Logical Neural Networks: Opening the black box
Neuron Paramaterisations

Daniel Braithwaite

April 13, 2017

1 Continous Logic Based Paramaterisation

1.1 Background

Based on the idea of continous logic (CL) [1] we can develop paramaterisations for various CL gates. First some background on CL. We define our CL on a closed interval $C = [A, B]$ i.e. if $a \in C \iff -1 \leq a \leq 1$. Let $M = \frac{(A+B)}{2}$. We define basic operations on CL as

$$a \vee b = max(a, b) \tag{1}$$
$$a \wedge b = min(a, b) \tag{2}$$
$$\bar{a} = 2M - a \tag{3}$$

We will omit the $\wedge$ from now on. The folowing are some of the laws of CL. The laws given here are the ones we find most useful. As we see these corospond to the laws of descrete logic (DL).

$$a \vee b = a, aa = a \tag{4}$$
$$a \vee b = b \vee a, ab = ba \tag{5}$$
$$(a \vee b) \vee c = a \vee (b \vee c), (ab)c = a(bc) \tag{6}$$
$$a(b \vee c) = ab \vee ac, a \vee bc = (a \vee b)(a \vee c) \tag{7}$$
$$a \bar{\vee} b = \bar{a}\bar{b}, \bar{ab} = \bar{(a)} \vee \bar{b} \tag{8}$$
$$\bar{\bar{a}} = a \tag{9}$$

1.2 Differentation In CL

We can use the folowing two rules in CL

$$\frac{\partial}{\partial x_1}\Big[(x_1 \vee x_2)\Big] = (x_1 - x_2) \tag{10}$$

$$\frac{\partial}{\partial x_1}\Big[(x_1 \wedge x_2)\Big] = (x_2 - x_1) \tag{11}$$

1.3 NAND Is Functionally Complete

Similarly to DL we can prove that NAND is functionally complete in CL.

Theorem 1 (NAND (↑) is Functionally Complete)**.** *The NAND gate, defined as NOT(AND) can be used to represent any logical expression in CL*

Proof. **NOTE: This is simply a reiteration of the proof in DL as the laws are the same** It is sufficient to show we can represent NOT, AND, OR using NAND. First we consider NOT

$$\bar{a} = \bar{a} \vee \bar{a} = \overline{a \wedge a} = a \uparrow a$$

Now we show this is true for AND

$$a \wedge b = \overline{(\overline{a \wedge b})} = (\overline{a \uparrow b}) \uparrow (\overline{a \uparrow b})$$

Finally consider the case of OR

$$a \vee b = \overline{\overline{a} \wedge \overline{b}} = \overline{(a \uparrow a) \wedge (b \uparrow b)} = (a \uparrow a) \uparrow (b \uparrow b)$$

All operations used are laws of CL, so we have proved what was required and thus NAND is functionally complete. □

1.4 Paramaterisation

We will consider a paramaterisation of a NAND gate. First we define our CL as folows, C = [-1, 1], we think of -1 as being as false as possible and 1 as being as positive as posible, giving us M = 0. We have n inputs to our gate, there value denoted $x_1, ..., x_n$, and our weights are defined as $\epsilon_1, ..., \epsilon_n$. We think of each input x_i as being the probability that node i is on and we consider ϵ_i to be the probality that the input x_i is relevent to our gate.

So how do we link our ϵ_i to x_i. Take some μ_i to be the input from x_i transformed according to ϵ_i, therefor $\mu_i = f(x_i, \epsilon_i)$. First lets consider the descrete case, So if $\epsilon_i = 1$ then we consider input x_i irrelevent to our gate, in the case of NAND this means that we want μ_i to be true, on the otherhand if $\epsilon_i = -1$ then input x_i is relevent and we want $\mu_i = x_i$. Therefor we say that $\mu_i = OR(x_i, \epsilon_i) = max(x_i, \epsilon_i)$

So the output from our NAND gate is given by $y = NOT(AND(\mu_1, ..., \mu_n) = -min(\mu_1, ..., \mu_n)$. So setting all ϵ_i's to -1 would give us a triditional NAND gate in CL

1.4.1 Comparason to DL

We want to see how this connects to NAND in the DL case, clearly if we have a CL defined over [0,1] we can see the connection. Consider a CL over [0,1] (Note that in this case $\bar{a} = 1 - a$), the table below represents a NAND gate with two inputs

x_1	x_2	$x_1 \uparrow x_2$	$1 - min(x_1, x_2)$
0	0	1	1 - 0 = 1
0	1	1	1 - 0 = 1
1	0	1	1 - 0 = 1
1	1	0	1 - 1 = 0

