Logical Neural Networks: Opening the black box

Logical Normal Form Network

Daniel Braithwaite

May 10, 2017

Given a simple feed foward network using neurons resembleing NAND gates is unable to learn any given boolean formula we switch our interest to something else. We know that any boolean expression can be represented in Conjunctive Normal Form (CNF) or Disjunctive Normal Form (DNF). So it seems prudent to ask wether its possible to construct a feedfoward netural netowrk which can learn these underlying representations. Findings [1] seem to sugest that it is possible however there is limited justification for some claims and because of this is diffcult to reproduce. We will take this general concept and reproduce the research in an attempt to develop a better understanding.

1 Noisy Neurons

During a previous investigation of Logical Neural Networks the concept of Noisy-OR and Noisy-AND neurons where derived [2]. Here we will simply state them as.

Definition 1.1. A **Noisy-OR** neuron is a perceptron with activation $a = 1 - e^{-z}$ where W is our weights, X is our inputs, b is our byas term and $z = WX + b$. We contrain each $w_i \in W$ and b to be in the interval $[0, \text{inf}]$

Definition 1.2. A **Noisy-OR** neuron is a perceptron with activation $a = e^{-z}$ where W is our weights, X is our inputs, b is our byas term and $z = WX + b$. We contrain each $w_i \in W$ and b to be in the interval $[0, \text{inf}]$

2 Logical Normal Form Networks

A Logical Normal Form Network is a neural net which stasfys one of the folowing definitions

Definition 2.1. CNF-Network A **CNF-Network** is a three layer network where neurons in the hidden layer consist soley of Noisy-OR's and the output layer is a single Noisy-AND.

Definition 2.2. DNF-Network A **DNF-Network** is a three layer network where neurons in the hidden layer consist soley of Noisy-AND's and the output layer is a single Noisy-OR.

It is worth noteing that CNF and DNF form can have the nots of atoms in there clauses, a simple way to account for this is to double the number of inputs where one represents the atom and the next represents the negation of that atom.

2.1 Learnability Of Boolean Gates

Theoretiaclly it makes sense for these networks to be able to learn the CNF and DNF representations of various boolean expressions, however before we attempt arbatary boolean functions we would like to start with something simple, namely expressions such as NOT, AND, NOR, NAND, XOR and IMPLIES. Results of which where promising, we where able to achieve a low error and from inspecting the weights we could see that the networks where infact learning the correct CNF and DNF representations.

2.2 Learnability Of Interesting Expressions

We now wish to see if we can use these networks to learn more interesting boolean formula, starting with expressions of 3 variables. While we are able to achieve a small error there is now some noise (i.e. small non zero weights for inputs that are irrelevant). While a small ammount of noise is okay and can be pruned out after training we could run into issues if the ammount of noise increases as the number of inputs does.

One other interesting observation to be made is that both the CNF and DNF Netoworks have trouble learning the boolean expression $(a\ XOR\ b)\ AND\ c$. They can't achevice an error lower than 1. Indivually we can learn an XOR gate and an AND gate but somehow combining the two results in something which is unlearnable.

During the investigation of this a more sinister issue was uncovered, namely that these LNF netowrks are unable to learn boolean expressions of 2 variables when given 3. I.e. we give the network all values for three inputs, a, b and c but we only want to learn $a\ OR\ b$. This is a fundimental issue as in practice most problems will be made up of boolean expressions which dont rely on all inputs.

It turns out this problem was not caused by a problem with our LNF Networks but with the weight initilizations, namely they where currently initilised to 0, changing the weights to be randomly distrubuted over the interval [0,1] fixed this problem and allows the LNF Networks to solve all attempted problems so far, along with there weight representations being intepretable.

References

[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.