# Learning logical formulas

Lola Sofer-Yadgaroff

Supervisor: Cristina David

# Project Summary

The overall aim of the project was to be able to determine a logical formula (one which either evaluates to true or false) from data. After spending time looking at propositional logic, where data variables were either assigned either true or false, we extended this further to look at conjunctions/disjunctions of linear arithmetic inequalities and data that followed them. For example, the formula $(2x_1 + 3x_2 + 1 < 0) \wedge (5x_1 - x_2 + 1 > 0)$. For the case of the project, I auto-generated data to fit a certain formula and then our goal was to attempt to return a logical formula that would also fit the data.

In order to do this, I used a simple neural network to extract information about the formula. A neural network is a computational model that takes in inputs and through the modification and optimisation of certain parameters is able to recognise data patterns and returns an output. A neural network is made up of multiple layers, with each layer containing nodes and 'neuron pathways' connecting these nodes. These pathways will have weights and biases attached, and these weights and biases are optimised through processes called forward and backpropagation, which uses a technique similar to gradient descent.
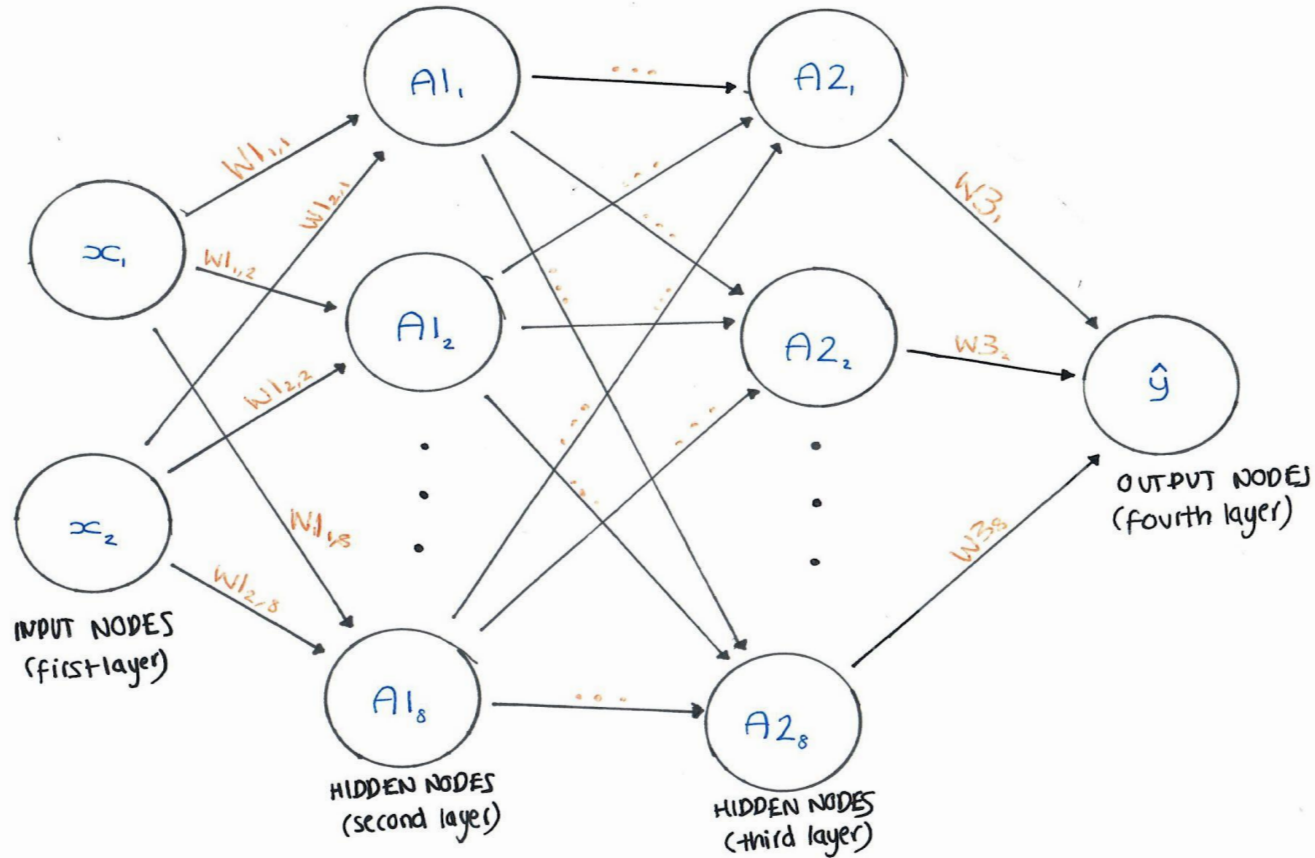
Usually, neural networks will use information about the relationship between data to return output but not return any of the information about the relationship, which is what we want as this would help generate a formula for the data. This meant we had to essentially look under the hood and study the architecture of the network. More specifically, I used the weights and biases to extract potential arithmetic expressions for the formula (Kobayashi, Sekiyama, Sato and Unno, 2021). After this, my program enumerates through potential formulas using optimised iteration and if a formula satisfies the data it is returned (a solver  is used to simplify this formula).

