, OR (if the network has the right configuration). Attempting to learn the implys function demonstraits a fundimental issue with this approach. The implys function is given by $p \implies q \iff p \uparrow (q \uparrow q)$, given that these networks must be feedfoward this would require a NAND neuron to act as an identity which it cant do.

3.2 Logical Normal Form Networks

Assume the problem we are trying to learn has some boolean formula which underpins it. Then it is known that this formula must have a Conjunctive Normal Form (CNF) or Disjunctive Normal Form (DNF). This section explores the posibility of designing networks which learn the CNF or DNF representation of any underlying boolean expression.

The paper "Backpropagation for Neural DNF and CNF Networks" [1] proposes networks which can solve this task, however the paper does not provide justification and consequently is diffcult to reproduce. This report takes this concept but rederives it using the Noisy-OR and Noisy-AND gates [2].

A CNF or DNF formula contains clauses of literals which is either an atom or a negation of an atom. To account for this the number of inputs to the network will be doubled, i.e. the inputs will be all the atoms and negations of thoughs atoms. 2^n is an upper bound on the number of Noisy gates needed to learn any boolean expression of n inputs. To ensure that an LNF Network can learn a boolean expression the number of Noisy gates in the hidden layer will be 2^n.

3.2.1 Noisy Gate Paramaterisation

The paramaterisation of Noisy gates require weight clipping, an expensive operation, to avoid manual clipping a new paramaterisation is derived that implisitley clips the weights. Consider that $\epsilon \in (0, 1]$, therefore let $\epsilon_i = \sigma(w_i)$, these w_i's can be trained without any clipping, after training the origonal ϵ_i's can be recovered.

Now these weights must be substutited into the Noisy activation. Consider the Noisy-OR activation.

$$
\begin{aligned}
a(X) &= 1 - \prod_{i=1}^{p} (\epsilon_i^{x_i}) \cdot \epsilon_b \\
&= 1 - \prod_{i=1}^{p} (\sigma(w_i)^{x_i}) \cdot \sigma(b) \\
&= 1 - \prod_{i=1}^{p} ((\frac{1}{1+e^{-w_i}})^{x_i}) \cdot \frac{1}{1+e^{-b}} \\
&= 1 - \prod_{i=1}^{p} ((1+e^{-w_i})^{-x_i}) \cdot (1+e^{-w_i})^{-1} \\
&= 1 - e^{\sum_{i=1}^{p} ln(1+e^{-w_i}) + ln(1+e^{-b})} \\
&Let\ w_i^{'} = ln(1+e^{-w_i}),\ b^{'} = ln(1+e^{-b}) \\
&= 1 - e^{-(W^{'} \cdot X + b^{'})}
\end{aligned}
$$

From a similar derivation we get the activation for a Noisy-AND, concisely giving equations 3.6, 3.7

$$a_{and}(X) = e^{W^{'} \cdot (1-X) + b^{'}} \tag{3.6}$$

$$a_{or}(X) = 1 - e^{-(W^{'} \cdot X + b^{'})} \tag{3.7}$$

The function taking w_i to $w_i^{'}$ is the soft ReLU function which is peforming a soft clipping on the w_i's.

3.2.2 Training LNF Networks

Using equations 3.7 and 3.6 for the Noisy-OR, Noisy-AND activations retrespectively allows the networks to be trained without expliscit clipping. The ADAM Optimizer is used for training firstly for the convenience of an adaptive learning rate but also because it inclueds the advantages of RMSProp which works well with on-line (single-example-training) learning, which these LNF Networks respond well to.

Preliminary testing showed that LNF Networks are able to learn good classifiers on boolean gates, i.e. NOT, AND, NOR, NAND, XOR and Implies. It is also possible to inspect the trained weights and see that the networks have learnt the correct CNF or DNF representation.

