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

1 Introduction **1**

2 Proposal Review **3**

3 Background Survey **5**
3.1 Core Concepts 5
3.2 Litrature Review 5

4 Logical Normal Form Networks **9**
4.1 Noisy Gate Paramaterisation 9
4.2 Training LNF Networks 10
4.3 LNF Network Peformance 10
4.4 LNF Network Rule Extraction 12
4.5 LNF Network Generalization 12
4.5.1 Rule Extraction 13

5 Future Plan **15**

6 Request For Feedback **17**

Chapter 1

Introduction

Neural Networks (NN's) are commenly used to model supervised learning problems. NN's often achieve higher accuracy than other methods because they are able to approximate any continous function. 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. Rule extraction algorythms have been developed to address this problem, some aim to extract rules to replace the NN and others extract rules which are combined with the NN to improve peformance.

There are a number of motivations for wanting to solve the black-box problem. 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.

By restricting the function set that each neuron can peform is it possible to create a more intepretable network? Restricted the functions for each neuron to be the function set taking some subset of inputs and peform a pre determined logical function, after training to identify the function each neuron is peforming on its inputs only the subset of inputs considered must be identified as the operation is fixed.

This report develops a decompositional approach to rule extraction which allows boolean expressions to be infered from a network with little effort once training has finished.

Neurons are restricted to peforming only AND's or OR's on some subset of their inputs, then placed in a confguration which allows learning Conjunctive Normal Form or Disjunctive Normal Form expressions, such networks are called Logical Normal Form (LNF) Networks.

When provided a truth table for a boolean expression, the peformance of LNFN's has no stistically significant differnces to that of a Multi-Layer Perceptron (MLPN) Network. The LNFN's are also able to generalize, obtaining stistically equal peformances as a MLPN when given incomplete truth tables.

Once trained an LNFN's weights directly corospond to the subset of inputs being acted on. A rule extraction algorythm can be defined which simply inspects the weights of each neuron to extract a boolean representing the network.

The restriction placed on the function space of each neuron, while improving the interpretability, intuatively will also hinder their ability to be universal approximators. Along with the development of a new rule extraction approach this report will identify and explore the issues introduced by the restrction placed on neurons to determin where the rule extraction algorythm can be used.

Chapter 2

Proposal Review

Origonally proposed was to restrict the function set of neurons to the NAND ($\uparrow$) operations, and construct feedfoward networks with these restricted neurons, however this idea is flawed. Consider the expression p implys q, $p \implies q \iff p \uparrow (q \uparrow q)$.

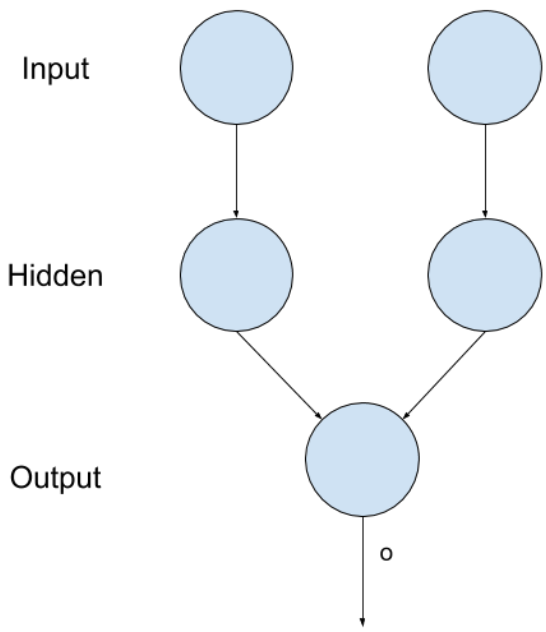

Figure 2.1

Figure 2.1 demonstraits the structure a feedfoward network attempting to represent implys would look like. For o to be the output required then one node in the hidden layer would have to act as an identiy, passing the input p to the ouput neuron. NAND cant act as an identity.

Chapter 3

Background Survey

3.1 Core Concepts

A boolean formula is in Conjunctive Normal Form (CNF) if and only if it is a conjunction (and) of clauses. A clause in a CNF formula is given by a disjunction (or) of literals. A literal is either an atom or the negation of an atom, an atom is one of the varables in the formula.