## Worked Example

My pairs of auto-generated data evaluate to true (1) if $(x_1 + 2x_2 - 5 > 0) \land (4x_1 - x_2 + 3 <= 0)$, and false (0) if not, and I will take you through how my program finds a logical formula which will evaluate to true or false in the same way. My program starts with the neural network optimising the weights and biasing, to improve the accuracy of the network. The accuracy is how many out of all the data input would the network predict the correct output for. In this example, the accuracy is 7982 out of 8000 (99.78%).

From the first set of weights and biases of the network, I can gather information about the ratios of coefficients for the arithmetic expressions of the form $c_0 + c_1x_1 + c_2x_2$ to be used in my final formula. The ratios for $c_0 : c_1 : c_2$ my program returned for this example were -2 : 1 : 2 , -3 : 1 : 2 , -1 : 1 : 2 , 1 : 2 : -1 and 2 : 2 : -1.

My program would then prioritises these and use the best distinct four or less to be used in the final formula. In this case, $2 + 2x_1 - x_2$, $-3 + x_1 + 2x_2$ and $-1 + x_1 + 2x_2$ are used. Finally my program, enumerations through potential conjunctions/disjunctions of possible inequalities of each expression, and prints the formula...
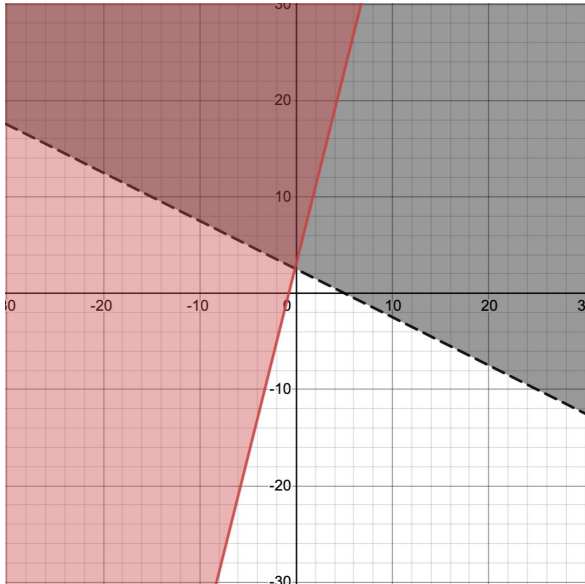
$$(2x_1 - x_2 <= -2) \lor ((-x_1 - 2x_2 <= -3) \land (x_1 + 2x_2 > 1))$$