3.2.3 LNF Network Peformance

How do LNF Networks peform against standard perceptron networks which we know to be universal function approximators. Two different perceptron networks will be used as a benchmark, one with the same configuration as the LNF Network, the other with less hidden neurons. The testing will consist of selecting 5 random boolean expressions for $2 \leq n \leq 9$ and training each network 5 times, each with random initial conditions. Figure 3.1 shows a comparason between all 4 of the networks and figure 3.2 shows just the LNF Networks.

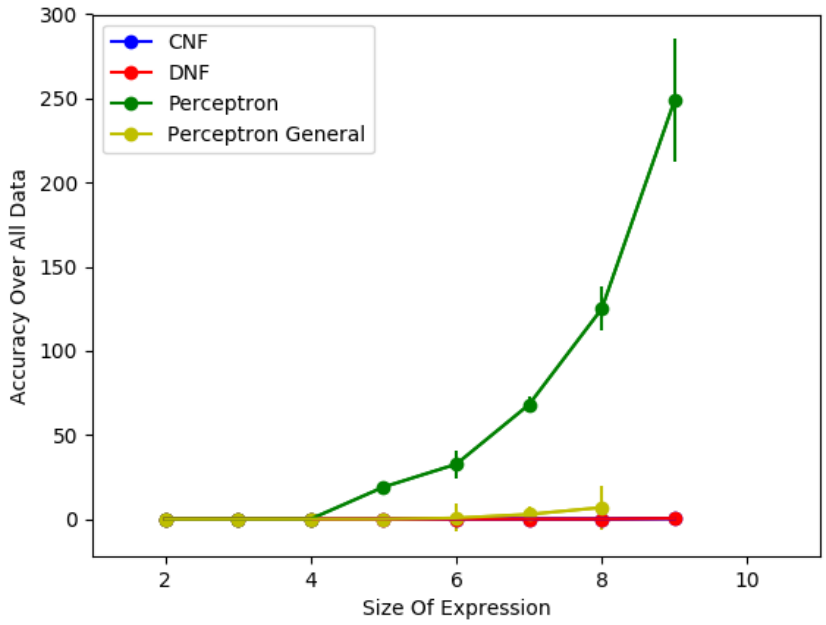


Figure 3.1

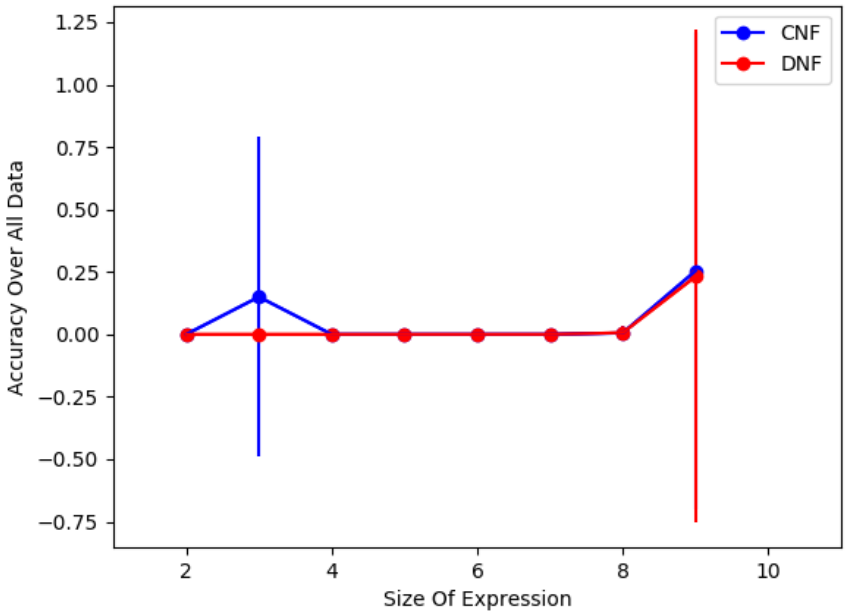


Figure 3.2

Figure 3.1 shows that neither of the perceptron networks peform as well as the LNF Networks as n increases. Figure 3.2 shows on avarage there are no stastically significant differences between the CNF or DNF networks. What is not present in 3.2 is that at $n = 9$ sometimes the CNF network far out peforms the DNF and visa versa, theoretically both should beable to learn any boolean expression.

