Symbolic Manipulation and Computation in the Same Graph

Darius Barbano

Contents

Abstract

General artificial intelligence refers to machine intelligence than performs a task as successfully as a human does. A fundamental difference between human neural network and current machine neural networks is that only human networks combine symbolic reasoning with computation.

1 Background

Neural networks have been used for decades to solve a wide variety of machine learning tasks without the need for explicit programming of the solutions to be done by humans. Typically the networks only make use of arithmetic computation, meaning that logical operations are not incorportated into the models. This hasn't been done because there is no clear method by which logical information (TRUE/FALSE) should be converted to numerical information, and vice versa. In Computer Science, True and False are generally represented as 1 and 0, respectively, whereas in Mathematics, they are often represented as -1 and 1 Which specific fields of Math and Compsci?. Our research aims to construct a neural network which effectively combines logical and arithmetic computation, and apply this network to a problem which demostrates it's ability to perform symbolic reasoning.

1. What is symbolic computation?

   A symbolic computation is a calculation performed with symbolic representations of values and operations. A simple example would be the expression $(x+1)(x-1)$ which would evaluate to x^2-1, rather than to some numerical result.

2. What is calculation?

   A calculation is a process by which one or more inputs is transformed into one or more results. One may calculate that the product of 5 and 4 is 20.

3. What is meant by a computational class? - do you mean complexity class?

Code for each subsection of this document can be found at NeuralNetworkResearch.

2 Methods

2.1 Network Construction

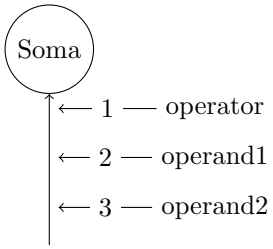


Figure 1: Single neuron receiving ordered intputs.

While this model is suitable for biological networks of neurons, a neural network in the computational sense would look more like an abstract syntax tree. In the following diagram, a network of operators and operands form a tree-like structure in which elements at lower levels represent values to manipulate which at higher levels interact with each other through arithmetic operations to produce a result. Any computation which involves a combination of operations on one or more values can be represented in this structure, such as logical or arithmetic operations.

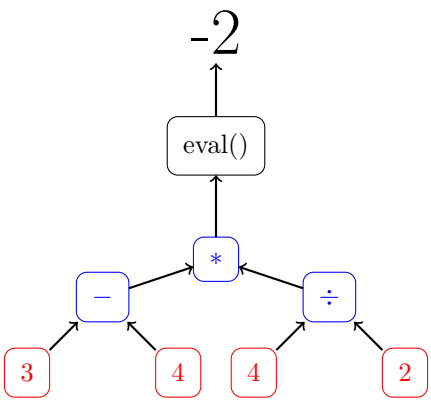


Figure 2: An Abstract Syntax Tree of arithmetic operations.

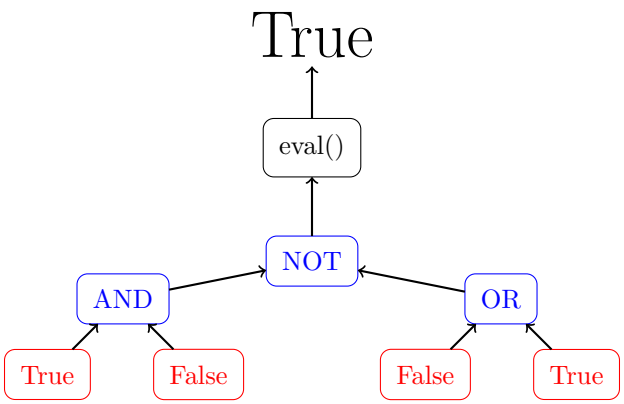


Figure 3: An Abstract Syntax Tree of logical operations.

2.2 Using each arithmetic operator with a logical operand

Code that corresponds with this section can be found in NeuralNetwork_5.py

Syntax trees such as the ones shown above work perfectly, since each contain exclusively arithmetic or logical computations. In the trees shown below, however, the two computation classes are combined through multiplication and addition. The tree's results depend entirely on how we choose to evaluate $2 * True$ and $2 + True$.