with a total runtime of 803.29s
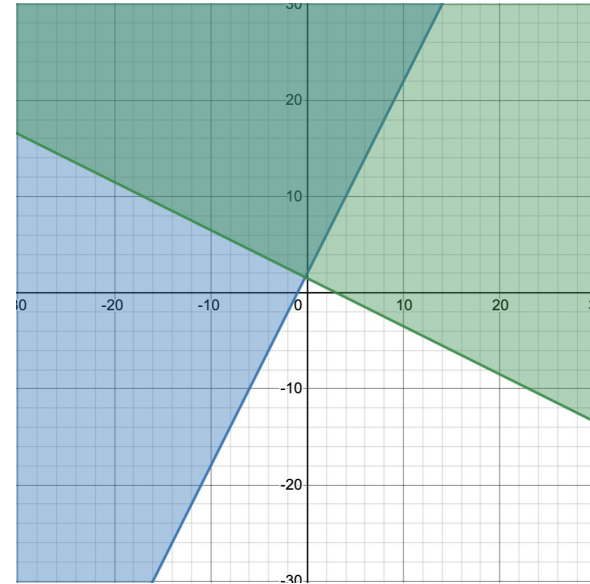
# Visual representation of worked example

The data followed the formula ...

$(x_1 + 2x_2 - 5 > 0) \wedge (4x_1 - x_2 + 3 <= 0)$



Formula proposed ....

$(2x_1 - x_2 <= -2) \vee ((-x_1 - 2x_2 <= -3) \wedge (x_1 + 2x_2 > 1))$

# More examples...

| Formula of data | Proposed formula | Accuracy (%) | Runtime (s) |
|---|---|---|---|
| (x1 + x2 -3 >= 0) $\wedge$ (4x1 -x2 + 2 < 0) | (x1 + x2 > 2) $\wedge$ (-4x1 + x2 > 3) $\wedge$ (-4x1 + x2 >= 3) $\wedge$ (2x1 + 3x2 > 8) | 99.67 | 651.4 |
| ((x1 + x2 -3 >= 0) $\wedge$ (4x1 -x2 + 2 < 0)) $\vee$ ( x1 + x2 > 0) | (x1 + x2 > 0) | 100.00 | 721.9 |
| ((x1 + 2x2 -3 >= 0) $\vee$ (4x1 -2x2 + 2 < 0)) $\wedge$ ( 5x1 + 3 <= 0) | ((-x1 -2x2 <= -3) $\wedge$ (x1 + 2x2 >1)) or (2x1 - x2 <= -2) | 99.78 | 824.4 |
| (-3x1 + 6x2 -2 >=0) | (-x1 + 2x2 > 0) | 100 .00 | 806.3 |
| ((x-31 +6 x2 -2 >= 0) $\wedge$ (x1 +x2 -5 >= 0)) $\wedge$ ((x1 + 7x2 -4 <= 0) $\wedge$ (x1 -2x2 +4 > 0)) | -- no formula found -- | 91.85 | 963.5 |

# Going forward...

Overall during the internship, I made good progress, and I was able to implement a program that returns a logical formula from data. However this is not always the case, e.g. example in the table from the previous slide, part of the next steps of the project would be to get to the bottom of why and alter this. Furthermore, I could optimise the program further not use enumeration, and even extend it to non-linear arithmetic formulas.

This kind of research also has applications in software verification. Logical formulas inferred from data such as data from program execution traces can be used to ensure that programs correctness. So automatically learning logical formulas automates the verification progress, which can be advantageous as programs get more complicated and therefore harder to authenticate.

Over the course of the project, I have learnt so much. Not only have I developed my programming skills but in particular I have developed a deeper understanding of neural networks and the algorithms that drive them. I have firsthand witnessed an application of one as well as the computational power they have. I hope to see and use more of them further in my studies.

*References :*

- *Kobayashi, N., Sekiyama, T., Sato, I. and Unno, H., 2021. Toward Neural-Network-Guided Program Synthesis and Verification.*