3.2.4 LNF Network Rule Extraction

The goal of this report is to make neural networks more intepretable, for LNF Networks to addres this problem there must be a way to extract rules from them. Take the weights of a trained LNF Network, these weights can be converted back into ϵ_i's by apply the sigmoid function to each w_i.

As $\epsilon_i \to 0$ then x_i becomes relevent and as $\epsilon_i \to 1$ then x_i becomes irrielevent. If the network has learnt the correct DNF or CNF representation the for every neuron if input i is relevent then $w_i \to -\infty$ and therefore $\epsilon_i \to 0$, othwerwise x_i is irelevent and $w_i \to \infty$ meaning $\epsilon_i \to 1$.

Consequently in ϵ form the network is binary and rules can be easily extracted. Many of the formulas extracted contain redundent terms, i.e. clauses that are a tautologie or a duplicate of another, filtering these out is not an expensive operation.

When training an LNF network over the entire truth table for a boolean expression, when a low error is achieved it is possible to extract boolean formula from the network which gets can generate the origonal truth table. This is a necessary first step but a more important question is, can formula still be extracted from the network when the LNF network is not trained with the entire truth table?

3.2.5 LNF Network Generalization

These networks are able to peform as well as standard perceptron networks but so far they have been getting the complete set of data, in practice this will almost never be the case. Perceptron networks are so widely used because of their ability to generalize, for LNF Networks to be useful they must also be able to generilize.

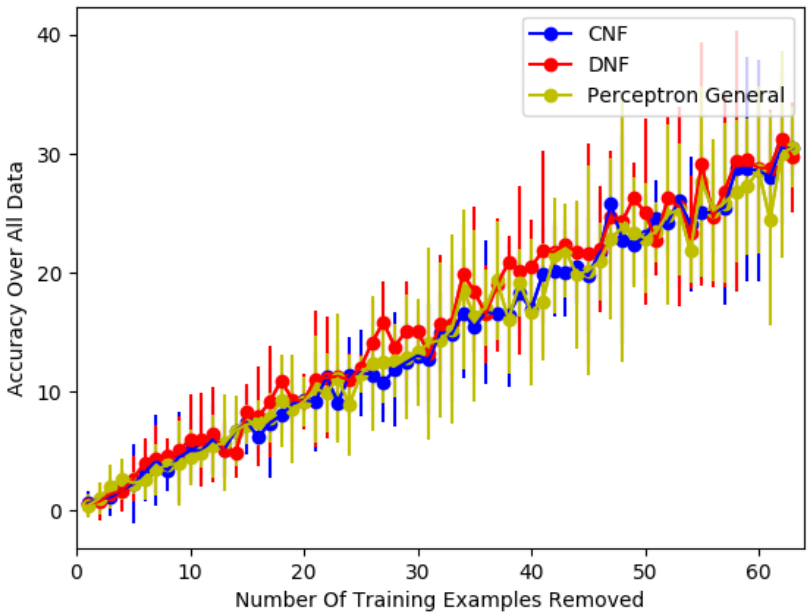


Figure 3.3

Figure **??** shows a comparason between the generalization ability of CNF, DNF and Perceptron networks. The graph shows the peformance over all training data when successively removing elements from the training set. It demonstraits that the CNF and DNF networks generalize as well as the perceptron networks when the boolean formula has 6 inputs.

Rule Extraction

Chapter 4

Future Plan

Investigate peformance, generalization and intepretability on some simple benchmark problems, both descrete and continous. The continous case will require an investigation of effective methods to descritize contnous inputs.

Chapter 5

Request For Feedback

Bibliography

[1] HERRMANN, C., AND THIER, A. Backpropagation for neural dnf-and cnf-networks. *Knowledge Representation in Neural Networks, S* (1996), 63–72.

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