Consider the boolean formula $\neg a \vee (b \wedge c)$, the CNF is $(\neg a \vee b) \wedge (\neg a \vee c)$. In this CNF formula the clauses are $(\neg a \vee b)$, $(\neg a \vee c)$, the literals used are $\neg a$, b, c and the atoms are a, b, c.

A boolean formula is in Disjunctive Normal Form (DNF) if and only if it is a disjunction (or) of clauses. A DNF clause is a conjunction (and) of literals. Literals and atoms are defined the same as in CNF formulas.

Consider the boolean formula $\neg a \wedge (b \vee c)$, the DNF is $(\neg a \wedge b) \vee (\neg a \wedge c)$.

3.2 Litrature Review

A survey in 1995 focuses on rule extraction algorythms [1], identifying the reasons for needing these algorythms and introducing ways to categorise and compare them. Motivation behind scientific study is always crutial so why is understanding the knowledge contained inside Artificial Neural Networks's (ANN's) important? The key points indentified are that the ANN might of discovered some rule or patten in the data which is currently not known, being able to extract these rules would give humans a greater understanding of the problem. Another, prehapse more significant reason is the application of ANN's to systems which can effect the safty of human lives, i.e. Aroplanes, Cars. If using an ANN in the context of a system inboling human safty it is important to be certian of the knowledge inside the network, to be sure that the ANN wont take any dangourous actions.

There are three categories that rule extraction algorythms fall into [1]. An algorythm in the **decompositional** category focuses on extracting rules from each hidden/output unit. If an algorythm is in the **pedagogical** category then rule extraction is thought of as a learning process, the ANN is treated as a black box and the algorythm learns a relationship between the input and output vectors. The third category, **electic**, is a combiniation of decompositional and pedagogical. Electic accounts for algorythms which inspect the hidden/output neurons individually but extracts rules which represent the ANN globally [5].

To further divide the categories two more classifications are introduced. One mesures the portability of rule extraction technequics, i.e. how easily can they be applyed to different types of ANN's. The second is criteria to assess the quallity of the extracted rules, these are accuracy, fidelity, consistency, comprehensibility [1].

1. A rule set is **Accurate** if it can generalize, i.e. classify previously unseen examples.

2. The behavour of a rule set with a high **fedelity** is close to that of the ANN it was extracted from.

3. A rule set is **consistent** if when trained under different conditions it generates rules which assign the same classifications to unseen examples.

4. The mesure of **comprehensibility** is defined by the numer of rules in the set and the number of literals per rule.

The paper "Backpropagation for Neural DNF- and CNF-Networks" presents an approach which relies on a special NN archetechure. The neurons in these networks have a restricted functon space, they are only able to peform a OR or AND on a subset of their inputs. By restricting the degreese of freedom in the network it is possible to understand the actions each neuron is taking. Rules can simply be extracted from the trained network by inspecting these neurons [2]. The conjunctive and disjunctive neurons presented, while making sense mathematically are combersom to implement. Insted a perfered way to implement logical neurons will be to use Noisy-OR and Noisy-AND neurons [3]. Noisy gates are derived from the Noisy-OR relation, developed by Judea Pearl [4], a concept in Bayesian Networks.

A Bayesian Network represents the conditional dependencies between random variables in the form of a directed acyclic graph.

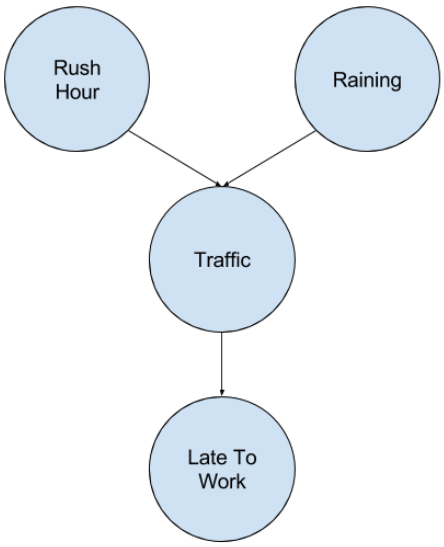

Figure 3.1

Figure 3.1 is a Bayesian network, it demonstraits the dependency bettwen the random variables "Rush Hour", "Raining", "Trafic", "Late To Work". The connections show the dependencys i.e. Traffic influences whether you are late to work, and it being rush hour or raining influences whether there is trafic.