We see this clearly represents a NAND gate, now we only need connect our CL over [-1, 1] to the CL over [0, 1]. We can define a bijection $f : [-1, 1] \longrightarrow [0, 1]$, take $f(x) = \frac{(x+1)}{2}$

Theorem 2. *$f(x) = \frac{(x+1)}{2}$ is a bijection*

Proof. Say we have $f(x_1) = f(x_2)$, then we see the folowing

$$\begin{aligned} f(x_1) &= f(x_2) \\ \frac{(x_1 + 1)}{2} &= \frac{(x_2 + 1)}{2} \\ x_1 + 1 &= x_2 + 1 \\ x_1 &= x_2 \end{aligned}$$

Therefor f is one-to-one. Now we only need show f is onto, consider the folowing

$$\begin{aligned} y &= \frac{x + 1}{2} \\ 2y &= x + 1 \\ 2y - 1 &= x \end{aligned}$$

Therefor x is also onto and therefor is a bijection. □

Now we can connect our NAND in a CL over [-1, 1] to a NAND in DL.

1.4.2 Deriving Backpropagation Gradients

NOTE: This is the origonal attempt at deriving gradients, before I realised I was differentatiing the logical functions incorrectly We want to find the folowing quantiy

$$\frac{\partial x_j}{\partial w_{k,j}} = \frac{\partial E}{\partial y_i} \frac{\partial d_i}{\partial x_j} \frac{\partial x_j}{\partial w_{k,j}} \tag{12}$$

Which involves solving for the individual components of this

$$\begin{aligned}\frac{\partial x_j}{\partial w_{k,j}} &= \frac{\partial}{\partial w_{k,j}}\Big[x_j\Big] \\ &= \frac{\partial}{\partial w_{k,j}}\Big[-\mu_{j,a_j}\Big] \\ &= \frac{\partial}{\partial w_{k,j}}\Big[-max(y_{a_j}, w_{a_j,j})\Big] \\ &= \begin{cases} 0 & k \neq a \ or \ y_a > w_{a_j,j} \\ -1 & otherwise \end{cases}\end{aligned}$$

Here we define the folowing function to simplyfy things

$$C(k,j) = \begin{cases} 0 & k \neq a \ or \ y_a > w_{a_j,j} \\ -1 & otherwise \end{cases}$$

We consider this function as being 0 if the weight $w_{k,j}$ had no effect on the output of neuron j, otherwise this function is 1. Making

$$\frac{\partial x_j}{\partial w_{k,j}} = C(k,j) \tag{13}$$

Now we derive

$$\begin{aligned}\frac{\partial y_j}{\partial x_j l} &= \frac{\partial}{\partial x_j}\Big[x_j\Big] \\ &= 1\end{aligned}$$

Now when deriving $\frac{\partial E}{\partial y_j}$ we must account for this quantity being different for when j is an output node or a node in the hidden layer. First we consider the output layer

$$\frac{\partial E}{\partial y_j} = -(t_j - -y_j) \tag{14}$$

Giving us the folowing final quantify for change in weight when **node j is an output node**

$$\frac{\partial E}{\partial w_{k,j}} = -C(k,j)(t_j - y_j) \tag{15}$$

Now we consider the more complicated case of node j being a hidden layer

$$\begin{aligned}\frac{\partial E}{\partial y_j} &= \sum_{i \in I_j} \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_i}\frac{\partial x_i}{\partial y_j} \\ &= \sum_{i \in I_j} \frac{\partial E}{\partial y_i}\frac{\partial x_i}{\partial y_j}\end{aligned}$$

We see that

$$\frac{\partial x_i}{\partial y_i} = C(i,j)$$

Giving us the folowing final quantity for change in weights **when j is a hidden node**

$$\frac{\partial E}{\partial w_{k,j}} = C(k,j)\Big[\sum_{i \in I_j} C(j,i)\frac{\partial E}{\partial y_i}\Big] \tag{16}$$

1.4.3 Experemtal Tests

A single NAND neuron is able to learn to be a NAND and NOT, this makes sense as a single nuron is able to represent these. We run in to trouble when trying to learn AND and OR as these require more than one NAND.

References

[1] Levin, V. I. Basic concepts of continous logic. *Studies in Logic, Grammar and Rhetoric 11*, 24 (2007), 67–84.