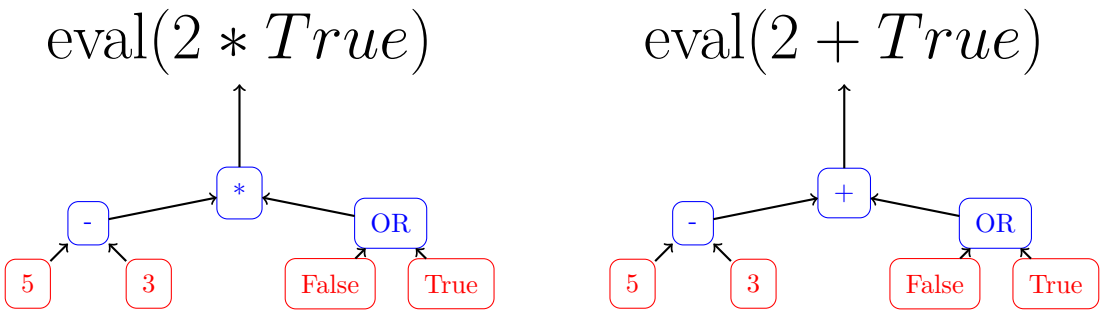
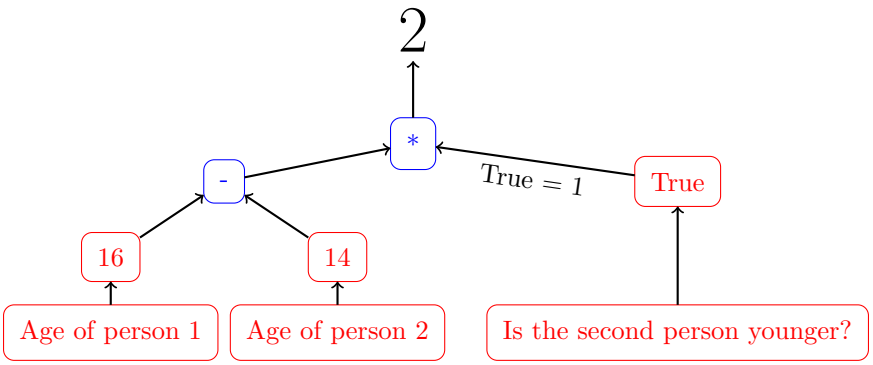
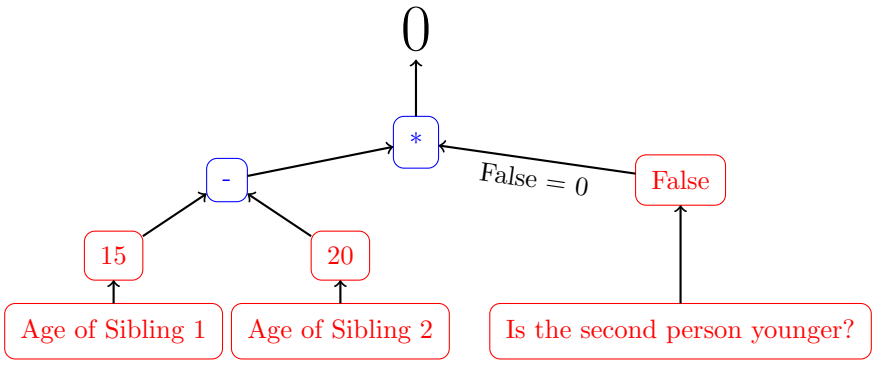


Figure 4: Abstract Syntax Trees of logical and arithmetic operations.

An error would be the result if the two expressions were to be evaluated since $True$ and $False$ are not numerical values so they aren't compatible with numerical operations like $*$ and $+$. Now the question arises, *How can logical symbols be converted into numerical values?* In the context of Computer Science and Programming, one may represent $False$ as 0 and $True$ as 1. In the case of multiplication, this format would cause $eval()$ to yield 0 if one of the operands is $False$ and whatever the other operand is if one is $True$. Let us now model a real-world situation in which logical and arithmetic computation are both necessary for a result.



(a) First sibling is younger



(b) Second sibling is younger

Figure 5: A syntax tree which models how many years one person is older than another, using a combination of Arithmetic and Logical computation.

In the above trees, logical and numerical information are effectively combined to produce an output. If only numerical information were used (i.e. the ages of the two people), only the difference of the two numbers would be calculated, yielding -5 in the second case. However, a person cannot be a negative number of years older than another, so 0 would be a more appropriate result. The above trees solve this problem by requiring the user to give a boolean value representing whether or not the second person is younger than the first, then the $True/False$ value is converted into a number according to previously defined policy and multiplied with the difference of the two ages.

We have just successfully formulated a method by which logical symbols and numbers can be combined through the multiplication operator, but how would the two be combined through the subtraction, division, and addition operators?

The process of multiplication can be broken down into a series of repeated additions. For example, $4*3$ is the same as $4+4+4$. Therefor, the expression $4*True$ can be broken down into $True+True+True+True$. Let us add $True$ to this expression and get $True + True + True + True + True$, and simplify it to just $5 * True$ (by the distributive property of multiplication), which by our multiplication rule would evaluate to 5. By this logic, adding $True$ to a value is equivalent to adding 1, and adding $False$ is equivalent to adding 0. We can extend this to subtraction, so that subtracting $True$ subtracts 1 and $False$ subtracts 0

The division and multiplication operators are inverse operations. In other words, if a number is divided by number, x, multiplying the result by x would yield the original number. Naturally, this relationship should remain when using logical operands. However, this poses a major problem since if a value is multiplied by $False$, the output will be 0, and there is no number that $False$ can be converted into such that dividing 0 by it will yield the initial value (if the initial value is not 0). - How can/should this problem be reconciled?

3 Results

4 Conclusions