Consider a bayesian network, take some node D with $S_1, ..., S_n$ as parents where each S_i is independent from all others, i.e. each S_i influences the node D. The relationship between D and its parents is if S_1 OR ... OR S_n is true then D is true. Let ϵ_i be the uncertanty that S_i infludence D then $P(D = 1|S_1, , S_n)$ can be defined.

$$P(D = 1|S_1, ..., S_n) = 1 - \prod_{i=1}^{n} \epsilon_i \tag{3.1}$$

In the context of a neuron, the inputs $x_1, ..., x_n$ represent the probability that inputs $1, ..., n$ are on. Each ϵ_i is the uncertanty as to whether x_i infulences the output of the neuron. The definitions for Noisy-OR and Noisy-AND gates are given below

Definition 3.2.1. A **Noisy-OR** Neuron has weights $\epsilon_1, ..., \epsilon_n \in [0, 1]$ which represent the irielevence of corosponding inputs $x_1, ..., x_n \in [0, 1]$. The activation of a Noisy-OR Neurons is.

$$a = 1 - \prod_{i=1}^{p} (\epsilon_i^{x_i}) \cdot \epsilon_b \tag{3.2}$$

Definition 3.2.2. A **Noisy-AND** Neuron has weights $\epsilon_1, ..., \epsilon_n \in [0, 1]$ which represent the irielevence of corosponding inputs $x_1, ..., x_n \in [0, 1]$. The activation of a Noisy-AND Neurons is.

$$a = \prod_{i=1}^{p} (\epsilon_i^{1-x_i}) \cdot \epsilon_b \tag{3.3}$$

Both these paramatersations reduce to descrete logic gates when there is no noise, i.e. $\epsilon_i = 0$ for all i.

While the concept presented in [2] is the foudation for the work presented in this report, the approach presented is different that what has been done. A different approach to disjunctive and conjunctive neurons is taken, along with this more investigation is carried out in terms of the capabilitys of these networks (when compared to a standard perceptron) and the rule extraction method.

Chapter 4

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 networks which learn the CNF or DNF representation of any underlying boolean expression.

The paper ”Backpropagation for Neural DNF and CNF Networks” [2] proposes networks which can solve this task, however the paper does not provide justification, consequently it is diffcult to understand and reproduce. This report takes this concept but rederives it using the Noisy-OR and Noisy-AND gates [3].

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.

4.1 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 4.1, 4.2

$$a_{and}(X) = e^{W^{'} \cdot (1-X) + b^{'}} \tag{4.1}$$

$$a_{or}(X) = 1 - e^{-(W^{'} \cdot X + b^{'})} \tag{4.2}$$

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

4.2 Training LNF Networks

Using equations 4.2 and 4.1 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.

4.3 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 4.1 shows a comparason between all 4 of the networks and figure 4.2 shows just the LNF Networks.

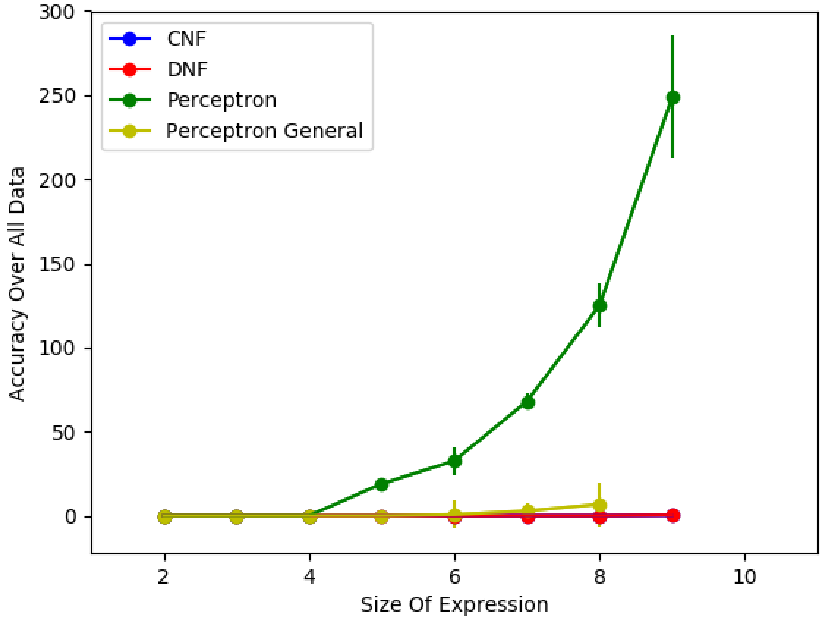

Figure 4.1

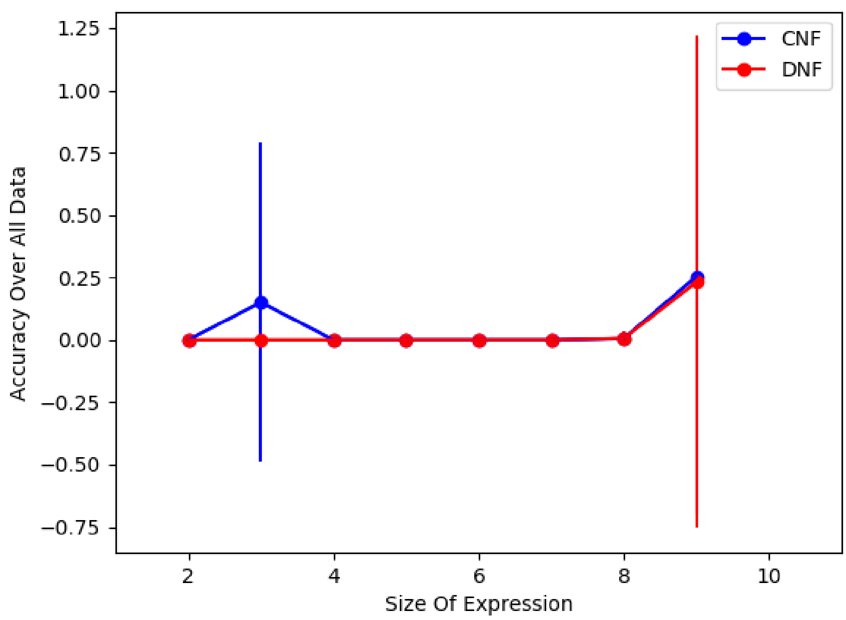

Figure 4.2

Figure 4.1 shows that neither of the perceptron networks peform as well as the LNF Networks as n increases. Figure 4.2 shows on avarage there are no stastically significant differences between the CNF or DNF networks. What is not present in 4.2 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.

4.4 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?

4.5 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.

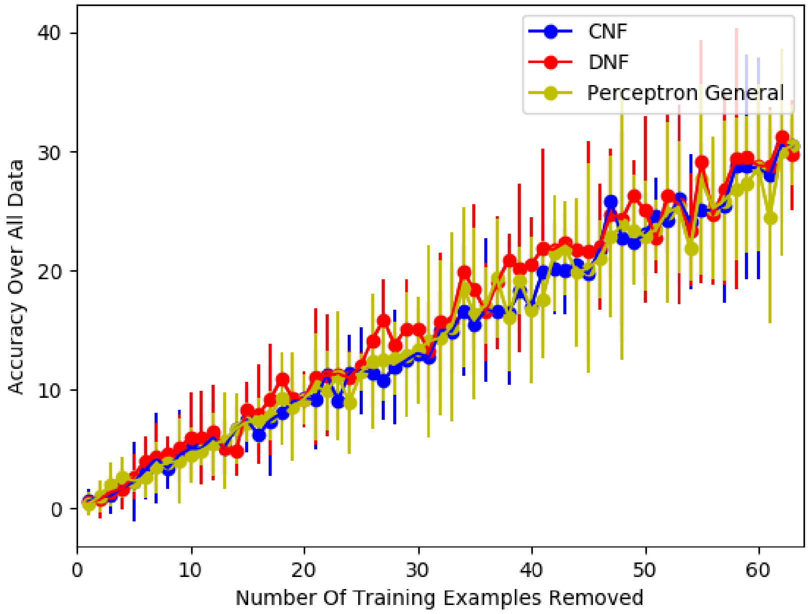

Figure 4.3

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.

4.5.1 Rule Extraction

Chapter 5

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 6

Request For Feedback

Bibliography

[1] ANDREWS, R., DIEDERICH, J., AND TICKLE, A. B. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems 8*, 6 (1995), 373–389.

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

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

[4] RUSSELL, S., NORVIG, P., AND INTELLIGENCE, A. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25* (1995), 27.

[5] TICKLE, A. B., ANDREWS, R., GOLEA, M., AND DIEDERICH, J. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks 9*, 6 (1998), 1057–1068.